TensorFlow

Release Notes

RN-08467-001_v25.02 | July 2025

Table of Contents

Chapter 1. TensorFlow Overview

The NVIDIA® Deep Learning SDK accelerates widely-used deep learning frameworks such as TensorFlow™.

TensorFlow is an open-source software library for numerical computation by using data flow graphs. Nodes in the graph represent mathematical operations, and the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture allows you to deploy computations to one or more CPUs or GPUs in a desktop, server, or mobile device without rewriting code.

TensorFlow was originally developed by researchers and engineers who work on the Google Brain team in Google's Machine Intelligence research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains.

In a container, go to the `/workspace/README.md` directory for information about customizing your TensorFlow image. For more information about TensorFlow, including tutorials, documentation, and examples, see:

- TensorFlow tutorials
- TensorFlow API

This document provides information about the key features, software enhancements and improvements, known issues, and how to run this container.

Chapter 2. Pulling A Container

About this task

Before you can pull a container from the NGC container registry:

- Install Docker.
  - For NVIDIA DGX™ users, see Preparing to use NVIDIA Containers Getting Started Guide.
  - For non-DGX users, see NVIDIA® GPU Cloud™ (NGC) container registry installation documentation based on your platform.
- Ensure that you have access and can log in to the NGC container registry.

  Refer to NGC Getting Started Guide for more information.

The deep learning frameworks, the NGC Docker containers, and the deep learning framework containers are stored in the `nvcr.io/nvidia` repository.

Chapter 3. Running TensorFlow

Before you begin

Before you can run an NGC deep learning framework container, your Docker environment must support NVIDIA GPUs. To run a container, issue the appropriate command as explained in Running A Container and specify the registry, repository, and tags.

About this task

On a system with GPU support for NGC containers, when you run a container, the following occurs:

- The Docker engine loads the image into a container that runs the software.
- You define the runtime resources of the container by including the additional flags and settings that are used with the command.

  These flags and settings are described in Running A Container.
- The GPUs are explicitly defined for the Docker container, which defaults to all GPUs, but can be specified by using the `NVIDIA_VISIBLE_DEVICES` environment variable.

  For more information, refer to the nvidia-docker documentation.

> **Note:** Starting in Docker 19.03, complete the steps below.

The method implemented in your system depends on the DGX OS version that you installed (for DGX systems), the NGC Cloud Image that was provided by a Cloud Service Provider, or the software that you installed to prepare to run NGC containers on TITAN PCs, Quadro PCs, or NVIDIA Virtual GPUs (vGPUs).

Procedure

1. Issue the command for the applicable release of the container that you want. The following command assumes you want to pull the latest container.

   - **For TensorFlow version 2.x**

```
docker pull nvcr.io/nvidia/tensorflow:25.01-tf2-py3
```

2. Open a command prompt and paste the `pull` command.

   Ensure that the pull process successfully completes before proceeding to step 3.
3. Run the container image.

   - If you have **Docker 19.03 or later**, a typical command to launch the container is:

     ```
     docker run --gpus all -it --rm -v local_dir:container_dir nvcr.io/nvidia/
     tensorflow:<xx.xx>-tf<x>-py<x>
     ```
   - If you have **Docker 19.02 or earlier**, a typical command to launch the container is:

     ```
     nvidia-docker run -it --rm -v local_dir:container_dir nvcr.io/nvidia/
     tensorflow:<xx.xx>-tf<x>-py<x>
     ```

   To run TensorFlow, import it as a Python module:

   ```
   $ python
    >>> import tensorflow as tf
   >>> print(tf.__version__)
   1.15.0
   ```

   To pull data and model descriptions from locations outside the container for use by TensorFlow or save results to locations outside the container, mount one or more host directories as Docker® data volumes.

> **Note:** To share data between GPUs, NVIDIA Collective Communications Library (NCCL) might require shared system memory for IPC and pinned (page-locked) system memory resources, so the operating system's limits on these resources might need to be increased. Refer to your system's documentation for more information.
>
> In particular, Docker containers default to limited shared and pinned memory resources. When using NCCL inside a container, we recommed that you increase these resources by issuing the following command:
>
> ```
> --shm-size=1g --ulimit memlock=-1
> ```
>
> in the command line to:
>
> ```
> docker run --gpus all
> ```
>
> Similarly, on some Redhat Enterprise Linux (RHEL) systems, Docker limits the number of simultaneous PIDs in the container to 4096, which might be too small, particularly for multi-GPU training tasks. To increase this limit, pass the following option to `docker run`:
>
> `--pids-limit=8192`

Chapter 4. TensorFlow Release 25.02

The NVIDIA container image of TensorFlow, release 25.02, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Note: Deprecation notice: After the 25.02 release, NVIDIA Optimized TensorFlow containers will no longer be released. Known issues may be resolved in a future product based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 24.04.1 LTS

  Note: The `25.02-tf2-py3` container image contains Python 3.12.3.

- NVIDIA CUDA 12.8.0.38
- NVIDIA cuBLAS 12.8.3.14
- cuTENSOR 2.1.1.1
- NVIDIA cuDNN 9.7.1.26
- NVIDIA NCCL 2.25.1
- NVIDIA RAPIDS™ 24.12
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.18.0

- SHARP 3.0.2
- GDRCopy 2.4.1
- NVIDIA HPC-X 2.21
- TensorBoard 2.17.1
- rdma-core 50.0
- NVIDIA TensorRT™ 10.8.0.43
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.46
- Nsight Compute 2025.1.0.14
- Nsight Systems 2025.1.1.65
- nvImageCodec 0.3.0.5
- JupyterLab 4.3.4 including:
  - Jupyter-TensorBoard
  - Jupyter Client
  - Jupyter Core
  - Jupyter Notebook

Driver Requirements

Release 25.02 is based on CUDA 12.8.0.38 which requires NVIDIA Driver release 570 or later. However, if you are running on a data center GPU (for example, B100, L40, or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 550.54 (or later R550) in forward-compatibility mode.

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 and R560 drivers, which are not forward-compatible with CUDA 12.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 25.02 is based on TensorFlow 2.17.0.

Announcements

- Starting with the 25.01 release, the NVIDIA Optimized Deep Learning Framework containers are optimized for Blackwell GPU architectures.

- NVIDIA Optimized TensorFlow containers will no longer be released after the 25.02 release. Known issues may be resolved in a future product based on customer demand.
- Volta GPU compute architecture support is discontinued in the 25.01 release. For containers that are tested on Volta GPU please refer to NVIDIA AI Enterprise.
- Starting with the 24.09 release, NVIDIA TensorFlow ARM SBSA wheel is published on DevZone. You can install the latest release with this command: `pip install nvidia-tensorflow[horovod] --extra-index-url=https://pypi.ngc.nvidia.com/`.
- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
25.02	24.04	NVIDIA CUDA 12.8.0.38	2.17.0	TensorRT 10.8.0.43
25.01		NVIDIA CUDA 12.8.0.038	2.17.0	TensorRT 10.8.0.40
24.12		NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.7.0.23
24.11		NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.6.0.26
24.10	22.04	NVIDIA CUDA 12.6.2	2.16.1	TensorRT 10.5.0.18
24.09		NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.

- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 5. TensorFlow Release 25.01

The NVIDIA container image of TensorFlow, release 25.01, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Note: Deprecation notice: After the 25.02 release, NVIDIA Optimized TensorFlow containers will no longer be released. Known issues may be resolved in a future product based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 24.04.1 LTS

  Note: The `25.01-tf2-py3` container image contains Python 3.12.3.

- NVIDIA CUDA 12.8.0
- NVIDIA cuBLAS 12.8.3.14
- cuTENSOR 2.1.0.9
- NVIDIA cuDNN 9.7.0.66
- NVIDIA NCCL 2.25.1
- NVIDIA RAPIDS™ 24.10
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.18.0

- SHARP 3.0.2
- GDRCopy 2.4.1
- NVIDIA HPC-X 2.21
- TensorBoard 2.17.1
- rdma-core 50.0
- NVIDIA TensorRT™ 10.8.0.40
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.45
- Nsight Compute 2025.1.0.14
- Nsight Systems 2024.6.2.225
- nvImageCodec 0.3.0.5
- JupyterLab 4.3.4 including:
  - Jupyter-TensorBoard
  - Jupyter Client
  - Jupyter Core
  - Jupyter Notebook

Driver Requirements

Release 25.01 is based on CUDA 12.8.0 which requires NVIDIA Driver release 570 or later. However, if you are running on a data center GPU (for example, B100, L40, or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 550.54 (or later R550) in forward-compatibility mode.

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 and R560 drivers, which are not forward-compatible with CUDA 12.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 25.01 is based on TensorFlow 2.17.0.

Announcements

- Starting with the 25.01 release, the NVIDIA Optimized Deep Learning Framework containers are optimized for Blackwell GPU architectures.

- NVIDIA Optimized TensorFlow containers will no longer be released after the 25.02 release. Known issues may be resolved in a future product based on customer demand.
- Volta GPU compute architecture support is discontinued in the 25.01 release. For containers that are tested on Volta GPU please refer to NVIDIA AI Enterprise.
- Starting with the 24.09 release, NVIDIA TensorFlow ARM SBSA wheel is published on DevZone. You can install the latest release with this command: `pip install nvidia-tensorflow[horovod] --extra-index-url=https://pypi.ngc.nvidia.com/`.
- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
25.01	24.04	NVIDIA CUDA 12.8.0.38	2.17.1	TensorRT 10.8.0.40
24.12		NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.7.0.23
24.11		NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.6.0.26
24.10	22.04	NVIDIA CUDA 12.6.2	2.16.1	TensorRT 10.5.0.18
24.09		NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.

- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 6. TensorFlow Release 24.12

The NVIDIA container image of TensorFlow, release 24.12, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 24.04

  Note: The `24.12-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.6.3
- NVIDIA cuBLAS 12.6.4.1
- cuTENSOR 2.0.2.5
- NVIDIA cuDNN 9.6.0.74
- NVIDIA NCCL 2.23.4
- NVIDIA RAPIDS™ 24.10
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.18.0
- SHARP 3.0.2
- GDRCopy 2.4.1
- NVIDIA HPC-X 2.21

- TensorBoard 2.16.2
- rdma-core 39.0
- NVIDIA TensorRT™ 10.7.0.23
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.44
- Nsight Compute 2024.3.2.3
- Nsight Systems 2024.7.1.84
- nvImageCodec 0.2.0.7
- JupyterLab 4.2.5 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.12 is based on CUDA 12.6.3 which requires NVIDIA Driver release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.12 is based on TensorFlow 2.17.0.

Announcements

- NVIDIA Optimized TensorFlow containers will no longer be released after the 25.02 release. Known issues may be resolved in a future product based on customer demand.
- Starting with the 24.10 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Volta GPU architectures. For containers that are tested on Volta GPU please refer to NVIDIA AI Enterprise. Volta GPU compute architecture support will be discontinued by the 25.01 release.

- Starting with the 24.09 release, NVIDIA TensorFlow ARM SBSA wheel is published on DevZone. You can install the latest release with this command: `pip install nvidia-tensorflow[horovod] --extra-index-url=https://pypi.ngc.nvidia.com/`.
- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.12	24.04	NVIDIA CUDA 12.6.3	2.18.0	TensorRT 10.7.0.23
24.11		NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.6.0.26
24.10	22.04	NVIDIA CUDA 12.6.2	2.16.1	TensorRT 10.5.0.18
24.09		NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.

- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 7. TensorFlow Release 24.11

The NVIDIA container image of TensorFlow, release 24.11, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 24.04

  > **Note:** The `24.11-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.6.3
- NVIDIA cuBLAS 12.6.4.1
- cuTENSOR 2.0.2.5
- NVIDIA cuDNN 9.5.1.17
- NVIDIA NCCL 2.23.4
- NVIDIA RAPIDS™ 24.10
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.18.0
- SHARP 3.0.2
- GDRCopy 2.4.1
- NVIDIA HPC-X 2.21

- TensorBoard 2.16.2
- rdma-core 39.0
- NVIDIA TensorRT™ 10.6.0.26
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.43
- Nsight Compute 2024.3.2.3
- Nsight Systems 2024.4.6.1.90
- nvImageCodec 0.2.0.7
- JupyterLab 4.2.5 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.11 is based on CUDA 12.6.3 which requires NVIDIA Driver release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.11 is based on TensorFlow 2.17.0.

Announcements

- NVIDIA Optimized TensorFlow containers will no longer be released after the 25.02 release. Known issues may be resolved in a future product based on customer demand.
- Starting with the 24.10 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Volta GPU architectures. For containers that are tested on Volta GPU please refer to NVIDIA AI Enterprise. Volta GPU compute architecture support will be discontinued by the 25.01 release.

- Starting with the 24.09 release, NVIDIA TensorFlow ARM SBSA wheel is published on DevZone. You can install the latest release with this command: `pip install nvidia-tensorflow[horovod] --extra-index-url=https://pypi.ngc.nvidia.com/`.
- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.11	24.04	NVIDIA CUDA 12.6.3	2.17.0	TensorRT 10.6.0.26
24.10	22.04	NVIDIA CUDA 12.6.2	2.16.1	TensorRT 10.5.0.18
24.09		NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2	TensorRT 8.2.1.8

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.5	
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.2	
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

The workaround is to run the following command:

```
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
```

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 8. TensorFlow Release 24.10

The NVIDIA container image of TensorFlow, release 24.10, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `24.10-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.6.2
- NVIDIA cuBLAS 12.6.3.3
- cuTENSOR 2.0.2.5
- NVIDIA cuDNN 9.5.0.50
- NVIDIA NCCL 2.22.3
- NVIDIA RAPIDS™ 24.08
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.20

- TensorBoard 2.16.2
- rdma-core 39.0
- NVIDIA TensorRT™ 10.5.0.18
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.42
- Nsight Compute 2024.3.2.3
- Nsight Systems 2024.4.6.1.90
- nvImageCodec 0.2.0.7
- JupyterLab 4.2.5 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.10 is based on CUDA 12.6.2 which requires NVIDIA Driver release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.10 is based on TensorFlow 2.16.1.

Announcements

- Starting with the 24.10 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Volta GPU architectures. For containers that are tested on Volta GPU please refer to NVIDIA AI Enterprise.
- Starting with the 24.09 release, NVIDIA TensorFlow ARM SBSA wheel is published on DevZone. You can install the latest release with this command: `pip install nvidia-tensorflow[horovod] --extra-index-url=https://pypi.ngc.nvidia.com/`.
- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes

FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.10	22.04	NVIDIA CUDA 12.6.2	2.16.1	TensorRT 10.5.0.18
24.09		NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 9. TensorFlow Release 24.09

The NVIDIA container image of TensorFlow, release 24.09, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `24.09-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.6.1
- NVIDIA cuBLAS 12.6.3.1
- cuTENSOR 2.0.2.5
- NVIDIA cuDNN 9.4.0.58
- NVIDIA NCCL 2.22.3
- NVIDIA RAPIDS™ 24.08
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.20

- TensorBoard 2.16.2
- rdma-core 39.0
- NVIDIA TensorRT™ 10.4.0.26
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.41
- Nsight Compute 2024.3.1.2
- Nsight Systems 2024.4.2.133
- nvImageCodec 0.2.0.7
- JupyterLab 4.2.5 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.09 is based on CUDA 12.6.1 which requires NVIDIA Driver release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.08 is based on TensorFlow 2.16.1.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.09	22.04	NVIDIA CUDA 12.6.1	2.16.1	TensorRT 10.4.0.26
24.08		NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0	TensorRT 7.2.3.4

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.5	
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.

- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 10. TensorFlow Release 24.08

The NVIDIA container image of TensorFlow, release 24.08, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `24.08-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.6
- NVIDIA cuBLAS 12.6.0.22
- cuTENSOR 2.0.2.5
- NVIDIA cuDNN 9.3.0.75
- NVIDIA NCCL 2.22.3
- NVIDIA RAPIDS™ 24.06
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.19

- TensorBoard 2.16.2
- rdma-core 39.0
- NVIDIA TensorRT™ 10.3.0.26
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.40
- Nsight Compute 2024.3.0.15
- Nsight Systems 2024.4.2.133
- nvImageCodec 0.2.0.7
- JupyterLab 4.2.4 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.08 is based on CUDA 12.6 which requires NVIDIA Driver release 560 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520, R530, R545 and R555 drivers, which are not forward-compatible with CUDA 12.6. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.08 is based on TensorFlow 2.16.1.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.08	22.04	NVIDIA CUDA 12.6	2.16.1	TensorRT 10.3.0.26
24.07		NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 11. TensorFlow Release 24.07

The NVIDIA container image of TensorFlow, release 24.07, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `24.07-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.5.1
- NVIDIA cuBLAS 12.5.3.2
- cuTENSOR 2.0.2.4
- NVIDIA cuDNN 9.2.1.18
- NVIDIA NCCL 2.22.3
- NVIDIA RAPIDS™ 24.04
- Horovod 0.28.1
- OpenMPI 4.1.7
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.19

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 10.2.0.19
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.39
- Nsight Compute 2024.2.1.2
- Nsight Systems 2024.4.2.133
- nvImageCodec 0.2.0.7
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.07 is based on CUDA 12.5.1 which requires NVIDIA Driver release 555 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, R520 and R545 drivers, which are not forward-compatible with CUDA 12.5. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.07 is based on TensorFlow 2.16.1.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.07	22.04	NVIDIA CUDA 12.5.1	2.16.1	TensorRT 10.2.0.19
24.06		NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```

- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 12. TensorFlow Release 24.06

The NVIDIA container image of TensorFlow, release 24.06, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `24.06-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.5.0.23
- NVIDIA cuBLAS 12.5.2.13
- cuTENSOR 2.0.1.2
- NVIDIA cuDNN 9.1.0.70
- NVIDIA NCCL 2.21.5
- NVIDIA RAPIDS™ 24.04
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.19

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 10.1.0.27
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.38
- Nsight Compute 2024.2.0.16
- Nsight Systems 2024.2.3.38
- nvImageCodec 0.2.0.7
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.06 is based on NVIDIA CUDA 12.5.0.23, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.06 is based on TensorFlow 2.16.1.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.06	22.04	NVIDIA CUDA 12.5.0.23	2.16.1	TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0	TensorRT 8.0.1.6

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.07		NVIDIA CUDA 11.4.0	1.15.5	
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```

- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 13. TensorFlow Release 24.05

The NVIDIA container image of TensorFlow, release 24.05, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `24.05-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.4.1
- NVIDIA cuBLAS 12.4.5.8
- cuTENSOR 2.0
- NVIDIA cuDNN 9.1.0.70
- NVIDIA NCCL 2.21.5
- NVIDIA RAPIDS™ 24.04
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.19

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 10.0.1.6
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.37.1
- Nsight Compute 2024.1.1.4
- Nsight Systems 2024.2.1.106
- nvImageCodec 0.2.0.7
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.05 is based on NVIDIA CUDA 12.4.1, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.05 is based on TensorFlow 2.15.0.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.05	22.04	NVIDIA CUDA 12.4.1	2.15.0	TensorRT 10.0.1.6
24.04			2.15.0	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0	TensorRT 8.5.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.02		NVIDIA CUDA 12.0.1	1.15.5	
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```

- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 14. TensorFlow Release 24.04

The NVIDIA container image of TensorFlow, release 24.04, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `24.04-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.4.1
- NVIDIA cuBLAS 12.4.5.8
- cuTENSOR 2.0
- NVIDIA cuDNN 9.1.0.70
- NVIDIA NCCL 2.21.5
- NVIDIA RAPIDS™ 24.02
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.18

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.3
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.36
- Nsight Compute 2024.1.1.4
- Nsight Systems 2024.2.1.106
- nvImageCodec 0.2.0.7
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.04 is based on NVIDIA CUDA 12.4.1, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.04 is based on TensorFlow 2.15.0.

Announcements

- Starting with the 24.03 release, NVIDIA/Transformer Engine will no longer be included with NVIDIA Optimized TensorFlow containers. Transformer Engine includes FP8 support to accelerate training LLM models with Ada and Hopper GPUs. We recommend using float8 training supported in Keras via XLA (PR).
- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.

- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>24.04</td><td rowspan="12">22.04</td><td>NVIDIA CUDA 12.4.1</td><td rowspan="3">2.15.0</td><td rowspan="3">TensorRT 8.6.3</td></tr>
<tr><td>24.03</td><td>NVIDIA CUDA 12.4.0.41</td></tr>
<tr><td>24.02</td><td rowspan="3">NVIDIA CUDA 12.3.2</td></tr>
<tr><td>24.01</td><td rowspan="3">2.14.0</td><td rowspan="8">TensorRT 8.6.1.6</td></tr>
<tr><td>23.12</td></tr>
<tr><td>23.11</td><td>NVIDIA CUDA 12.3.0</td></tr>
<tr><td>23.10</td><td>NVIDIA CUDA 12.2.1</td><td rowspan="3">2.13.0</td></tr>
<tr><td>23.09</td><td rowspan="2">NVIDIA CUDA 12.2.1</td></tr>
<tr><td>23.08</td></tr>
<tr><td>23.07</td><td rowspan="3">NVIDIA CUDA 12.1.1</td><td rowspan="4">2.12.0</td></tr>
<tr><td>23.06</td></tr>
<tr><td>23.05</td><td>TensorRT 8.6.1.2</td></tr>
<tr><td>23.04</td><td rowspan="2">20.04</td><td rowspan="2">NVIDIA CUDA 12.1.0</td><td>TensorRT 8.6.1</td></tr>
<tr><td>23.03</td><td>2.11.0</td><td>TensorRT 8.5.3</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.02		NVIDIA CUDA 12.0.1	1.15.5	
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```

- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 15. TensorFlow Release 24.03

The NVIDIA container image of TensorFlow, release 24.03, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `24.03-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA 12.4.0.41
- NVIDIA cuBLAS 12.4.2.65
- cuTENSOR 2.0
- NVIDIA cuDNN 9.0.0.306
- NVIDIA NCCL 2.20
- NVIDIA RAPIDS™ 24.02
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.18

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.3
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.35
- Nsight Compute 2024.1.0.13
- Nsight Systems 2024.2.1.38
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.03 is based on NVIDIA CUDA 12.4.0.41, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.03 is based on TensorFlow 2.15.0.

Announcements

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.
- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.

- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.03	22.04	NVIDIA CUDA 12.4.0.41	2.15	TensorRT 8.6.3
24.02		NVIDIA CUDA 12.3.2		
24.01			2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.10			1.15.5	TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.

- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 16. TensorFlow Release 24.02

The NVIDIA container image of TensorFlow, release 24.02, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `24.02-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuTENSOR 2.0
- NVIDIA cuDNN 9.0.0.306
- NVIDIA NCCL 2.19.4
- NVIDIA RAPIDS™ 23.12
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.16rc4

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.3
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.34
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1.97
- JupyterLab 2.3.2 including:
    - Jupyter-TensorBoard
    - Jupyter Client 8.6.0
    - Jupyter Core 5.5.0
    - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.02 is based on CUDA 12.3.2, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.02 is based on TensorFlow 2.15.0.

Announcements

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.
- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.

- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>24.02</td><td rowspan="10">22.04</td><td rowspan="3">NVIDIA CUDA 12.3.2</td><td></td><td>TensorRT 8.6.3</td></tr>
<tr><td>24.01</td><td rowspan="3">2.14.0</td><td rowspan="8">TensorRT 8.6.1.6</td></tr>
<tr><td>23.12</td></tr>
<tr><td>23.11</td><td>NVIDIA CUDA 12.3.0</td></tr>
<tr><td>23.10</td><td>NVIDIA CUDA 12.2.1</td><td rowspan="3">2.13.0</td></tr>
<tr><td>23.09</td><td rowspan="2">NVIDIA CUDA 12.2.1</td></tr>
<tr><td>23.08</td></tr>
<tr><td>23.07</td><td rowspan="3">NVIDIA CUDA 12.1.1</td><td rowspan="4">2.12.0</td></tr>
<tr><td>23.06</td></tr>
<tr><td>23.05</td><td>TensorRT 8.6.1.2</td></tr>
<tr><td>23.04</td><td rowspan="8">20.04</td><td rowspan="2">NVIDIA CUDA 12.1.0</td><td>TensorRT 8.6.1</td></tr>
<tr><td>23.03</td><td rowspan="3">2.11.0
1.15.5</td><td rowspan="2">TensorRT 8.5.3</td></tr>
<tr><td>23.02</td><td rowspan="2">NVIDIA CUDA 12.0.1</td></tr>
<tr><td>23.01</td><td>TensorRT 8.5.2.2</td></tr>
<tr><td>22.12</td><td rowspan="4">NVIDIA CUDA 11.8.0</td><td>2.10.1
1.15.5</td><td rowspan="2">TensorRT 8.5.1</td></tr>
<tr><td>22.11</td><td rowspan="2">2.10.0
1.15.5</td></tr>
<tr><td>22.10</td><td rowspan="2">TensorRT 8.5 EA</td></tr>
<tr><td>22.09</td><td>2.9.1</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.08		NVIDIA CUDA 11.7.1	1.15.5	TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>20.12</td><td></td><td>NVIDIA CUDA 11.1.1</td><td rowspan="3">2.3.1
1.15.4</td><td>TensorRT 7.2.2</td></tr>
<tr><td>20.11</td><td rowspan="12">18.04</td><td rowspan="2">NVIDIA CUDA 11.1.0</td><td rowspan="2">TensorRT 7.2.1</td></tr>
<tr><td>20.10</td></tr>
<tr><td>20.09</td><td rowspan="2">NVIDIA CUDA 11.0.3</td><td>2.3.0
1.15.3</td><td rowspan="3">TensorRT 7.1.3</td></tr>
<tr><td>20.08</td><td rowspan="2">2.2.0
1.15.3</td></tr>
<tr><td>20.07</td><td>NVIDIA CUDA 11.0.194</td></tr>
<tr><td>20.06</td><td>NVIDIA CUDA 11.0.167</td><td>2.2.0
1.15.2</td><td>TensorRT 7.1.2</td></tr>
<tr><td>20.03
20.02</td><td rowspan="3">NVIDIA CUDA 10.2.89</td><td>2.1.0
1.15.2</td><td rowspan="2">TensorRT 7.0.0</td></tr>
<tr><td>20.01</td><td rowspan="2">2.0.0
1.15.0</td></tr>
<tr><td>19.12
19.11</td><td rowspan="3">TensorRT 6.0.1</td></tr>
<tr><td>19.10</td><td rowspan="3">NVIDIA CUDA 10.1.243</td><td rowspan="3">1.14.0</td></tr>
<tr><td>19.09</td></tr>
<tr><td>19.08</td><td>TensorRT 5.1.5</td></tr>
</table>

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is up to 22% perf regression for efficientnet training in TF that affects some GPUs (A2, A10, A40, L40, V100). This is under investigation.

- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation.

Chapter 17. TensorFlow Release 24.01

The NVIDIA container image of TensorFlow, release 24.01, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `24.01-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.7.29
- NVIDIA NCCL 2.19.3
- NVIDIA RAPIDS™ 23.12
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.16rc4

- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.33
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1.97
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 24.01 is based on CUDA 12.3.2, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 470.57 (or later R470), 525.85 (or later R525), 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R450, R460, R510, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 24.01 is based on TensorFlow 2.14.0.

Announcements

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.
- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.

- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
24.01	22.04	NVIDIA CUDA 12.3.2	2.14.0	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known CUPTI permissions issue that is new in the 24.01 iGPU container. This issue prevents the profiler from being able to capture cuda events and manifests itself as a CUPTI Runtime Error with error code 35. This may be worked around by running the following command:

  ```
  rm -rf /usr/local/cuda/compat/lib.real
  ```
- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.

- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 18. TensorFlow Release 23.12

The NVIDIA container image of TensorFlow, release 23.12, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `23.12-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.7.29
- NVIDIA NCCL 2.19.3
- NVIDIA DALI® 1.31.0
- NVIDIA RAPIDS™ 23.08
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.16
- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.32
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.3.4.1
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 23.12 is based on CUDA 12.3.2, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525) 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.12 is based on TensorFlow 2.14.0.

Announcements

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.
- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.12	22.04	NVIDIA CUDA 12.3.2	2.14.0	TensorRT 8.6.1.6
23.11		NVIDIA CUDA 12.3.0	2.14.0	
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known performance drop of about 10% with ResNet. This is under investigation.
- There is a known performance drop of about 40% with efficientdet. This is under investigation.
- Several networks crash with a CUDA OOM when used in multi-GPU configuration using NVLS. This could be a possible TF memory carveout issue, and it is suggested to try increasing the carveout using TF_DEVICE_MIN_SYS_MEMORY_IN_MB.
- There is a known performance drop with Electra on V100 and T4 GPUs. This is under investigation.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.

- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 19. TensorFlow Release 23.11

The NVIDIA container image of TensorFlow, release 23.11, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `23.11-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.3.0
- NVIDIA cuBLAS 12.3.2.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.6
- NVIDIA NCCL 2.19.3
- NVIDIA DALI® 1.31.0
- NVIDIA RAPIDS™ 23.08
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.16
- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.31.0
- Nsight Compute 2023.3.0.12
- Nsight Systems 2023.3.1.92
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.6.0
  - Jupyter Core 5.5.0
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 23.11 is based on CUDA 12.3.0, which requires NVIDIA Driver release 545 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525) 535.86 (or later R535), or 545.23 (or later R545).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.3. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.11 is based on TensorFlow 2.14.0.

Announcements

- Starting with the 23.11 release, NVIDIA Optimized TensorFlow containers supporting iGPU architectures are published, and run on Jetson devices. Please refer to the Frameworks Support Matrix for information regarding which iGPU hardware/software is supported by which container.
- Starting with the 23.11 release, numpy has been updated to v1.24. This version removed some deprecated APIs. Here is a list of APIs that have expired.

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.11	22.04	NVIDIA CUDA 12.3.0	2.14.0	TensorRT 8.6.1.6
23.10		NVIDIA CUDA 12.2.1	2.13.0	
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known performance drop on many networks in XLA mode due to TF picking OAI triton gemm kernels over cuBLAS kernels. Performance can be restored by disabling the use of OAI triton kernels in TF by setting the following TF environment variable. `XLA_FLAGS="--xla_gpu_enable_triton_gemm=false "`.
- There is a known issue with installing mpi4py in the TF 23.11 container.
  To install mpi4py successfully, please unset the environment variable SETUPTOOLS_USE_DISTUTILS using unset SETUPTOOLS_USE_DISTUTILS.
- Efficientnet is presently not compatible with horovod, resulting in application crash.
- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.

- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 20. TensorFlow Release 23.10

The NVIDIA container image of TensorFlow, release 23.10, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `23.10-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.2.2
- NVIDIA cuBLAS 12.2.5.6
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.5
- NVIDIA NCCL 2.19.3
- NVIDIA DALI® 1.30.0
- NVIDIA RAPIDS™ 23.08
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.16
- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.3.1.92
- JupyterLab 2.3.2 including:
  - Jupyter-TensorBoard
  - Jupyter Client 8.3.1
  - Jupyter Core 5.3.2
  - Jupyter Notebook 6.4.10

Driver Requirements

Release 23.10 is based on CUDA 12.2.2, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.10 is based on TensorFlow 2.13.0.

Announcements

- Starting with the 23.10 release, NVIDIA TensorFlow containers supporting integrated GPU embedded systems will be published.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base

OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.10	22.04	NVIDIA CUDA 12.2.2	2.13.0	TensorRT 8.6.1.6
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.07		NVIDIA CUDA 11.0.194	1.15.3	
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- Efficientnet is presently not compatible with horovod, resulting in application crash. This will be fixed in a future release.
- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 21. TensorFlow Release 23.09

The NVIDIA container image of TensorFlow, release 23.09, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `23.09-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.2.1
- NVIDIA cuBLAS 12.2.5.6
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.5
- NVIDIA NCCL 2.18.5
- NVIDIA DALI® 1.29.0
- NVIDIA RAPIDS™ 23.08
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.16
- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.3.1.92
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 23.09 is based on CUDA 12.2.1, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.09 is based on TensorFlow 2.13.0.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.09	22.04	NVIDIA CUDA 12.2.1	2.13.0	TensorRT 8.6.1.6
23.08				
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12 19.11			1.15.0	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- Efficientnet is presently not compatible with horovod, resulting in application crash. This will be fixed in a future release.
- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 22. TensorFlow Release 23.08

The NVIDIA container image of TensorFlow, release 23.08, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `23.08-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.2.1
- NVIDIA cuBLAS 12.2.5
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.4
- NVIDIA NCCL 2.18.3
- NVIDIA DALI® 1.28.0
- NVIDIA RAPIDS™ 23.06
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.15
- TensorBoard 2.13.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.2.3.1001
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 23.08 is based on CUDA 12.2.1, which requires NVIDIA Driver release 535 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 525.85 (or later R525), or 535.86 (or later R535).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.2. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.08 is based on TensorFlow 2.13.0.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.08	22.04	NVIDIA CUDA 12.2.1	2.13.0	TensorRT 8.6.1.6
23.07		NVIDIA CUDA 12.1.1	2.12.0	
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 23. TensorFlow Release 23.07

The NVIDIA container image of TensorFlow, release 23.07, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `23.07-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.3
- NVIDIA NCCL 2.18.3
- NVIDIA DALI® 1.27.0
- NVIDIA RAPIDS™ 23.06
- Horovod 0.28.1
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.15
- TensorBoard 2.12.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.6
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.2.3.1001
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 23.07 is based on CUDA 12.1.1, which requires NVIDIA Driver release 530 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), 525.85 (or later R525), or 530.30 (or later R530).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.1. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.07 is based on TensorFlow 2.12.0.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.07	22.04	NVIDIA CUDA 12.1.1	2.12.0	TensorRT 8.6.1.6
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

Known Issues

- There is a known cuDNN performance regression that can reduce performance by up to 30% for the EfficientNet model on H100. This will be fixed in a future release.
- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. This will be fixed in an upcoming release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.

Chapter 24. TensorFlow Release 23.06

The NVIDIA container image of TensorFlow, release 23.06, is available on NGC.

> **Note:** Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  > **Note:** The `23.06-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.2
- NVIDIA NCCL 2.18.1
- NVIDIA DALI® 1.26.0
- NVIDIA RAPIDS™ 23.04
- Horovod 0.28.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.15
- TensorBoard 2.12.0
- rdma-core 39.0
- NVIDIA TensorRT™ 8.6.1.2
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.2.3.1001
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 23.06 is based on CUDA 12.1.1, which requires NVIDIA Driver release 530 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), 525.85 (or later R525), or 530.30 (or later R530).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.1. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.06 is based on TensorFlow 2.12.0.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- As of the 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.06	22.04	NVIDIA CUDA 12.1.1	2.12.0	TensorRT 8.6.1.6
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

Known Issues

- The TensorFlow DLRM model may see a performance regression of up to 30% on A40 GPUs compared to the 23.05 release. This will be fixed in a future release.
- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. The root cause is under investigation and will be fixed in a later release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

Chapter 25. TensorFlow Release 23.05

The NVIDIA container image of TensorFlow, release 23.05, is available on NGC.

Note: Deprecation notice: As of the 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 22.04

  Note: The `23.05-tf2-py3` container image contains Python 3.10.6.

- NVIDIA CUDA® 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuTENSOR 1.7.0.1
- NVIDIA cuDNN 8.9.1.23
- NVIDIA NCCL 2.17.1
- NVIDIA DALI® 1.25.0
- NVIDIA RAPIDS™ 23.04
- Horovod 0.27.0
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.14
- TensorBoard 2.12.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.6.1.2
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.12
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 23.05 is based on CUDA 12.1.1, which requires NVIDIA Driver release 530 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), 525.85 (or later R525), or 530.30 (or later R530).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.1. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.05is based on TensorFlow 2.12.0.

Announcements

- As of the current 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.05	22.04	NVIDIA CUDA 12.1.1	2.12.0	TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

Known Issues

- An illegal memory access violation is exposed in TensorFlow 2.12 by the Electra model as implemented in JoC. The root cause is under investigation and will be fixed in a later release.
- Up to 99% perf regressions across all EfficientDet model configs.
- Collecting profiles with the native TensorFlow profiler may result in an application crash with the error “double free or corruption” due to a bug in the CUPTI library. This will be fixed in a future release.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

Chapter 26. TensorFlow Release 23.04

The NVIDIA container image of TensorFlow, release 23.04, is available on NGC.

> **Note:** Deprecation notice: As of the current 23.04 release, TF1 is no longer released monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `23.04-tf2-py3` container image contains Python 3.8.

- NVIDIA CUDA® 12.1.0
- NVIDIA cuBLAS 12.1.3
- cuTENSOR 1.7.0
- NVIDIA cuDNN 8.9.0
- NVIDIA NCCL 2.17.1
- NVIDIA DALI® 1.24.0
- NVIDIA RAPIDS™ 23.02
- Horovod 0.27.0
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.13
- TensorBoard 2.12.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.6.1
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.1.0.15
- Nsight Systems 2023.1.1.127
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 23.04 is based on CUDA 12.1.0, which requires NVIDIA Driver release 530 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), 525.85 (or later R525), or 530.30 (or later R530).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.1. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 23.04 is based on TensorFlow 2.12.0.

Announcements

- As of the current 23.04 release, TF1 is no longer released monthly. Known issues may be solved in a future release based on customer demand.
- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.04	20.04	NVIDIA CUDA 12.1.0	2.12.0	TensorRT 8.6.1
23.03			2.11.0 1.15.5	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1		
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

Known Issues

- Up to 99% perf regressions across all EfficientDet model configs.
- Collecting profiles with the native TensorFlow profiler may result in an application crash with the error “double free or corruption” due to a bug in the CUPTI library. This will be fixed in a future release.
- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Using the old optimizers (available now under tf.keras.optimizers.legacy) resolves the errors.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

Chapter 27. TensorFlow Release 23.03

The NVIDIA container image of TensorFlow, release 23.03, is available on NGC.

> **Note:** Deprecation notice: After the current 23.03 release, TF1 will no longer release monthly. Known issues may be resolved in a future release based on customer demand.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `23.03-tf1-py3` and `23.03-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 12.1.0
- NVIDIA cuBLAS from CUDA 12.1.0
- cuTENSOR 1.6.2.3
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.17.1
- NVIDIA DALI® 1.23.0
- NVIDIA RAPIDS™ 23.02
- Horovod 0.27.0
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3

- NVIDIA HPC-X 2.13
- TensorBoard
  - `23.03-tf1-py3` includes version 1.15.5
  - `23.03-tf2-py3` includes version 2.11.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.3
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2023.1.0.15
- Nsight Systems 2023.1.1.127
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 23.03 is based on CUDA 12.1.0, which requires NVIDIA Driver release 530 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), 525.85 (or later R525), or 530.30 (or later R530).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.1. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 23.03 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 23.03 are based on TensorFlow 1.15.5 and 2.11.0.

Announcements

- After the 23.03 release, TF1 will no longer release monthly. Known issues may be solved in a future release based on customer demand.

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>23.03</td><td rowspan="12">20.04</td><td>NVIDIA CUDA 12.1.0</td><td rowspan="3">2.11.0
1.15.5</td><td rowspan="2">TensorRT 8.5.3</td></tr>
<tr><td>23.02</td><td rowspan="2">NVIDIA CUDA 12.0.1</td></tr>
<tr><td>23.01</td><td>TensorRT 8.5.2.2</td></tr>
<tr><td>22.12</td><td rowspan="4">NVIDIA CUDA 11.8.0</td><td>2.10.1
1.15.5</td><td rowspan="2">TensorRT 8.5.1</td></tr>
<tr><td>22.11</td><td rowspan="2">2.10.0
1.15.5</td></tr>
<tr><td>22.10</td><td rowspan="2">TensorRT 8.5 EA</td></tr>
<tr><td>22.09</td><td rowspan="4">2.9.1
1.15.5</td></tr>
<tr><td>22.08</td><td>NVIDIA CUDA 11.7.1</td><td>TensorRT 8.4.2.4</td></tr>
<tr><td>22.07</td><td rowspan="2">NVIDIA CUDA 11.7 Update 1 Preview</td><td>TensorRT 8.4.1</td></tr>
<tr><td>22.06</td><td rowspan="2">TensorRT 8.2.5</td></tr>
<tr><td>22.05</td><td>NVIDIA CUDA 11.7.0</td><td rowspan="2">2.8.0
1.15.5</td></tr>
<tr><td>22.04</td><td>NVIDIA CUDA 11.6.2</td><td>TensorRT 8.2.4.2</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.07		NVIDIA CUDA 11.0.194	1.15.3	
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.

- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod

- Tensor Cores (mixed precision) training
- Static loss scaling for Tensor Cores (mixed precision) training

This model script is available on GitHub and NGC.

Known Issues

- Collecting profiles with the native TensorFlow profiler may result in an application crash with the error "double free or corruption" due to an apparent bug in the CUPTI library. The root cause is under investigation and will be fixed in a future release.
- CUDNN autotuning in XLA can take up to 9x longer than in previous releases for certain combinations of fused layers. We are working to improve this in a future release.
- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Using the old optimizers (available now under tf.keras.optimizers.legacy) resolves the errors.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal

instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access.

Chapter 28. TensorFlow Release 23.02

The NVIDIA container image of TensorFlow, release 23.02, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  Note: The `23.02-tf1-py3` and `23.02-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 12.0.1
- NVIDIA cuBLAS from CUDA 12.0.1
- cuTENSOR 1.6.2.3
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.16.5
- NVIDIA DALI® 1.22.0
- NVIDIA RAPIDS™ 22.12.0
- Horovod 0.26.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- TensorBoard
  - `23.02-tf1-py3` includes version 1.15.5

- `23.02-tf2-py3` includes version 2.11.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.3
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2022.4.1.6
- Nsight Systems 2022.5.1.93
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 23.02 is based on CUDA 12.0.1, which requires NVIDIA Driver release 525 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), or 525.85 (or later R525).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, R460, and R520 drivers, which are not forward-compatible with CUDA 12.0. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 23.02 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 23.02 are based on TensorFlow 1.15.5 and 2.11.0.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.02	20.04	NVIDIA CUDA 12.0.1	2.11.0 1.15.5	TensorRT 8.5.3
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Using the old optimizers (available now under tf.keras.optimizers.legacy) resolves the errors.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 29. TensorFlow Release 23.01

The NVIDIA container image of TensorFlow, release 23.01, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `23.01-tf1-py3` and `23.01-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 12.0.1
- NVIDIA cuBLAS from CUDA 12.0.1
- cuTENSOR 1.6.2.3
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.16.5
- NVIDIA DALI® 1.21.0
- NVIDIA RAPIDS™ 22.12.0
- Horovod 0.24.3
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- TensorBoard
  - `23.01-tf1-py3` includes version 1.15.5

  - `23.01-tf2-py3` includes version 2.11.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.2.2
- TensorFlow-TensorRT (TF-TRT)
- Nsight Compute 2022.4.1.6
- Nsight Systems 2022.5.1
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 23.01 is based on CUDA 12.0.1, which requires NVIDIA Driver release 525 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), 515.65 (or later R515), or 525.85 (or later R525).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 12.0. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 23.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 23.01 are based on TensorFlow 1.15.5 and 2.11.0.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
23.01	20.04	NVIDIA CUDA 12.0.1	2.11.0 1.15.5	TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- In 23.01 containers, certain cuDNN cases that use runtime compilation via NVRTC, particularly on ARM SBSA systems, can fail with

`CUDNN_STATUS_RUNTIME_PREREQUISITE_MISSING`. A workaround for this situation is to export `LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11/lib64` . This will be fixed in an upcoming release.

- Note that if you wish to make modifications to the source and rebuild TensorFlow, starting from container release 22.10 (TensorFlow 2.10) you will need a C++ 17-compatible compiler.
- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Reverting to the old optimizers (available now under tf.keras.optimizers.legacy) resolves the errors. We also have an in-flight Horovod PR 3822 that fixes more cases.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 30. TensorFlow Release 22.12

The NVIDIA container image of TensorFlow, release 22.12, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

> **Note:** The `22.12-tf1-py3` and `22.12-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuTENSOR 1.6.1.5
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.15.5
- NVIDIA DALI® 1.20.0
- NVIDIA RAPIDS™ 22.10.01 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- TensorBoard

    - `22.12-tf1-py3` includes version 1.15.5
    - `22.12-tf2-py3` includes version 2.10.1
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.1
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.20.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.12 is based on CUDA 11.8.0, which requires NVIDIA Driver release 520 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 515.65 (or later R515).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.12 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.12 are based on TensorFlow 1.15.5 and 2.10.1.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.12	20.04	NVIDIA CUDA 11.8.0	2.10.1 1.15.5	TensorRT 8.5.1
22.11			2.10.0 1.15.5	
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- Some DLRM models may regress by 10-40%. We are currently investigating.

- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 31. TensorFlow Release 22.11

The NVIDIA container image of TensorFlow, release 22.11, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `22.11-tf1-py3` and `22.11-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuTENSOR 1.6.1.5
- NVIDIA cuDNN 8.7.0
- NVIDIA NCCL 2.15.5
- NVIDIA DALI® 1.18.0
- NVIDIA RAPIDS™ 22.10 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.2tp1
- TensorBoard

  - `22.11-tf1-py3` includes version 1.15.5
  - `22.11-tf2-py3` includes version 2.10.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.1
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.18.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.11 is based on CUDA 11.8.0, which requires NVIDIA Driver release 520 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 515.65 (or later R515).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.11 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.11 are based on TensorFlow 1.15.5 and 2.10.0.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.11	20.04	NVIDIA CUDA 11.8.0	2.10.0 1.15.5	TensorRT 8.5.1
22.10				TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- Certain models using RELU activation may exhibit extreme (and easily noticeable) performance regressions. We have root-cased this to a cuDNN issue and will release the fix in a future release.
- Certain models may crash with an out-of-memory error.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 32. TensorFlow Release 22.10.1

The NVIDIA container image of TensorFlow, release 22.10.1, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

> **Note:** The `22.10-tf1-py3` and `22.10-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuTENSOR 1.6.1.5
- NVIDIA cuDNN 8.6.0.163
- NVIDIA NCCL 2.15.5
- NVIDIA RAPIDS™ 22.08.01 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- SHARP 3.0.2
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.2tp1
- TensorBoard

  - `22.10-tf1-py3` includes version 1.15.5
  - `22.10-tf2-py3` includes version 2.10.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.0.12
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.18.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.10.1 is based on CUDA 11.8.0, which requires NVIDIA Driver release 520 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 515.65 (or later R515).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.10.1 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.10.1 are based on TensorFlow 1.15.5 and 2.10.0.
- Atex layer norm improvement: Starting with the 22.10 release, Atex layer normalization supports vectorization in load and store operations as well as shared memory usage for some cases which can boost the performance by ~20%.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.10	20.04	NVIDIA CUDA 11.8.0	2.10.0 1.15.5	TensorRT 8.5 EA
22.09			2.9.1 1.15.5	
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2	TensorRT 8.2.1.8

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.5	
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.2	
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- TF-TRT may exhibit substantial performance regressions in albert and electra FP16 models. We have root-caused this regression and are working towards a fix. Starting in 22.10 engine build times may also increase This is due to the addition of the new engines in TRT which lengthens the autotuning stage.
- TensorFlow Bert model may exhibit severe regressions in 22.10.1. We have root caused this issue and are working towards a fix.
- Some multi-GPU TF2 models (such as EfficientNet) may crash with a segmentation fault. As a potential workaround, try increasing the host memory limit from the default of 64GB, by setting the environment variable `TF_GPU_HOST_MEM_LIMIT_IN_MB=131072`, which is MBs.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 33. TensorFlow Release 22.10

The NVIDIA container image of TensorFlow, release 22.10, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

> **Note:** The `22.10-tf1-py3` and `22.10-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuTENSOR 1.6.1.5
- NVIDIA cuDNN 8.6.0.163
- NVIDIA NCCL 2.15.5
- NVIDIA RAPIDS™ 22.08.01 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.5a1
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.1a0
- TensorBoard
  - `22.10-tf1-py3` includes version 1.15.5

  - `22.10-tf2-py3` includes version 2.10.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.0.12
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.18.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.10 is based on CUDA 11.8.0, which requires NVIDIA Driver release 520 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 515.65 (or later R515).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.10 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.10 are based on TensorFlow 1.15.5 and 2.10.0.
- Atex layer norm improvement: Starting with the 22.10 release, Atex layer normalization supports vectorization in load and store operations as well as shared memory usage for some cases which can boost the performance by ~20%.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>22.10</td><td rowspan="13">20.04</td><td rowspan="2">NVIDIA CUDA 11.8.0</td><td>2.10.0
1.15.5</td><td rowspan="2">TensorRT 8.5 EA</td></tr>
<tr><td>22.09</td><td rowspan="4">2.9.1
1.15.5</td></tr>
<tr><td>22.08</td><td>NVIDIA CUDA 11.7.1</td><td>TensorRT 8.4.2.4</td></tr>
<tr><td>22.07</td><td rowspan="2">NVIDIA CUDA 11.7 Update 1 Preview</td><td>TensorRT 8.4.1</td></tr>
<tr><td>22.06</td><td rowspan="2">TensorRT 8.2.5</td></tr>
<tr><td>22.05</td><td>NVIDIA CUDA 11.7.0</td><td rowspan="3">2.8.0
1.15.5</td></tr>
<tr><td>22.04</td><td>NVIDIA CUDA 11.6.2</td><td>TensorRT 8.2.4.2</td></tr>
<tr><td>22.03</td><td>NVIDIA CUDA 11.6.1</td><td>TensorRT 8.2.3</td></tr>
<tr><td>22.02</td><td rowspan="2">NVIDIA CUDA 11.6.0</td><td rowspan="2">2.7.0
1.15.5</td><td>TensorRT 8.2.3</td></tr>
<tr><td>22.01</td><td>TensorRT 8.2.2</td></tr>
<tr><td>21.12</td><td rowspan="2">NVIDIA CUDA 11.5.0</td><td>2.6.2
1.15.5</td><td>TensorRT 8.2.1.8</td></tr>
<tr><td>21.11</td><td>2.6.0
1.15.5</td><td>TensorRT 8.0.3.4 for x64 Linux</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- TF-TRT may exhibit substantial performance, accuracy, and engine build time issues. We recommend staying with the 22.09 release. Issues are under active investigation.

- Some multi-GPU TF2 models (such as EfficientNet) may crash with a segmentation fault. As a potential workaround, try increasing the host memory limit from the default of 64GB, by setting the environment variable `TF_GPU_HOST_MEM_LIMIT_IN_MB=131072`, which is MBs.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

Chapter 34. TensorFlow Release 22.09

The NVIDIA container image of TensorFlow, release 22.09, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `22.09-tf1-py3` and `22.09-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuTENSOR 1.6.1.5
- NVIDIA cuDNN 8.6.0.163
- NVIDIA NCCL 2.15.1
- NVIDIA RAPIDS™ 22.08.01 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.1a0
- TensorBoard
  - `22.09-tf1-py3` includes version 1.15.5

- `22.09-tf2-py3` includes version 2.9.1
- rdma-core 36.0
- NVIDIA TensorRT™ 8.5.0.12
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.17.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.3.1.43
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.09 is based on CUDA 11.8.0, which requires NVIDIA Driver release 520 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), 510.47 (or later R510), or 515.65 (or later R515).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.8. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.09 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, and NVIDIA Hopper™ architecture families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.09 are based on TensorFlow 1.15.5 and 2.9.1.
- We introduced a new environment variable `TF_GRAPPLER_GRAPH_DEF_PATH` to output the Graphdef files before and after the TF grappler optimizations. For more information about the grappler optimizations, see the TensorFlow graph optimization with Grappler. In checking the optimized operation graph during the TF runtime, users can specify `TF_GRAPPLER_GRAPH_DEF_PATH=/path/to/graphdef`.
- We provided a visualization tool to convert (and compare) the given Graphdef files by `graphdef2pydot`, which was preinstalled in the 22.09 container. For more information about usage, use `graphdef2pydot -h`.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.09	20.04	NVIDIA CUDA 11.8.0	2.9.1 1.15.5	TensorRT 8.5.0.12
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- Some multi-GPU TF2 models (such as EfficientNet) may crash with a segmentation fault. As a potential workaround, try increasing the host memory limit from the default of 64GB, by setting the environment variable `TF_GPU_HOST_MEM_LIMIT_IN_MB=131072`, which is MBs.

- Additionally, we have introduced another feature to be able to switch to the channel-last layout (NHWC) for harnessing the power of Tensor Core math (see the Key Features and Enhancements section above). If you observed a performance regression in TF-TRT models, consider enabling the environment variable using `` `export TF_ENABLE_LAYOUT_NHWC=1` `` to check whether it helps regain the lost performance.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.

Chapter 35. TensorFlow Release 22.08

The NVIDIA container image of TensorFlow, release 22.08, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

> **Note:** The `22.08-tf1-py3` and `22.08-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.7.1
- NVIDIA cuBLAS 11.10.3.66
- cuTENSOR 1.6.0.2
- NVIDIA cuDNN 8.5.0.96
- NVIDIA NCCL 2.12.12 (built with CUDA 11.7)
- NVIDIA RAPIDS™ 22.06 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- TensorBoard
  - `22.08-tf1-py3` includes version 1.15.5

    - `22.08-tf2-py3` includes version 2.9.1
- rdma-core 36.0
- NVIDIA TensorRT™ 8.4.2.4
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.16.0
- Nsight Compute 2022.2.1.3
- Nsight Systems 2022.1.3.18
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.08 is based on CUDA 11.7.1, which requires NVIDIA Driver release 515 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), or 510.47 (or later R510).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.7. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.08 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.08 are based on TensorFlow 1.15.5 and 2.9.1.
- We introduced a new environment variable `TF_ENABLE_LAYOUT_NHWC` to enforce the NHWC layout at runtime. In some models with fp32 on NVIDIA Ampere Architecture GPUs, users may obtain better performance when specifying "`TF_ENABLE_LAYOUT_NHWC=1`", which can better utilize the TF32 tensor cores.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.08	20.04	NVIDIA CUDA 11.7.1	2.9.1 1.15.5	TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview		TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- We have implemented a new feature called the Async Allocator that may cause a set of isolated issues such as hangs or crashes in a multi-GPU setting or it may affect performance by a substantial margin. If you observe any of these issues when upgrading from 22.07, consider turning off this feature by unsetting the corresponding environment variable using `` `unset TF_GPU_ALLOCATOR` ``. We are actively working to address this issue in the next release.
- TF-TRT inference performance may also be affected by the above issue, so the above workaround also applied to TF-TRT models.
- Additionally, we have introduced another feature to be able to switch to the channel-last layout (NHWC) for harnessing the power of Tensor Core math (see the Key Features and Enhancements section above). If you observed a performance regression

in TF-TRT models, consider enabling the environment variable using `` `export TF_ENABLE_LAYOUT_NHWC=1` `` to check whether it helps regain the lost performance.

- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.

- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

- IO-dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known issue in XLA that could cause performance regressions of up to 55% as compared to the previous release; however, training with XLA is still faster than without XLA. This performance regression affects certain models such as EfficientNet with TF32 enabled. A potential workaround is disabling TF32 using the TensorFlow API. The root cause is under investigation and will be fixed in a future release.
- TF-TRT 22.07 may fail to build TensorRT engines for HF BERT or HF BART, which may manifest as large performance regressions. Please revert back to the previous version 22.06 if you see a TF-TRT warning stating that Myelin graph could not be created or see a substantial performance regression.
- Since the 22.08 release, the `GPU_TF_ALLOCATOR` is set to `CUDA_MALLOC_ASYNC`by default, which may cause severe regression in some particular configurations (for example, BERT training in `fp16` mode). When encountered such perf regressions, unset the environment variable: **unset TF_GPU_ALLOCATOR**

Chapter 36. TensorFlow Release 22.07

The NVIDIA container image of TensorFlow, release 22.07, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `22.07-tf1-py3` and `22.07-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.7 Update 1 Preview
- NVIDIA cuBLAS 11.10.3.66
- cuTENSOR 1.5.0.3
- NVIDIA cuDNN 8.4.1
- NVIDIA NCCL 2.12.12 (built with CUDA 11.7)
- NVIDIA RAPIDS™ 22.06 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- TensorBoard
  - `22.07-tf1-py3` includes version 1.15.5

  - `22.07-tf2-py3` includes version 2.9.1
- rdma-core 36.0
- NVIDIA TensorRT™ 8.4.1
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.15.0
- Nsight Compute 2022.2.1.3
- Nsight Systems 2022.1.3.3
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.07 is based on CUDA 11.7 Update 1 Preview, which requires NVIDIA Driver release 515 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), or 510.47 (or later R510).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.7. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.07 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.07 are based on TensorFlow 1.15.5 and 2.9.1.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.07	20.04	NVIDIA CUDA 11.7 Update 1 Preview	2.9.1 1.15.5	TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0	TensorRT 7.2.3.4

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.5	
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. NVIDIA's BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.
- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.

- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

- There is a known issue in XLA that could cause performance regressions of up to 55% as compared to the previous release; however, training with XLA is still faster than without XLA. This performance regression affects certain models such as EfficientNet with TF32 enabled. A potential workaround is disabling TF32 using the TensorFlow API. The root cause is under investigation and will be fixed in a future release.
- TF-TRT 22.07 may fail to build TensorRT engines for HF BERT or HF BART, which may manifest as large performance regressions. Please revert back to the previous version 22.06 if you see a TF-TRT warning stating that Myelin graph could not be created or see a substantial performance regression.

Chapter 37. TensorFlow Release 22.06

The NVIDIA container image of TensorFlow, release 22.06, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `22.06-tf1-py3` and `22.06-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.7 Update 1 Preview
- NVIDIA cuBLAS 11.10.3.66
- cuTENSOR 1.5.0.3
- NVIDIA cuDNN 8.4.1
- NVIDIA NCCL 2.12.12 (built with CUDA 11.7)
- NVIDIA RAPIDS™ 22.04 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.3
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- TensorBoard
  - `22.06-tf1-py3` includes version 1.15.5

  - `22.06-tf2-py3` includes version 2.9.1
- rdma-core 36.0
- NVIDIA TensorRT™ 8.2.5
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.14.0
- Nsight Compute 2022.2.0.13
- Nsight Systems 2022.1.3.3
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.06 is based on CUDA 11.7 Update 1 Preview, which requires NVIDIA Driver release 515 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), or 510.47 (or later R510).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.7. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.06 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.06 are based on TensorFlow 1.15.5 and 2.9.1.
- Added support for NHWC TF32 2D convolutions in XLA.
- TensorFlow 2.9 improves the functionality of `prefetch_to_device` to allow for concurrent kernel execution and data transfer. To make use of this feature, ensure that your dataset pipeline ends by applying the `prefetch_to_device` operation as follows:

```
dataset = dataset.batch(batch_size=1024)
…
dataset = dataset.apply(tf.data.experimental.prefetch_to_device('/gpu:0'))
```

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.06	20.04	NVIDIA CUDA 11.7 Update 1 Preview	2.9.1 1.15.5	TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	2.8.0 1.15.5	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. NVIDIA's BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.

- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known issue in XLA that could cause performance regressions of up to 55% as compared to the previous release; however, training with XLA is still faster than without XLA. This performance regression affects certain models such as EfficientNet with TF32 enabled. A potential workaround is disabling TF32 using the TensorFlow API. The root cause is under investigation and will be fixed in a future release.

Chapter 38. TensorFlow Release 22.05

The NVIDIA container image of TensorFlow, release 22.05, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

> **Note:** The `22.05-tf1-py3` and `22.05-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.7.0
- NVIDIA cuBLAS 11.10.1.25
- cuTensor 1.5.0.3
- NVIDIA cuDNN 8.4.0.27
- NVIDIA NCCL 2.12.10 (optimized for NVIDIA NVLink™)
- NVIDIA RAPIDS™ 22.04 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.2
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- TensorBoard
    - `22.05-tf1-py3` includes version 1.15.5

    - `22.05-tf2-py3` includes version 2.8.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.2.5.1
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.13.0
- Nsight Compute 2022.2.0.13
- Nsight Systems 2022.1.3.3
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.05 is based on CUDA 11.7, which requires NVIDIA Driver release 515 or later. However, if you are running on a data center GPU (for example, T4 or any other data center GPU), you can use NVIDIA driver release 450.51 (or later R450), 470.57 (or later R470), or 510.47 (or later R510).

The CUDA driver's compatibility package only supports particular drivers. Thus, users should upgrade from all R418, R440, and R460 drivers, which are not forward-compatible with CUDA 11.7. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 22.05 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.05 are based on TensorFlow 1.15.5 and 2.8.0.
- Fixed segfault in SparseToDense when `validate_indices` if false for both TF1 and TF2.
- Fixed XLA device indexing issue in TF2 that caused out-of-memory errors when using Horovod to distribute work to multiple GPUs
- Removed unneeded copies when saving resource variables. This lowers the effective memory footprint for models with large layers (for example, embedding layers in recommender models).
- Optimized depthwise convolution backprop filter kernel, providing speedups between 10 and 100x over previous implementation.

Announcements

- Starting with the 22.05 release, the TensorFlow 1 and 2 containers are available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.05-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for Slurm PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in Slurm. Users who depend on Slurm integration might need to configure Slurm for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.05	20.04	NVIDIA CUDA 11.7.0	2.8.0 1.15.5	TensorRT 8.2.5.1
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on NVIDIA Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. NVIDIA's BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUs for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet, which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.
- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the reweighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.
- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

  To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.
- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

Chapter 39. TensorFlow Release 22.04

The NVIDIA container image of TensorFlow, release 22.04, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** The `22.04-tf1-py3` and `22.04-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.6.2
- cuBLAS 11.9.3.115
- cuTensor
  - 1.5.0.1 (TensorFlow1)
  - 1.5.0.3 (TensorFlow2)
- NVIDIA cuDNN 8.4.0.27
- NVIDIA NCCL 2.12.10 (optimized for NVIDIA NVLink™)
- NVIDIA RAPIDS™ 22.02 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.24.2
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10

- TensorBoard
  - `22.04-tf1-py3` includes version 1.15.5
  - `22.04-tf2-py3` includes version 2.8.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.2.4.2
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.12.0
- Nsight Compute 2022.1.1.2
- Nsight Systems 2022.2.1.31-5fe97ab
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.04 is based on CUDA 11.6.2, which requires NVIDIA Driver release 510 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports specific drivers. For a complete list of supported drivers, see CUDA Application Compatibility. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 22.04 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.04 are based on Tensorflow 1.15.5 and 2.8.0.
- Container sizes were reduced by about 500MB (uncompressed) by removing redundant PTX code sections.
- Fixed the race condition in the cuDNN heuristics lookup that can sometimes lead to segmentation faults.
- TF2 added cuTENSOR support for the einsum single label case.
- Fixed pooling operations to support tensors with dimensions that exceed the 32-bit integer indexing.

Announcements

- NVIDIA Deep Learning Profiler (DLProf) v1.8, which was included in the 21.12 container, was the last release of DLProf.

  Starting with the 22.01 container, DLProf is no longer included, but it can still be manually installed by using a pip wheel on `nvidia-pyindex`.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.02-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for SLURM PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in SLURM. Users who depend on SLURM integration might need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.04	20.04	NVIDIA CUDA 11.6.2	2.8.0 1.15.5	TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1	2.8.0 1.15.5	TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
				TensorRT 8.0.2.2 for ARM SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

  The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

  This model script is available on GitHub and NGC.
- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. NVIDIA's BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUS for faster training times and maintains target accuracy.

This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow, TF-TRT will crash.

To prevent the conversion of an OP that causes a native segment fallback, use `export TF_TRT_OP_DENYLIST="ProblematicOp"`.

- A known issue affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```

Chapter 40. TensorFlow Release 22.03

The NVIDIA container image of TensorFlow, release 22.03, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is prebuilt and installed as a system Python module.

To achieve optimum TensorFlow performance for image-based training, the container includes a sample script that demonstrates the efficient training of convolutional neural networks (CNNs). The sample script might need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  Note: The `22.03-tf1-py3` and `22.03-tf2-py3` container images contain Python 3.8.

- NVIDIA CUDA® 11.6.1
- cuBLAS 11.8.1.74
- cuTensor
  - 1.5.0.1 (TensorFlow1)
  - 1.5.0.3 (TensorFlow2)
- NVIDIA cuDNN 8.3.3.40
- NVIDIA NCCL 2.12.9 (optimized for NVIDIA NVLink™)
- NVIDIA RAPIDS™ 22.02 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- Horovod 0.23.0
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- SHARP 2.5
- GDRCopy 2.3
- NVIDIA HPC-X 2.10

- TensorBoard
  - `22.03-tf1-py3` includes version 1.15.5
  - `22.03-tf2-py3` includes version 2.8.0
- rdma-core 36.0
- NVIDIA TensorRT™ 8.2.3
- TensorFlow-TensorRT (TF-TRT)
- NVIDIA DALI® 1.11.1
- Nsight Compute 2022.1.1.2
- Nsight Systems 2021.5.2.53
- JupyterLab 2.3.2 including Jupyter-TensorBoard
- XLA-Lite (TensorFlow2 only)

Driver Requirements

Release 22.03 is based on CUDA 11.6.1, which requires NVIDIA Driver release 510 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports specific drivers. For a complete list of supported drivers, see CUDA Application Compatibility. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 22.03 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal, NVIDIA Volta™, NVIDIA Turing™, and NVIDIA Ampere Architecture GPU families. For a list of GPUs to which this compute capability corresponds, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.03 are based on Tensorflow 1.15.5 and 2.8.0.
- The following vulnerabilities were patched in this release of TensorFlow 1.15.5: CVE-2020-10531, CVE-2021-41197, CVE-2021-41206, CVE-2021-41208, CVE-2022-21725, CVE-2022-21728, CVE-2022-21729, CVE-2022-21730, CVE-2022-21731, CVE-2022-21732, CVE-2022-21733, CVE-2022-21734, CVE-2022-21735, CVE-2022-21736, CVE-2022-21737, CVE-2022-21739, CVE-2022-21741, CVE-2022-23557, CVE-2022-23558, CVE-2022-23559, CVE-2022-23562, CVE-2022-23563, CVE-2022-23564, CVE-2022-23565, CVE-2022-23566, CVE-2022-23567, CVE-2022-23568, CVE-2022-23569, CVE-2022-23571, CVE-2022-23573, CVE-2022-23575, CVE-2022-23576,

CVE-2022-23577, CVE-2022-23578, CVE-2022-23579, CVE-2022-23580, CVE-2022-23581, CVE-2022-23582, CVE-2022-23583, CVE-2022-23584, CVE-2022-23585, CVE-2022-23586, CVE-2022-23588, CVE-2022-23589, and CVE-2022-23591.

- Fixed a bug in the XLA convolution autotuner which appeared in the `22.01-tf2` release that sometimes caused **Failed to determine best cudnn convolution algorithm: RESOURCE_EXHAUSTED** errors.
- TensorFlow 1.15 has been patched for compatibility with numpy 1.21.1, and the numpy version has been updated to that version.

  With older numpy releases, certain matrix operations resulted in NaN and Inf values on ARM SBSA.

Announcements

- NVIDIA Deep Learning Profiler (DLProf) v1.8, which was included in the 21.12 container, was the last release of DLProf.

  Starting with the 22.01 container, DLProf is no longer included, but it can still be manually installed by using a pip wheel on `nvidia-pyindex`.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform.

  For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.02-tf2-py3` Docker image on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for SLURM PMI2 has been removed from the 22.01 release.

  PMIX is supported by the container, but is not supported by default in SLURM. Users who depend on SLURM integration might need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.03	20.04	NVIDIA CUDA 11.6.1	2.8.0 1.15.5	TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	2.7.0	TensorRT 8.2.3
22.01				TensorRT 8.2.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.5	
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0	TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
			1.15.2	
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model: This model is a convolutional neural network for 2D image segmentation.

  This repository contains a U-Net implementation as described in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper, without any alteration.

  This model script is available on GitHub and NGC.
- SSD320 v1.2 model: This model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as *a method for detecting objects in images using a single deep neural network*.

  Our implementation is based on the existing model from the TensorFlow models repository.

  This model script is available on GitHub and NGC.
- Neural Collaborative Filtering (NCF) model: This model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions.

The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

This model script is available on GitHub and NGC.

- BERT model: Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

  This model is based on *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* paper. NVIDIA's BERT is an optimized version of Google's official implementation, which leverages mixed-precision arithmetic and Tensor Cores on V100 GPUS for faster training times and maintains target accuracy.

  This model script is available on GitHub and NGC.

- U-Net Industrial Defect Segmentation model: This model is adapted from the original version of the U-Net model, which is a convolutional auto-encoder for 2D image segmentation.

  U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the *U-Net: Convolutional Networks for Biomedical Image Segmentation* paper. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with high accuracy on the industrial anomaly dataset DAGM2007.

  This model script is available on GitHub and NGC.

- GNMT v2 model: This model is similar to the one discussed in the *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation* paper.

  The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

  This model script is available on GitHub and NGC.

- ResNet-50 v1.5 model: This model is a modified version of the original ResNet-50 v1 model.

  The difference between v1 and v1.5 is in the bottleneck blocks that require downsampling. For example, v1 has `stride = 2` in the first 1x1 convolution, and `v1.5 has stride = 2` in the 3x3 convolution. The following features were implemented in this model:

  - Data-parallel multi-GPU training with Horovod
  - Tensor Cores (mixed precision) training
  - Static loss scaling for Tensor Cores (mixed precision) training

  This model script is available on GitHub and NGC.

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, refer to the XLA Best Practices document, which offers information about how to diagnose symptoms and possibly address them.

- Experimental support for `cudaGraphs` in XLA has been dropped from the TensorFlow2 release.

  This support will be implemented again in a future release.
- TensorFlow 2.8.0 suffers from a known performance regression of up to 60%, which has been observed for some Wide & Deep recommender models that are running under XLA.

  This issue is under investigation and will be fixed in a future release. If you notice a slowdown, a temporary workaround to improve performance is to disable XLA.
- When you import the `tensorflow_addons` python module, the following spurious warning is printed:

  ```
  UserWarning: Tensorflow Addons supports using Python ops for all Tensorflow
   versions above or equal to 2.5.0 and strictly below 2.8.0 (nightly versions are
   not supported).
  The versions of TensorFlow you are currently using is 2.8.0 and is not
   supported...
  ```

  This warning can be safely ignored.
- For TensorFlow 1.15, TF-TRT inference throughput, when compared to the 21.06-tf1 release, might regress for certain models by up to 37%.

  This issue will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models.

  This issue will be fixed in a future release.
- There is a known performance regression that affects the UNet Medical 3D model training by up to 23%.

  This issue will be addressed in a future release.
- The TF-TRT native segment fallback has a known issue that causes a crash.

  This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build.

  Instead of falling back to native TensorFlow, TF-TRT crashes. You can use `export TF_TRT_OP_DENYLIST="ProblematicOp"` to prevent the conversion of an OP that causes a native segment fallback.
- There is a known issue that affects `aarch64 libgomp`, which might sometimes cause `cannot allocate memory in static TLS block` errors.

The workaround is to run the following command:

```
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
```

Chapter 41. TensorFlow Release 22.02

The NVIDIA container image of TensorFlow, release 22.02, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `22.02-tf1-py3` and `22.02-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.6.0
- cuBLAS 11.8.1.74
- cuTensor 1.4.0.6
- NVIDIA cuDNN 8.3.2
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- RAPIDS 21.10 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- rdma-core 36.0
- NVIDIA HPC-X 2.10
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- GDRCopy 2.3
- Nsight Compute 2022.1.0.12
- Nsight Systems 2021.5.2.53
- TensorRT 8.2.3

- TensorFlow-TensorRT (TF-TRT)
- SHARP 2.5
- DALI 1.10.0
- TensorBoard
  - `22.02-tf1-py3` includes version 1.15.5
  - `22.02-tf2-py3` includes version TensorBoard 2.7.0
- Horovod 0.23.0
- XLA-Lite (TF2 only)
- JupyterLab 2.3.2 including Jupyter-TensorBoard

Driver Requirements

Release 22.02 is based on NVIDIA CUDA 11.6.0, which requires NVIDIA Driver release 510 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 22.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.02 are based on Tensorflow 1.15.5 and 2.7.0.
- For TF2 added `CudnnMHA` Keras op to expose CUDNN's optimized multi-head attention implementation.
- Fixed segmentation fault when VLOG logging was enabled in TF1.
- Updated TF-TRT with latest upstream changes.
- Fixed bug in TF2 where CUDNN's fused batched norm grad kernels could be called when `training = false`.
- Extended autotuning over CUDNN fallback engines. This change may increase the execution time of the first few iterations, but can result in substantially better engines being chosen during later iterations.

Announcements

- DLProf v1.8, which was included in the 21.12 container, was the last release of DLProf. Starting with the 22.01 container, DLProf is no longer included. It can still be manually installed via a pip wheel on the nvidia-pyindex.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform. For example, pulling the Docker image `nvcr.io/nvidia/tensorflow:22.02-tf2-py3` on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Support for SLURM PMI2 has been removed from the 22.01 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
22.02	20.04	NVIDIA CUDA 11.6.0	2.7.0 1.15.5	TensorRT 8.2.3
22.01				TensorRT 8.2.2
21.12		NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- There is a known issue affecting aarch64 libgomp that may cause `cannot allocate memory in static TLS block` errors in some cases. A workaround is to `export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1`.

Chapter 42. TensorFlow Release 22.01

The NVIDIA container image of TensorFlow, release 22.01, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `22.01-tf1-py3` and `22.01-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.6.0
- cuBLAS 11.8.1.74
- NVIDIA cuDNN 8.3.2
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- RAPIDS 21.10 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- rdma-core 36.0
- NVIDIA HPC-X 2.10
- OpenMPI 4.1.2rc4+
- OpenUCX 1.12.0
- GDRCopy 2.3
- Nsight Compute 2021.3.0.13
- Nsight Systems 2021.5.2.53
- TensorRT 8.2.2
- TensorFlow-TensorRT (TF-TRT)

- SHARP 2.5
- DALI 1.9
- TensorBoard
  - `22.01-tf1-py3` includes version 1.15.5
  - `22.01-tf2-py3` includes version TensorBoard 2.7.0
- Horovod 0.23.0
- XLA-Lite (TF2 only)
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 22.01 is based on NVIDIA CUDA 11.6.0, which requires NVIDIA Driver release 510 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 22.01 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 22.01 are based on Tensorflow 1.15.5 and 2.7.0.
- Fixed circular dependency in monolithic builds to address github issue 21.
- TF-TRT respects enable_tensor_float_32_execution Python API.
- TF-TRT supports Structured Sparsity on NVIDIA Ampere architecture GPUs. This can be enabled by passing `enable_sparse_compute=True` to `TrtGraphConverterV2`.

Announcements

- DLProf v1.8, which was included in the 21.12 container, was the last release of DLProf. Starting with the 22.01 container, DLProf is no longer included. It can still be manually installed via a pip wheel on the nvidia-pyindex.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform. For example, pulling the Docker image `nvcr.io/`

`nvidia/tensorflow:22.01-tf2-py3` on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Support for SLURM PMI2 has been removed from the 22.01 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>22.01</td><td rowspan="11">20.04</td><td>NVIDIA CUDA 11.6.0</td><td>2.7.0
1.15.5</td><td>TensorRT 8.2.2</td></tr>
<tr><td>21.12</td><td rowspan="2">NVIDIA CUDA 11.5.0</td><td>2.6.2
1.15.5</td><td>TensorRT 8.2.1.8</td></tr>
<tr><td>21.11</td><td rowspan="3">2.6.0
1.15.5</td><td rowspan="2">TensorRT 8.0.3.4 for x64 Linux
TensorRT 8.0.2.2 for Arm SBSA Linux</td></tr>
<tr><td>21.10</td><td>NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2</td></tr>
<tr><td>21.09</td><td>NVIDIA CUDA 11.4.2</td><td>TensorRT 8.0.3</td></tr>
<tr><td>21.08</td><td>NVIDIA CUDA 11.4.1</td><td rowspan="2">2.5.0
1.15.5</td><td rowspan="2">TensorRT 8.0.1.6</td></tr>
<tr><td>21.07</td><td>NVIDIA CUDA 11.4.0</td></tr>
<tr><td>21.06</td><td>NVIDIA CUDA 11.3.1</td><td>2.5.0
1.15.5</td><td rowspan="3">TensorRT 7.2.3.4</td></tr>
<tr><td>21.05</td><td rowspan="2">NVIDIA CUDA 11.3.0</td><td rowspan="3">2.4.0
1.15.5</td></tr>
<tr><td>21.04</td></tr>
<tr><td>21.03</td><td>NVIDIA CUDA 11.2.1</td><td>TensorRT 7.2.2.3</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- Debugging with TF_CPP_MIN_VLOG_LEVEL=3 can result in a segmentation while auto-tuning convolution algorithms. This will be fixed in the 22.02 release.

Chapter 43. TensorFlow Release 21.12

The NVIDIA container image of TensorFlow, release 21.12, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  Note: Container image `21.12-tf1-py3` and `21.12-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.5.0
- cuBLAS 11.7.3.1
- NVIDIA cuDNN 8.3.1.22
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- RAPIDS 21.10 (Only these libraries are included: cudf, xgboost, rmm, cuml, and cugraph)
- rdma-core 36.0
- OpenMPI 4.1.1+
- OpenUCX 1.11.0rc1
- GDRCopy 2.3
- NVIDIA HPC-X 2.9
- Nsight Compute 2021.3.0.13
- Nsight Systems 2021.3.2.4
- TensorRT 8.2.1.8
- TensorFlow-TensorRT (TF-TRT)

- SHARP 2.5
- DALI 1.8
- DLProf 1.8.0
  - Included only in `21.12-tf1-py3`
- TensorBoard
  - `21.12-tf1-py3` includes version 1.15.5
  - `21.12-tf2-py3` includes version TensorBoard 2.6.0
- Horovod 0.22.0
- XLA-Lite (TF2 only)
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.12 is based on NVIDIA CUDA 11.5.0, which requires NVIDIA Driver release 495 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.12 are based on Tensorflow 1.15.5 and 2.6.2.
- The environment variable TF_DISABLE_REDUCED_PRECISION_REDUCTION=1 can now be set to disable intermediate reductions in lower precision than the requested math type.
- Patched the following CVEs in TensorFlow 1.15.5: CVE-2021-29571, CVE-2021-29592, CVE-2021-29601, CVE-2021-29608, CVE-2021-29609, CVE-2021-29613, CVE-2021-22876, CVE-2021-22897, CVE-2021-22898, CVE-2021-22901, CVE-2021-37636, CVE-2021-37640, CVE-2021-37642, CVE-2021-37644, CVE-2021-37646, CVE-2021-37653, CVE-2021-37660, CVE-2021-37661, CVE-2021-37668, CVE-2021-37669, CVE-2021-37670, CVE-2021-37672, CVE-2021-37673, CVE-2021-37674, CVE-2021-37675, CVE-2021-37684,

CVE-2021-37686, CVE-2021-37690, CVE-2021-37691, CVE-2021-41195, CVE-2021-41196, CVE-2021-41197, CVE-2021-41198, CVE-2021-41199, CVE-2021-41200, CVE-2021-41201, CVE-2021-41202, CVE-2021-41203, CVE-2021-41204, CVE-2021-41206, CVE-2021-41207, CVE-2021-41208, CVE-2021-41213, CVE-2021-41215, CVE-2021-41216, CVE-2021-41217, CVE-2021-41218, CVE-2021-41219, CVE-2021-41221, CVE-2021-41222, CVE-2021-41223, CVE-2021-41224, CVE-2021-41225, CVE-2021-41228, CVE-2021-22922, CVE-2021-22923, CVE-2021-22924, CVE-2021-22925, CVE-2021-22926.

Announcements

- DLProf v1.8, which is included in the 21.12 container, will be the last release of DLProf. Starting with the 22.01 container, DLProf will no longer be included. It can still be manually installed via a pip wheel on the nvidia-pyindex.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform. Pulling the Docker image `nvcr.io/nvidia/tensorflow:21.12-tf2-py3` on an Arm SBSA machine will automatically fetch the Arm-specific image.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). Instead they can be obtained from Github or the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, users may need to add some packages that were previously pre-installed.
- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.12	20.04	NVIDIA CUDA 11.5.0	2.6.2 1.15.5	TensorRT 8.2.1.8
21.11			2.6.0	TensorRT 8.0.3.4 for x64 Linux

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	1.15.5	TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12 19.11			1.15.0	TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100

GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using

`export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.

- The version of OpenUCX included with TensorFlow container image version 21.11 has known issues with RAPIDS UCX-Py. When using Dask with this container version, pass `protocol="tcp"` to `LocalCUDACluster()`, not `protocol="ucx"`, to work around these issues. Additionally, LocalCUDACluster UCX-specific configurations must remain unspecified; they are: `enable_tcp_over_ucx`, `enable_nvlink`, `enable_infiniband`, `enable_rdmacm` and `ucx_net_devices`.

Chapter 44. TensorFlow Release 21.11

The NVIDIA container image of TensorFlow, release 21.11, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.11-tf1-py3` and `21.11-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.5.0
- cuBLAS 11.7.3.1
- NVIDIA cuDNN 8.3.0.96
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- RAPIDS 21.08
- rdma-core 36.0
- OpenMPI 4.1.1+
- OpenUCX 1.11.0rc1
- GDRCopy 2.3
- NVIDIA HPC-X 2.9
- Nsight Compute 2021.3.0.13
- Nsight Systems 2021.3.2.4
- TensorRT 8.0.3.4 for x64 Linux
- TensorRT 8.0.2.2 for ARM SBSA Linux
- TensorFlow-TensorRT (TF-TRT)

- SHARP 2.5
- DALI 1.7
- DLProf 1.7.0
  - Included only in `21.11-tf1-py3`
- TensorBoard
  - `21.11-tf1-py3` includes version 1.15.5
  - `21.11-tf2-py3` includes version TensorBoard 2.6.0
- Horovod 0.22.0
- XLA-Lite (TF2 only)
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.11 is based on NVIDIA CUDA 11.5.0, which requires NVIDIA Driver release 495 or later. However, if you are running on a Data Center GPU (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 460.27 (or later R460), or 470.57 (or later R470). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.11 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.11 are based on Tensorflow 1.15.5 and 2.6.0.
- TensorFlow container image version 21.11 introduces RAPIDS libraries cuDF, cuML, cuGraph, RMM, and XGBoost.
- Patched CVE-2021-37663 in TF1.
- Disabled Pandas, Scikit-learn, and Dask integrations by default in TF1. This addresses a performance regression when these libs are installed, as they now are as dependencies of RAPIDS. To re-enable use of these libraries in TF1, users will need to export the environment variable `TF_ALLOW_IOLIBS=1`.

Announcements

- DLProf v1.8, which will be included in the 21.12 container, will be the last release of DLProf. Starting with the 22.01 container, DLProf will no longer be included. It can still be manually installed via a pip wheel on the nvidia-pyindex.
- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform. Pulling the Docker image `nvcr.io/nvidia/tensorflow:21.11-tf2-py3` on an Arm SBSA machine will automatically fetch the Arm-specific image.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). Instead they can be obtained from Github or the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, users may need to add some packages that were previously pre-installed.
- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.11	20.04	NVIDIA CUDA 11.5.0	2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.10		NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2		
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 includes a feature that works around this issue, but TF1 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- The version of OpenUCX included with TensorFlow container image version 21.11 has known issues with RAPIDS UCX-Py. When using Dask with this container version, pass `protocol="tcp"` to `LocalCUDACluster()`, not `protocol="ucx"`, to work around these issues. Additionally, LocalCUDACluster UCX-specific configurations must remain

unspecified; they are: `enable_tcp_over_ucx`, `enable_nvlink`, `enable_infiniband`, `enable_rdmacm` and `ucx_net_devices`.

Chapter 45. TensorFlow Release 21.10

The NVIDIA container image of TensorFlow, release 21.10, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.10-tf1-py3` and `21.10-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2
- NVIDIA cuDNN 8.2.4.15
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- rdma-core 36.0
- OpenMPI 4.1.1+
- OpenUCX 1.11.0rc1
- GDRCopy 2.3
- NVIDIA HPC-X 2.9
- Nsight Compute 2021.2.2.1
- Nsight Systems 2021.3.2.4
- TensorRT 8.0.3.4 for x64 Linux
- TensorRT 8.0.2.2 for ARM SBSA Linux
- SHARP 2.5
- DALI 1.6
- DLProf 1.6.0

    - Included only in `21.10-tf1-py3`
- TensorBoard
    - `21.10-tf1-py3` includes version 1.15.5
    - `21.10-tf2-py3` includes version TensorBoard 2.6.0
- Horovod 0.22.0
- XLA-Lite (TF2 only)
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.10 is based on NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.10 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.10 are based on Tensorflow 1.15.5 and 2.6.0.
- Improved handling of exp ops in XLA.
- Enabled pointwise row vectorization for small rows in XLA.
- Integrated latest TF-TRT features for dynamic shape support.
- Gemm+bias+relu cublasLt based epilogue fusion in XLA. This feature can be enabled by setting the environment variable TF_USE_CUBLASLT=1.

Announcements

- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the Arm SBSA platform. Pulling the Docker image `nvcr.io/nvidia/tensorflow:21.10-tf2-py3` on an Arm SBSA machine will automatically fetch the Arm-specific image.

- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). Instead they can be obtained from Github or the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, users may need to add some packages that were previously pre-installed.
- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).
- The `nvtx-plugins` utility package pre-installed in previous releases has been removed. Users depending on `nvtx-plugins` can install it using ``pip install nvtx-plugins``.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.10	20.04	NVIDIA CUDA 11.4.2 with cuBLAS 11.6.5.2	2.6.0 1.15.5	TensorRT 8.0.3.4 for x64 Linux TensorRT 8.0.2.2 for Arm SBSA Linux
21.09		NVIDIA CUDA 11.4.2	2.6.0 1.15.5	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- The OpenSeq2Seq toolkit has been removed from the TensorFlow 1.x container.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 includes a feature that works around this issue, but TF1 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.

Chapter 46. TensorFlow Release 21.09

The NVIDIA container image of TensorFlow, release 21.09, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  Note: Container image `21.09-tf1-py3` and `21.09-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.4.2
- cuBLAS 11.6.1.51
- NVIDIA cuDNN 8.2.4.15
- NVIDIA NCCL 2.11.4 (optimized for NVLink™)
- Horovod 0.22.0
- rdma-core 36.0
- OpenMPI 4.1.2a1
- OpenUCX 1.11.0rc1
- GDRCopy 2.3
- NVIDIA HPC-X 2.9
- Nsight Compute 2021.2.2.1
- Nsight Systems 2021.3.1.57
- TensorRT 8.0.3
- TensorBoard
  - `21.09-tf1-py3` includes version 1.15.0

- 21.09-tf2-py3 includes version TensorBoard 2.6.0
- DALI 1.5
- DLProf 1.5.0
  - Included only in 21.09-tf1-py3
- XLA-Lite
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.09 is based on NVIDIA CUDA 11.4.2, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.09 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.09 are based on Tensorflow 1.15.5 and 2.6.0.
- The following vulnerabilities have been patched in the TensorFlow 1.x container: CVE-2021-37678, CVE-2021-37679, CVE-2021-37659, CVE-2021-37676, CVE-2021-37667, CVE-2021-37650, CVE-2021-37671, CVE-2021-37665, CVE-2021-37664, CVE-2021-37655, CVE-2021-37641, CVE-2021-37662, CVE-2021-37656, CVE-2021-37658, CVE-2021-37643, CVE-2021-37648, CVE-2021-37647, CVE-2021-37635, CVE-2021-37638, CVE-2021-37657, CVE-2021-37639, CVE-2021-37666, CVE-2021-37652, CVE-2021-37654, and CVE-2021-37651.
- CublasLT integration in native TensorFlow and XLA; allows for more flexible matmul fusions in XLA.

Announcements

- The TensorCore example models are no longer provided in the core container (previously shipped in /workspace/nvidia-examples). Instead they can be obtained from Github or the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, users may need to add some packages that were previously pre-installed.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.09	20.04	NVIDIA CUDA 11.4.2	2.6.0 1.15.5	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is slurm-wlm-basic-plugins).

- The nvtx-plugins utility package pre-installed in previous releases has been removed. Users depending on nvtx-plugins can install it as `pip install nvtx-plugins`.
- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- The OpenSeq2Seq toolkit has been removed from the TensorFlow 1.x container.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 21.08 includes a feature that works around this issue, but TF1 21.08 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.

Chapter 47. TensorFlow Release 21.08

The NVIDIA container image of TensorFlow, release 21.08, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.08-tf1-py3` and `21.08-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.4.1
- cuBLAS 11.5.4
- NVIDIA cuDNN 8.2.2.26
- NVIDIA NCCL 2.10.3 (optimized for NVLink™)
- Horovod 0.22.0
- rdma-core 36.0
- OpenMPI 4.1.1+
- OpenUCX 1.11.0rc1
- GDRCopy 2.2
- NVIDIA HPC-X 2.9
- Nsight Systems 2021.2.4.12
- TensorRT 8.0.1.6
- TensorBoard
  - `21.08-tf1-py3` includes version 1.15.0
  - `21.08-tf2-py3` includes version TensorBoard 2.6.0

- OpenSeq2Seq at commit 8f040a49
  - Included only in `21.08-tf1-py3`
- DALI 1.4
- DLProf 1.4.0
  - Included only in `21.08-tf1-py3`
- XLA-Lite
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.08 is based on NVIDIA CUDA 11.4.1, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.08 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.08 are based on Tensorflow 1.15.5 and 2.5.0
- Experimental integration of the cuTENSOR library for einsum operations is included in the 21.08-tf2-py3 container. This should improve performance for many einsum operations. To enable, export TF_ENABLE_CUTENSOR_EINSUM=1.
- Added XLA feature to de-select compilation candidates based on shape inference. To enable this feature, use the environment variable TF_XLA_DO_NOT_COMPILE_POSSIBLE_DYNAMIC_OPS.
- Bug fixes for the `cudaMallocAsync` GPU memory allocator.
- MKL is enabled for better performance in CPU-only workloads. To enable, set OMP_NUM_THREADS to a value >= 1.

Announcements

- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). Instead they can be obtained from Github or the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, users may need to add some packages that were previously pre-installed.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.08	20.04	NVIDIA CUDA 11.4.1	2.5.0 1.15.5	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0		
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.03		NVIDIA CUDA 10.2.89	2.1.0	TensorRT 7.0.0
20.02			1.15.2	
20.01			2.0.0 1.15.0	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without Tensor Cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model

is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- The OpenSeq2Seq toolkit is deprecated and will be removed starting in the 21.09-tf1-py3 release. This only affects the TensorFlow 1.x release.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 21.08 includes a feature that works around this issue, but TF1 21.08 does not include that feature and

may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.

- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 48. TensorFlow Release 21.07

The NVIDIA container image of TensorFlow, release 21.07, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.07-tf1-py3` and `21.07-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.4.0
- cuBLAS 11.5.2.43
- NVIDIA cuDNN 8.2.2.26
- NVIDIA NCCL 2.10.3 (optimized for NVLink™)
- Horovod 0.22.0
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.1
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.18
- Nsight Systems 2021.2.4.12
- TensorRT 8.0.1.6
- TensorBoard
  - `21.07-tf1-py3` includes version 1.15.0

    - `21.07-tf2-py3` includes version TensorBoard 2.5.0
- OpenSeq2Seq at commit 8f040a49
    - Included only in `21.07-tf1-py3`
- DALI 1.3
- DLProf 1.3.0
    - Included only in `21.07-tf1-py3`
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.07-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.07 is based on NVIDIA CUDA 11.4.0, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Increased GPU memory reservation to avoid OOM errors in some cases.
- Integrated TRT 8 Support.
- Improved NVTX markers to include XLA cluster names.

- Fixed a deadlock in XLA by backporting upstream PR 50280 to TF1 and TF2.
- Fixed an issue so that CUDNN now respects the TF32 disable switch.
- TF2 implements support for embedding ops on GPU:
  - SparseFillEmptyRows[Grad]
  - fp16 embedding_lookup_sparse
  - fp16 SparseSegmentSumGrad
  - SparseSegmentSum/Mean
  - SparseSegmentSum/MeanGrad
  - hash value to string
- TF2 - Use CUDA occupancy calculator to improve the performance of BiasAdd.
- TensorFlow container images version 21.07 are based on Tensorflow 1.15.5 and 2.5.0

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.07	20.04	NVIDIA CUDA 11.4.0	2.5.0 1.15.5	TensorRT 8.0.1.6
21.06		NVIDIA CUDA 11.3.1	2.5.0 1.15.5	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	2.4.0 1.15.5	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data

for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 21.07 includes a feature that works around this issue, but TF1 21.07 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- The TF1 21.07 container includes Django 3.2.2, which has a known vulnerability that was discovered late in our QA process. See CVE-2021-35042 for details. This will be fixed in the next release. TF2 21.07 is not vulnerable to this issue.
- The 21.07 release includes libsystemd and libudev versions that have a known vulnerability that was discovered late in our QA process. See CVE-2021-33910 for details. This will be fixed in the next release.
- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 49. TensorFlow Release 21.06

The NVIDIA container image of TensorFlow, release 21.06, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.06-tf1-py3` and `21.06-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.3.1
- cuBLAS 11.5.1.109
- NVIDIA cuDNN 8.2.1
- NVIDIA NCCL 2.9.9 (optimized for NVLink™)
- Horovod 0.22.0
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.1
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.18
- Nsight Systems 2021.2.1.58
- TensorRT 7.2.3.4
- TensorBoard
  - `21.06-tf1-py3` includes version 1.15.0

    - `21.06-tf2-py3` includes version TensorBoard 2.5.0
- OpenSeq2Seq at commit 8f040a49
    - Included only in `21.06-tf1-py3`
- DALI 1.2
- DLProf 1.2.0
    - Included only in `21.06-tf1-py3`
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.06-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.06 is based on NVIDIA CUDA 11.3.1, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.05 are based on Tensorflow 1.15.5 and 2.5.0
- Fixed an issue that caused XLA to initialize TensorFlow on all visible GPUs leading to OOM errors in Horovod and other multi-process configurations.

- Fixed an issue in the `FakeQuantizeAndDequantize` op that would result in non-symmetric quantization when `max=-min`.
- Implemented GPU kernels for ops common in recommender model input pipelines: `SparseApplyFtrl`, `[Sparse]ApplyProximalAdagrad`, `SparseReshape`, and `SparseToDense`.
- Vectorized GPU Gather op to improve performance.
- Introduced the environment variable `TF_CPP_VLOG_FILENAME` to direct VLOG output to a file.
- Improved CUDNN kernel selection by switching to `CUDNN_HEUR_B` kernel selector.
- Updated `tensorflow-addons` to r0.13.
- Added support for `FussedBatchNormGrad` op to optimize side-inputs and activations.
- Patched recently announced vulnerabilities in TF 1.15.5: CVE-2021-29591, CVE-2021-29605, CVE-2021-29606, and CVE-2021-29614.
- Ubuntu 20.04 with May 2021 updates

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>21.06</td><td rowspan="6">20.04</td><td>NVIDIA CUDA 11.3.1</td><td>2.5.0
1.15.5</td><td rowspan="3">TensorRT 7.2.3.4</td></tr>
<tr><td>21.05</td><td rowspan="2">NVIDIA CUDA 11.3.0</td><td rowspan="4">2.4.0
1.15.5</td></tr>
<tr><td>21.04</td></tr>
<tr><td>21.03</td><td>NVIDIA CUDA 11.2.1</td><td>TensorRT 7.2.2.3</td></tr>
<tr><td>21.02</td><td>NVIDIA CUDA 11.2.0</td><td>TensorRT 7.2.2.3+cuda11.1.0.024</td></tr>
<tr><td>20.12</td><td>NVIDIA CUDA 11.1.1</td><td rowspan="3">2.3.1
1.15.4</td><td>TensorRT 7.2.2</td></tr>
<tr><td>20.11</td><td rowspan="4">18.04</td><td rowspan="2">NVIDIA CUDA 11.1.0</td><td rowspan="2">TensorRT 7.2.1</td></tr>
<tr><td>20.10</td></tr>
<tr><td>20.09</td><td rowspan="2">NVIDIA CUDA 11.0.3</td><td>2.3.0
1.15.3</td><td rowspan="2">TensorRT 7.1.3</td></tr>
<tr><td>20.08</td><td>2.2.0</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.07		NVIDIA CUDA 11.0.194	1.15.3	
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the

specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF1 and TF2 containers include a version of of Django with a known vulnerability that was discovered late in our QA process. See CVE-2021-31542 for details. This will be fixed in the next release.
- The TF1 container includes a version of Pillow with known vulnerabilities discovered late in our QA process. See CVE-2021-25287, CVE-2021-28676, CVE-2021-28677, and CVE-2021-25288 for details. This will be fixed in the next release.
- In certain cases, TensorFlow may claim too much memory on Pascal-based GPUs leading to failures due to OOM and potentially an application hang. This can be worked around by setting the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB` to 675. This will be fixed in the 21.07 release.
- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 50. TensorFlow Release 21.05

The NVIDIA container image of TensorFlow, release 21.05, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.05-tf1-py3` and `21.05-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.3.0
- cuBLAS 11.5.1.101
- NVIDIA cuDNN 8.2.0.51
- NVIDIA NCCL 2.9.8 (optimized for NVLink™)
- Horovod 0.21.3
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.0
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.18
- Nsight Systems 2021.1.3.14
- TensorRT 7.2.3.4
- TensorBoard
  - `21.05-tf1-py3` includes version 1.15.0+nv21.4

    - `21.05-tf2-py3` includes version TensorBoard 2.4.1
- OpenSeq2Seq at commit 8f040a49
    - Included only in `21.05-tf1-py3`
- DALI 1.0.0
- DLProf 1.1.0
    - Included only in `21.05-tf1-py3`
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.05-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.05 is based on NVIDIA CUDA 11.3.0, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.05 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.05 are based on Tensorflow 1.15.5 and 2.4.0

- The environment variable TF_CUDNN_ENGINE_MAX_LIMITS can limit the number of CUDNN algos that are attempted for each convolutional layer during autotuning. This can reduce model startup costs potentially at the cost of some training throughput.
- A deterministic implementation of sparse tensor dense matmul is now available.
- Ubuntu 20.04 with April 2021 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.05	20.04	NVIDIA CUDA 11.3.0	2.4.0 1.15.5	TensorRT 7.2.3.4
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
 TF_NUM_INTEROP_THREADS=1
      horovodrun -np 8 bash -c 'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK
 python
    ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- The DLProf TensorBoard plugin included with the 21.04 and 21.05 releases is an incorrect version with respect to the DLProf command line tool included in those releases. To correct this, use the following command:

```
$ pip install --index-urlhttps://developer.download.nvidia.com/compute/redist
 nvidia_tensorboard_plugin_dlprof==1.1.0
```

Chapter 51. TensorFlow Release 21.04

The NVIDIA container image of TensorFlow, release 21.04, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.04-tf1-py3` and `21.04-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.3.0
- cuBLAS 11.5.1.101
- NVIDIA cuDNN 8.2.0.41
- NVIDIA NCCL 2.9.6 (optimized for NVLink™)
- Horovod 0.21.3
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.10.0
- GDRCopy 2.2
- NVIDIA HPC-X 2.8.2rc3
- Nsight Compute 2021.1.0.18
- Nsight Systems 2021.1.3.14
- TensorRT 7.2.3.4
- TensorBoard
  - `21.04-tf1-py3` includes version 1.15.0+nv21.4

    - `21.04-tf2-py3` includes version TensorBoard 2.4.1
- OpenSeq2Seq at commit 8f040a49
    - Included only in `21.04-tf1-py3`
- DALI 1.0.0
- DLProf 1.1.0
    - Included only in `21.04-tf1-py3`
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.04-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 2.3.1 including Jupyter-TensorBoard

Driver Requirements

Release 21.04 is based on NVIDIA CUDA 11.3.0, which requires NVIDIA Driver release 465.19.01 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), or 460.27 (or later R460). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.04 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.04 are based on Tensorflow 1.15.5 and 2.4.0
- Ubuntu 20.04 with March 2021 updates

- Improved performance by caching the compilation result after LLVM IR creation and removing subsequent LLVM and PTXAS compilation phases.
- Added GPU-deterministic `tf.sparse.sparse_dense_matmul` support (for the `tf.float32` data type). When `TF_DETERMINISTIC_OPS` is set to "true" or "1" then `tf.sparse.sparse_dense_matmul` will operate deterministically in both the forward and backward direction.
- Integrated CUDNN v8 API for RNN and fused conv+bias+activation ops.
- Fixed an issue that caused OOM errors in some cases when using a batch size of 1.
- Improved XLA handling of dynamic ops to avoid frequent recompilation.
- Implemented XLA persistent cache.
- Implemented custom learning rate support in Horovod.

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.04	20.04	NVIDIA CUDA 11.3.0	2.4.0 1.15.5	TensorRT 7.2.3.4
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
 TF_NUM_INTEROP_THREADS=1
      horovodrun -np 8 bash -c 'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK
 python
    ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- There is a known CUDNN performance regression affecting certain batch sizes of VGG based models by up to 45%. This will be fixed in a later release.
- The DLProf TensorBoard plugin included with the 21.04 release is an incorrect version with respect to the DLProf command line tool included in those releases. To correct this, use the following command:

```
$ pip install --index-urlhttps://developer.download.nvidia.com/compute/redist
 nvidia_tensorboard_plugin_dlprof==1.1.0
```

Chapter 52. TensorFlow Release 21.03

The NVIDIA container image of TensorFlow, release 21.03, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.03-tf1-py3` and `21.03-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.
- NVIDIA cuDNN 8.1.0
- NVIDIA NCCL 2.8.4 (optimized for NVLink™)
- Horovod 0.21.3
- OpenMPI 4.0.5
- TensorBoard
  - `21.03-tf1-py3` includes version 1.15.0+nv21.3
  - `21.03-tf2-py3` includes version TensorBoard 2.4.1
- MLNX_OFED 5.1
- OpenSeq2Seq at commit 8f040a49
  - Included only in `21.03-tf1-py3`
- TensorRT 7.2.2.3
- DALI 0.31.0
- DLProf 1.0.0
  - Included only in `21.03-tf1-py3`

- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.03-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 21.03 is based on NVIDIA CUDA 11.2.1, which requires NVIDIA Driver release 460.32.03 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51(or later R450). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.03 are based on Tensorflow 1.15.5 and 2.4.0
- The latest version of NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.1026
- The latest version of NVIDIA cuDNN 8.1.1
- The latest version of Horovod 0.21.3
- The latest version of TensorBoard
    - `21.03-tf1-py3` includes version 1.15.0+nv21.3
    - `21.03-tf2-py3` includes version TensorBoard 2.4.1

- The latest version of TensorRT 7.2.2.3
- The latest version of DALI 0.31.0
- The latest version of DLProf 1.0.0
- Ubuntu 20.04 with February 2021 updates
- NVTX profiling annotation ranges more accurately report the execution of asynchronous operations. Note that when profiling NVTX ranges must now be explicitly enabled by setting the environment variable TF_ENABLE_NVTX_RANGES=1.
- The CUDNN backend API is now used for convolutional ops. This provides a significant performance benefit by reducing CPU overheads of convolutions.
- The fused Conv+Bias+Relu op regression in CUDNN has been fixed and this op has been re-enabled in both XLA and the TF grappler optimizers. This improves performance particularly for inference in convolutional models.
- Bugs relating to auto-graph in TensorFlow 1.15 with Python 3.8 were fixed.

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.03	20.04	NVIDIA CUDA 11.2.1	2.4.0 1.15.5	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0		TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
 TF_NUM_INTEROP_THREADS=1
      horovodrun -np 8 bash -c 'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK
 python
    ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- Training the UNET3D models with a batch size of 1 can result in OOM (Out-Of-Memory) in the TensorFlow 1 container. This is caused by the map_and_batch_fusion optimizer from using the tf.datasets. One workaround solution is to add:

```
if self._batch_size == 1:
  options = dataset.options()
  options.experimental_optimization.map_and_batch_fusion = False
  dataset = dataset.with_options(options)
```

Chapter 53. TensorFlow Release 21.02

The NVIDIA container image of TensorFlow, release 21.02, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `21.02-tf1-py3` and `21.02-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.2.0 including cuBLAS 11.3.1
- NVIDIA cuDNN 8.1.0
- NVIDIA NCCL 2.8.4 (optimized for NVLink™)
- Horovod 0.21.0
- OpenMPI 4.0.5
- TensorBoard
  - `21.02-tf1-py3` includes version 1.15.0+nv21.2
  - `21.02-tf2-py3` includes version TensorBoard 2.4.1
- MLNX_OFED 5.1
- OpenSeq2Seq at commit 8f040a49
  - Included only in `21.02-tf1-py3`
- TensorRT 7.2.2
- DALI 0.29
- DLProf 0.19.0
  - Included only in `21.02-tf1-py3`

- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- XLA-Lite
- Tensor Core optimized examples: (Included only in `21.02-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 21.02 is based on NVIDIA CUDA 11.2.0, which requires NVIDIA Driver release 460.27.04 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51(or later R450). The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades and NVIDIA CUDA and Drivers Support.

GPU Requirements

Release 21.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 21.02 are based on Tensorflow 1.15.5 and 2.4.0
- The latest version of NVIDIA CUDA 11.2.0 including cuBLAS 11.3.1
- The latest version of NVIDIA cuDNN 8.1.0
- The latest version of NVIDIA NCCL 2.8.4
- The latest version of Horovod 0.20.2
- The latest version of TensorBoard
    - `21.02-tf1-py3` includes version 1.15.0+nv21.2

  - `21.02-tf2-py3` includes version TensorBoard 2.4.1
- The latest version of TensorRT 7.2.2.3+cuda11.1.0.024
- The latest version of DALI 0.29
- The latest version of DLProf 0.19.0
- The latest version of Nsight Compute 2020.3.0.18
- The latest version of Nsight Systems 2020.4.3.7
- Ubuntu 20.04 with January 2021 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
21.02	20.04	NVIDIA CUDA 11.2.0	2.4.0 1.15.5	TensorRT 7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- A regression (only observed with NVIDIA Ampere GPU architecture) in CUDNN's fused Convolution+Bias+Activation implementation can cause performance regressions of up to 24% in some models such as UNet Medical. This will be fixed in a future CUDNN release.

- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8 which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- Training the UNET3D models with a batch size of 1 can result in OOM (Out-Of-Memory) in the TensorFlow 1 container.

Chapter 54. TensorFlow Release 21.01

The NVIDIA container image release for TensorFlow 21.01 has been canceled. The next release will be the 21.02 release which is expected to be released at the end of February.

Chapter 55. TensorFlow Release 20.12

The NVIDIA container image of TensorFlow, release 20.12, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 20.04

  > **Note:** Container image `20.12-tf1-py3` and `20.12-tf2-py3` contains Python 3.8

- NVIDIA CUDA 11.1.1 including cuBLAS 11.3.0
- NVIDIA cuDNN 8.0.5
- NVIDIA NCCL 2.8.3 (optimized for NVLink™)
- Horovod 0.20.2
- OpenMPI 4.0.5
- TensorBoard
  - `20.12-tf1-py3` includes version 1.15.0+nv20.11
  - `20.12-tf2-py3` includes version 2.3.0+nv20.11
- MLNX_OFED 5.1
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.12-tf1-py3`
- TensorRT 7.2.2
- DALI 0.28
- DLProf 0.18.0
  - Included only in `20.12-tf1-py3`

- Nsight Compute 2020.2.1.8
- Nsight Systems 2020.3.4.32
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.12-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.12 is based on NVIDIA CUDA 11.1.1, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.12 are based on Tensorflow 1.15.4 and 2.3.1
- The latest version of NVIDIA CUDA 11.1.1 including cuBLAS 11.3.0
- The latest version of NVIDIA cuDNN 8.0.5
- The latest version of NVIDIA NCCL 2.8.3
- The latest version of Horovod 0.20.2
- The latest version of TensorRT 7.2.2
- The latest version of DALI 0.28
- The latest version of DLProf 0.18.0
- The latest version of Nsight Compute 2020.2.1.8

- Ubuntu 20.04 with November 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.12	20.04	NVIDIA CUDA 11.1.1	2.3.1 1.15.4	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D

image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

> **Note:** If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- A regression (only observed with NVIDIA Ampere GPU architecture) in CUDNN's fused Convolution+Bias+Activation implementation can cause performance regressions of up to 24% in some models such as UNet Medical. This will be fixed in a future CUDNN release.
- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8 which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.

- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- Training the UNET3D models with a batch size of 1 can result in OOM (Out-Of-Memory) in the TensorFlow 1 container.

Chapter 56. TensorFlow Release 20.11

The NVIDIA container image of TensorFlow, release 20.11, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.11-tf1-py3` and `20.11-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.8.2 (optimized for NVLink™)
- Horovod 0.20.2
- OpenMPI 4.0.5
- TensorBoard
  - `20.11-tf1-py3` includes version 1.15.0+nv20.11
  - `20.11-tf2-py3` includes version 2.3.0+nv20.11
- MLNX_OFED 5.1
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.11-tf1-py3`
- TensorRT 7.2.1
- DALI 0.27
- DLProf 0.17.0
  - Included only in `20.11-tf1-py3`

- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.11-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.11 is based on NVIDIA CUDA 11.1.0, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.11 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.11 are based on Tensorflow 1.15.4 and 2.3.1
- The latest version of NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- The latest version of NVIDIA NCCL 2.8.2
- The latest version of Horovod 0.20.2
- The latest version of DALI 0.27
- The latest version of DLProf 0.17.0
- Ubuntu 18.04 with October 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.11	18.04	NVIDIA CUDA 11.1.0	2.3.1 1.15.4	TensorRT 7.2.1
20.10				
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- In certain cases running on Pascal GPUs may result in out-of-memory errors which may present as apparent job hangs. This can be worked around by exporting the following environment variable:

  ```
  TF_DEVICE_MIN_SYS_MEMORY_IN_MB=550
  ```
- A regression in CUDNN's fused Convolution+Bias+Activation implementation can cause performance regressions of up to 24% in some models such as UNet Medical. This will be fixed in a future CUDNN release.
- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4 which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.

- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 57. TensorFlow Release 20.10

The NVIDIA container image of TensorFlow, release 20.10, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.10-tf1-py3` and `20.10-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- Horovod 0.20.0
- OpenMPI 3.1.6
- TensorBoard
  - `20.10-tf1-py3` includes version 1.15.0+nv
  - `20.10-tf2-py3` includes version 2.3.2
- MLNX_OFED
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.10-tf1-py3`
- TensorRT 7.2.1
- DALI 0.26
- DLProf 0.16.0
  - Included only in `20.10-tf1-py3`

- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.10-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.10 is based on NVIDIA CUDA 11.1.0, which requires NVIDIA Driver release 455 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx, 440.30, or 450.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.10 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.10 are based on Tensorflow 1.15.4 and 2.3.1
- The latest version of NVIDIA CUDA 11.1.0 including cuBLAS 11.2.1
- The latest version of NVIDIA cuDNN 8.0.4
- The latest version of Horovod 0.20.0
- The latest version of TensorRT 7.2.1
- The latest version of Nsight Compute 2020.2.0.18
- The latest version of Nsight Systems 2020.3.4.32
- The latest version of DALI 0.26
- The latest version of DLProf 0.16.0

- Ubuntu 18.04 with September 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.10	18.04	NVIDIA CUDA 11.1.0	2.3.1 1.15.4	TensorRT 7.2.1
20.09		NVIDIA CUDA 11.0.3	2.3.0 1.15.3	TensorRT 7.1.3
20.08			2.2.0 1.15.3	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which

performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4 which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 58. TensorFlow Release 20.09

The NVIDIA container image of TensorFlow, release 20.09, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.09-tf1-py3` and `20.09-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- NVIDIA cuDNN 8.0.4
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- Horovod 0.19.5
- OpenMPI 3.1.6
- TensorBoard
  - `20.09-tf1-py3` includes version 1.15.0+nv
  - `20.09-tf2-py3` includes version 2.3.2
- MLNX_OFED
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.09-tf1-py3`
- TensorRT 7.1.3
- DALI 0.25.1
- DLProf 0.15.0
  - Included only in `20.09-tf1-py3`

- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.08-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.09 is based on NVIDIA CUDA 11.0.3, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.09 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.09 are based onTensorflow 1.15.3 and 2.3.2
- The latest version of NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- The latest version of NVIDIA cuDNN 8.0.4
- The latest version of TensorBoard 2.3.2
- The latest version of DALI 0.25.1
- The latest version of DLProf 0.15.0
- Ubuntu 18.04 with August 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>20.09</td><td rowspan="10">18.04</td><td rowspan="2">NVIDIA CUDA 11.0.3</td><td>2.3.0
1.15.3</td><td rowspan="3">TensorRT 7.1.3</td></tr>
<tr><td>20.08</td><td rowspan="2">2.2.0
1.15.3</td></tr>
<tr><td>20.07</td><td>NVIDIA CUDA 11.0.194</td></tr>
<tr><td>20.06</td><td>NVIDIA CUDA 11.0.167</td><td>2.2.0
1.15.2</td><td>TensorRT 7.1.2</td></tr>
<tr><td>20.03
20.02</td><td rowspan="3">NVIDIA CUDA 10.2.89</td><td>2.1.0
1.15.2</td><td rowspan="2">TensorRT 7.0.0</td></tr>
<tr><td>20.01</td><td rowspan="2">2.0.0
1.15.0</td></tr>
<tr><td>19.12
19.11</td><td rowspan="3">TensorRT 6.0.1</td></tr>
<tr><td>19.10</td><td rowspan="3">NVIDIA CUDA 10.1.243</td><td rowspan="3">1.14.0</td></tr>
<tr><td>19.09</td></tr>
<tr><td>19.08</td><td>TensorRT 5.1.5</td></tr>
</table>

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100 and Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 59. TensorFlow Release 20.08

The NVIDIA container image of TensorFlow, release 20.08, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  Note: Container image `20.08-tf1-py3` and `20.08-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- NVIDIA cuDNN 8.0.2
- NVIDIA NCCL 2.7.8 (optimized for NVLink™)
- Horovod 0.19.5
- OpenMPI 3.1.6
- TensorBoard
  - `20.08-tf1-py3` includes version 1.15.0+nv
  - `20.08-tf2-py3` includes version 2.2.1
- MLNX_OFED
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.08-tf1-py3`
- TensorRT 7.1.3
- DALI 0.24
- DLProf 0.14.0
  - Included only in `20.08-tf1-py3`

- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.08-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.08 is based on NVIDIA CUDA 11.0.3, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.08 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.08 are based onTensorflow 1.15.3 and 2.2.0
- The latest version of NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- The latest version of NVIDIA cuDNN 8.0.2
- The latest version of NVIDIA NCCL 2.7.8
- The latest version of DALI 0.24
- The latest version of DLProf 0.14.0
- The latest version of Nsight Compute 2020.1.2.4
- The latest version of Nsight Systems 2020.3.2.6
- The latest version of TensorRT 7.1.3

- The latest version of OpenSeq2Seq at commit 8f040a49
- The latest version of Horovod 0.19.5
- The latest version of JupyterLab 1.2.14 including Jupyter-TensorBoard
- Ubuntu 18.04 with July 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.08	18.04	NVIDIA CUDA 11.0.3	2.2.0 1.15.3	TensorRT 7.1.3
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The memory required to train MaskRCNN with a given batch size has increased from 20.07 to 20.08. As a result, the batch size may need to be decreased.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- There is a known performance regression of 15% compared to the 20.03 release when training the JoC V-Net Medical models with small batch size and fp32 data type on Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.

- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 60. TensorFlow Release 20.07

The NVIDIA container image of TensorFlow, release 20.07, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.07-tf1-py3` and `20.07-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.0.194 including cuBLAS 11.1.0
- NVIDIA cuDNN 8.0.1
- NVIDIA NCCL 2.7.6 (optimized for NVLink™)
- Horovod 0.19.5
- OpenMPI 3.1.6
- TensorBoard
  - `20.07-tf1-py3` includes version 1.15.0+nv
  - `20.07-tf2-py3` includes version 2.2.1
- MLNX_OFED
- OpenSeq2Seq at commit 8f040a49
  - Included only in `20.07-tf1-py3`
- TensorRT 7.1.3
- DALI 0.23
- DLProf 0.13.0
  - Included only in `20.07-tf1-py3`

- Nsight Compute 2020.1.1.8
- Nsight Systems 2020.3.2.6
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.07-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.14 including Jupyter-TensorBoard

Driver Requirements

Release 20.07 is based on NVIDIA CUDA 11.0.194, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container images version 20.07 are based onTensorflow 1.15.3 and 2.2.0
- The latest version of NVIDIA CUDA 11.0.194 including cuBLAS 11.1.0
- The latest version of NVIDIA cuDNN 8.0.1
- The latest version of NVIDIA NCCL 2.7.6
- The latest version of DALI 0.23
- The latest version of DLProf 0.13.0
- The latest version of Nsight Compute 2020.1.1.8
- The latest version of Nsight Systems 2020.3.2.6
- The latest version of TensorRT 7.1.3

- The latest version of OpenSeq2Seq at commit 8f040a49
- The latest version of Horovod 0.19.5
- The latest version of JupyterLab 1.2.14 including Jupyter-TensorBoard
- Integrated latest NVIDIA Deep Learning SDK to support NVIDIA A100 using CUDA 11 and cuDNN 8
- Improved NVTX annotations for XLA clusters for use with DLProf
- Improved XLA to avoid excessive recompilations
- Enhancements for Automatic Mixed Precision with einsum, 3D Convolutions, and list operations
- Improved 3D Convolutions to support NDHWC format
- Default TF32 support
- Ubuntu 18.04 with June 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.07	18.04	NVIDIA CUDA 11.0.194	2.2.0 1.15.3	TensorRT 7.1.3
20.06		NVIDIA CUDA 11.0.167	2.2.0 1.15.2	TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0	

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

> **Note:** TF1 TF-TRT is infrequently updated. In order to benefit from the latest performance improvements, optimizations and features such as implicit batch mode and dynamic shape support, we recommend using TF2.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.
```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```
- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.
```
TensorRTOptimizer is probably called on funcdef! This optimizer must *NOT* be called
 on function objects.
```
- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend using separate processes for different precisions until this issue gets resolved.

- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

  ```
  AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
  ```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

  ```
  import tensorflow.contrib.tensorrt as trt
  import nets.nets_factory
  ```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit()`, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the

existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100. This will be addressed in a future release.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.07 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- There is a known performance regression of 15% compared to the 20.03 release when training the JoC V-Net Medical models with small batch size and fp32 data type on Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

Chapter 61. TensorFlow Release 20.06

The NVIDIA container image of TensorFlow, release 20.06, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.06-tf1-py3` and `20.06-tf2-py3` contains Python 3.6

- NVIDIA CUDA 11.0.167 including cuBLAS 11.1.0
- NVIDIA cuDNN 8.0.1
- NVIDIA NCCL 2.7.5 (optimized for NVLink™)
- Horovod
  - `20.06-tf1-py3` includes version 0.19.1
  - `20.06-tf2-py3` includes version 0.19.2
- OpenMPI 3.1.6
- TensorBoard
  - `20.06-tf1-py3` includes version 1.15.2
  - `20.06-tf2-py3` includes version 2.1.1
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `20.06-tf1-py3`
- TensorRT 7.1.2
- DALI 0.22

- DLProf 0.12.0
    - Included only in `20.06-tf1-py3`
- Nsight Compute 2020.1.0.33
- Nsight Systems 2020.2.5.8
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.06-tf1-py3`)
    - U-Net Medical
    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- JupyterLab 1.2.2 including Jupyter-TensorBoard

Driver Requirements

Release 20.06 is based on NVIDIA CUDA 11.0.167, which requires NVIDIA Driver release 450 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the NVIDIA Pascal, Volta, Turing, and Ampere Architecture GPU families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- The latest version of NVIDIA CUDA 11.0.167 including cuBLAS 11.1.0
- The latest version of NVIDIA cuDNN 8.0.1
- The latest version of NVIDIA NCCL 2.7.5
- TensorFlow container image version 20.06 is based on TensorFlow 1.15.2 and TensorFlow 2.2.0.
- The latest version of DALI 0.21.2

- The latest version of DLProf 0.12.0
- The latest version of Nsight Compute 2020.1.0.33
- The latest version of Nsight Systems 2020.2.5.8
- The latest version of TensorRT 7.1.2
- The latest version of Horovod 0.19.1
- The latest version of JupyterLab 1.2.2 including Jupyter-TensorBoard
- Integrated latest NVIDIA Deep Learning SDK to support NVIDIA A100 using CUDA 11 and cuDNN 8
- Improved NVTX annotations for XLA clusters for use with DLProf
- Improved XLA to avoid excessive recompilations
- Enhancements for Automatic Mixed Precision with einsum, 3D Convolutions, and list operations
- Improved 3D Convolutions to support NDHWC format
- Default TF32 support
- Ubuntu 18.04 with May 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.06	18.04	NVIDIA CUDA 11.0.167	2.2.0 1.15.2	- TensorRT 7.1.2
20.03 20.02		NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.01			2.0.0	
19.12			1.15.0	TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

> **Note:** TF1 TF-TRT is infrequently updated. In order to benefit from the latest performance improvements, optimizations and features such as implicit batch mode and dynamic shape support, we recommend using TF2.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

```
TensorRTOptimizer is probably called on funcdef! This optimizer must *NOT* be called
 on function objects.
```

- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend using separate processes for different precisions until this issue gets resolved.

- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

```
AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

```
import tensorflow.contrib.tensorrt as trt
import nets.nets_factory
```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit()`, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the

existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100. This will be addressed in a future release.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.06 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- NCF depends on CuPy and will not work until a CuPy release supporting CUDA 11 is available.
- There is an issue that causes the error "`failed call to cuInit: CUDA_ERROR_UNKNOWN: unknown error`" in certain cases on NVIDIA A100/GA100 GPUs. The workaround is to start the container and then use the following:

```
$ dpkg -l '*nccl*'
$ dpkg -r libnccl-dev_2.7.5 libnccl2_2.5;# remove current nccl
 libs
$ apt-get update
$ apt-get install build-essential devscripts debhelper -y
$ git clone https://github.com/NVIDIA/nccl.git
$ cd nccl
$ git fetch
$ git checkout v2.7.6-1
$ make -j src.build pkg.debian.build
$ dpkg -i build/pkg/deb/libnccl-dev_2.7.6-1+cuda11.0_amd64.deb
$ dpkg -i build/pkg/deb/libnccl2_2.7.6-1+cuda11.0_amd64.deb
```

Chapter 62. TensorFlow Release 20.03

The NVIDIA container image of TensorFlow, release 20.03, is available on NGC.

Contents of the TensorFlow container

This container image includes the complete source of the NVIDIA version of TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.03-tf1-py3` and `20.03-tf2-py3` contains Python 3.6

- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.6.3 (optimized for NVLink™)
- Horovod 0.19.0
- OpenMPI 3.1.4
- TensorBoard
  - `20.03-tf1-py3` includes version 1.15.0+nv
  - `20.03-tf2-py3` includes version 2.1.1
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `20.03-tf1-py3`
- TensorRT 7.0.0
- DALI 0.19.0
- DLProf 0.10.0
  - Included only in `20.03-tf1-py3`

- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.03-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook
    - `20.03-tf1-py3` includes version 6.0.2
    - `20.03-tf2-py3` includes version 5.7.8
  - JupyterLab
    - `20.03-tf1-py3` includes version 1.2.2
    - `20.03-tf2-py3` includes version 1.0.2
  - JupyterLab Server
    - `20.03-tf1-py3` includes version 1.0.7
    - `20.03-tf2-py3` includes version 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 20.03 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 20.03 is based on TensorFlow 1.15.2 and TensorFlow 2.1.0.
- The lastest version of NVIDIA NCCL 2.6.3 (optimized for NVLink™)
- The latest version of DALI 0.19.0
- The latest version of DLProf 0.10.0
- When the environment variable TF_DETERMINISTIC_OPS is set to 'true' or '1', bilinear resizing will operate deterministically in both the forward and backward directions. In the TF 1 container image variant, the default way of accessing this functionality is via tf.image.resize_bilinear. In the TF 2 container image variant, the default way of accessing this functionality is via tf.image.resize with method=ResizeMethod.BILINEAR (which is the default method setting). This feature is also exposed through tf.keras.layers.UpSampling2D with interpolation='bilinear' (which is not the default interpolation setting). Enabling determinism may reduce performance. For more information, see NVIDIA's tensorflow-determinism repository on GitHub.
- Ubuntu 18.04 with February 2020 updates

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>20.03
20.02</td><td rowspan="6">18.04
16.04</td><td rowspan="3">NVIDIA CUDA 10.2.89</td><td>2.1.0
1.15.2</td><td rowspan="2">TensorRT 7.0.0</td></tr>
<tr><td>20.01</td><td rowspan="2">2.0.0
1.15.0</td></tr>
<tr><td>19.12
19.11</td><td rowspan="3">TensorRT 6.0.1</td></tr>
<tr><td>19.10</td><td rowspan="3">NVIDIA CUDA 10.1.243</td><td rowspan="3">1.14.0</td></tr>
<tr><td>19.09</td></tr>
<tr><td>19.08</td><td>TensorRT 5.1.5</td></tr>
</table>

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

> **Note:** TF1 TF-TRT is infrequently updated. In order to benefit from the latest performance improvements, optimizations and features such as implicit batch mode and dynamic shape support, we recommend using TF2.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

```
TensorRTOptimizer is probably called on funcdef! This optimizer must *NOT* be called
 on function objects.
```

- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend using separate processes for different precisions until this issue gets resolved.
- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

  ```
  AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
  ```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

  ```
  import tensorflow.contrib.tensorrt as trt
  import nets.nets_factory
  ```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit()`, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues since the 19.11 release for NCF inference with XLA and VGG16 training without XLA; these benchmarks have performance that is lower than expected.
- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx since the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 63. TensorFlow Release 20.02

The NVIDIA container image of TensorFlow, release 20.02, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `20.02-tf1-py3` and `20.02-tf2-py3` contains Python 3.6

- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- Horovod 0.19.0
- OpenMPI 3.1.4
- TensorBoard
  - `20.02-tf1-py3` includes version 1.15.0+nv
  - `20.02-tf2-py3` includes version 2.1.0
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `20.02-tf1-py3`
- TensorRT 7.0.0
- DALI 0.18.0 Beta
- DLProf 20.02
  - Included only in `20.02-tf1-py3`

- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.02-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook
    - `20.02-tf1-py3` includes version 6.0.2
    - `20.02-tf2-py3` includes version 5.7.8
  - JupyterLab
    - `20.02-tf1-py3` includes version 1.2.2
    - `20.02-tf2-py3` includes version 1.0.2
  - JupyterLab Server
    - `20.02-tf1-py3` includes version 1.0.6
    - `20.02-tf2-py3` includes version 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 20.02 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 20.02 is based on TensorFlow 1.15.2 and TensorFlow 2.1.0.
- Latest version of DLProf 20.02
- Latest version of DALI 0.18.0 Beta
- `20.02-tf2-py3` includes version 2.1.0
- Latest version of Nsight Systems 2020.1.1
- Latest version of Horovod 0.19.0
- Ubuntu 18.04 with January 2020 updates
- Improved AMP logging messages to include instructions for tweaking AMP lists.
- Added nvtx markers in TF 2.1 eager execution path for improved profiling with nvtx.

Announcements

- Python 2.7 is no longer supported in this TensorFlow container release.
- The TF_ENABLE_AUTO_MIXED_PRECISION environment variables are no longer supported in the tf2 container because it is not possible to automatically enable loss scaling in many cases in the tf 2.x API. Instead tf.train.experimental.enable_mixed_precision_graph_rewrite() should be used to enable AMP.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
20.02	18.04 16.04	NVIDIA CUDA 10.2.89	2.1.0 1.15.2	TensorRT 7.0.0
20.01			2.0.0 1.15.0	
19.12				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	TensorFlow	TensorRT
19.11				
19.10		NVIDIA CUDA 10.1.243	1.14.0	
19.09				
19.08				TensorRT 5.1.5

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

  ```
  OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
   does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
  ```
- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

  ```
  TensorRTOptimizer is probably called on funcdef! This optimizer must *NOT* be called
   on function objects.
  ```
- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend using separate processes for different precisions until this issue gets resolved.
- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

  ```
  AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
  ```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

  ```
  import tensorflow.contrib.tensorrt as trt
  import nets.nets_factory
  ```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit`(), automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For backward compatibility with previous container releases, AMP can also be enabled for `tf.train` optimizers by defining the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues since the 19.11 release for NCF inference with XLA and VGG16 training without XLA; these benchmarks have performance that is lower than expected.
- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx since the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 64. TensorFlow Release 20.01

The NVIDIA container image of TensorFlow, release 20.01, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image:
  >
  > - `20.01-tf1-py2` contains Python 2.7
  > - `20.01-tf1-py3` and `20.01-tf2-py3` contains Python 3.6

- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- Horovod 0.18.2
- OpenMPI 3.1.4
- TensorBoard
  - `20.01-tf1-py2` and `20.01-tf1-py3` include version 1.15.0+nv
  - `20.01-tf2-py3` includes version 2.0.1
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `20.01-tf1-py2` and `20.01-tf1-py3`
- TensorRT 7.0.0
- DALI 0.17.0 Beta

- DLProf 20.01
  - Included only in `20.01-tf1-py2` and `20.01-tf1-py3`
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- XLA-Lite
- Tensor Core optimized examples: (Included only in `20.01-tf1-py2` and `20.01-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook
    - `20.01-tf1-py2` and `20.01-tf1-py3` includes version 6.0.2
    - `20.01-tf2-py3` includes version 5.7.8
  - JupyterLab
    - `20.01-tf1-py2` and `20.01-tf1-py3` includes version 1.2.2
    - `20.01-tf2-py3` includes version 1.0.2
  - JupyterLab Server
    - `20.01-tf1-py2` and `20.01-tf1-py3` includes version 1.0.6
    - `20.01-tf2-py3` includes version 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 20.01 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.33.01. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 20.01 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 20.01 is based on TensorFlow 1.15.0 and TensorFlow 2.0.0.
- Latest version of TensorRT 7.0.0
- Latest version of DALI 0.17.0 Beta
- Latest version of DLProf 20.01
- XLA-Lite, reduced featured version of XLA focused on stable GPU performance
- Ubuntu 18.04 with December 2019 updates

Announcements

- We will stop support for Python 2.7 in the next TensorFlow container release.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA TensorFlow Container Versions

The following table shows what versions of Ubuntu, CUDA, TensorFlow, and TensorRT are supported in each of the NVIDIA containers for TensorFlow. For older container versions, refer to the Frameworks Support Matrix.

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>TensorFlow</th><th>TensorRT</th></tr>
<tr><td>20.01</td><td rowspan="5">18.04
16.04</td><td rowspan="2">NVIDIA CUDA 10.2.89</td><td rowspan="2">2.0.0
1.15.0</td><td>TensorRT 7.0.0</td></tr>
<tr><td>19.12
19.11</td><td rowspan="3">TensorRT 6.0.1</td></tr>
<tr><td>19.10</td><td rowspan="3">NVIDIA CUDA 10.1.243</td><td rowspan="3">1.14.0</td></tr>
<tr><td>19.09</td></tr>
<tr><td>19.08</td><td>TensorRT 5.1.5</td></tr>
</table>

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Fixed Issues

- Fixed a bug in Deconvolution (`conv2d_transpose`) that caused wrong outputs.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

```
TensorRTOptimizer is probably called on funcdef! This optimizer must *NOT* be called
 on function objects.
```

- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend using separate processes for different precisions until this issue gets resolved.
- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

```
AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

```
import tensorflow.contrib.tensorrt as trt
import nets.nets_factory
```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit`(), automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For backward compatibility with previous container releases, AMP can also be enabled for `tf.train` optimizers by defining the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues since the 19.11 release for NCF inference with XLA and VGG16 training without XLA; these benchmarks have performance that is lower than expected.
- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx since the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 65. TensorFlow Release 19.12

The NVIDIA container image of TensorFlow, release 19.12, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image:
  >
  > - `19.12-tf1-py2` contains Python 2.7
  > - `19.12-tf1-py3` and `19.12-tf2-py3` contains Python 3.6

- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- Horovod 0.18.2
- OpenMPI 3.1.4
- TensorBoard
  - `19.12-tf1-py2` and `19.12-tf1-py3` include version 1.15.0+nv
  - `19.12-tf2-py3` includes version 2.0.1
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `19.12-tf1-py2` and `19.12-tf1-py3`
- TensorRT 6.0.1
- DALI 0.16.0 Beta

- DLProf 19.12
  - Included only in `19.12-tf1-py2` and `19.12-tf1-py3`
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- Tensor Core optimized examples: (Included only in `19.12-tf1-py2` and `19.12-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook
    - `19.12-tf1-py2` and `19.12-tf1-py3` includes version 6.0.2
    - `19.12-tf2-py3` includes version 5.7.8
  - JupyterLab
    - `19.12-tf1-py2` and `19.12-tf1-py3` includes version 1.2.2
    - `19.12-tf2-py3` includes version 1.0.2
  - JupyterLab Server
    - `19.12-tf1-py2` and `19.12-tf1-py3` includes version 1.0.6
    - `19.12-tf2-py3` includes version 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.12 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.30. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410, 418.xx or 440.30. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.12 is based on TensorFlow 1.15.0 and TensorFlow 2.0.0.
- Latest version of DALI 0.16.0 Beta
- Latest version of DLProf 19.12
- Latest version of Horovod 0.18.2
- Latest version of Nsight Systems 2019.6.1
- Latest version of TensorBoard for `19.12-tf2-py3` includes version 2.0.2
- Jupyter Notebook, JupyterLab, and JupyterLab Server versions are now specific to which TensorFlow container version you choose to use.
- Added optimized `GenerateBoxPorposals` op for object detection models.
- Deterministic cuDNN convolutions, enabled via `TF_CUDNN_DETERMINISTIC` or `TF_DETERMINISTIC_OPS` are now available on a wider range of layer configurations. Prior to this version, some layer configurations would result in an exception with the message `No algorithm worked!`
- Ubuntu 18.04 with November 2019 updates

Announcements

- We will stop support for Python 2.7 in a future TensorFlow container release.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Per channel and QDQ op support for Quantization API in TensorFlow 1.15 container

Known Issues

- We have seen a performance regression in SSD Mobilenet V1 in 19.12 with both native TensorFlow and TF-TRT, mostly with batch size 8 but also 1 and 2, and with all types of GPUs. This could be due to a change in the SSD graph. We are still investigating this issue.

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this issue. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. This will be fixed in future releases of TensorRT.
- The following sentence that appears in the log of TensorRT 6.0 can be safely ignored. This will be removed in the future releases of TensorRT.

```
Calling isShapeTensor before the entire network is constructed may result in an
 inaccurate result.
```

- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend to use separate processes for different precisions until this issue gets resolved.
- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

```
AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

```
import tensorflow.contrib.tensorrt as trt
import nets.nets_factory
```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit`(), automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For backward compatibility with previous container releases, AMP can also be enabled for `tf.train` optimizers by defining the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues since the 19.11 release for NCF inference with XLA and VGG16 training without XLA; these benchmarks have performance that is lower than expected.
- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx since the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 66. TensorFlow Release 19.11

The NVIDIA container image of TensorFlow, release 19.11, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image:
  >
  > - `19.11-tf1-py2` contains Python 2.7
  > - `19.11-tf1-py3` and `19.11-tf2-py3` contains Python 3.6

- NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- NVIDIA cuDNN 7.6.5
- NVIDIA NCCL 2.5.6 (optimized for NVLink™)
- Horovod 0.18.1
- OpenMPI 3.1.4
- TensorBoard
  - `19.11-tf1-py2` and `19.11-tf1-py3` include version 1.15.0+nv
  - `19.11-tf2-py3` includes version 2.0.1
- MLNX_OFED
- OpenSeq2Seq at commit a81babd
  - Included only in `19.11-tf1-py2` and `19.11-tf1-py3`
- TensorRT 6.0.1
- DALI 0.15.0 Beta

- DLProf 19.11
  - Included only in `19.11-tf1-py2` and `19.11-tf1-py3`
- Nsight Compute 2019.5.0
- Nsight Systems 2019.5.2
- Tensor Core optimized examples: (Included only in `19.11-tf1-py2` and `19.11-tf1-py3`)
  - U-Net Medical
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.4
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.1
  - JupyterLab 1.0.2
  - JupyterLab Server 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.11 is based on NVIDIA CUDA 10.2.89, which requires NVIDIA Driver release 440.30. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+, 410 or 418.xx. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.11 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.11 is based on TensorFlow 1.15.0 and TensorFlow 2.0.0.
- Added a TensorFlow 2.x container.
- Latest version of NVIDIA CUDA 10.2.89 including cuBLAS 10.2.2.89
- Latest version of NVIDIA cuDNN 7.6.5
- Latest version of NVIDIA NCCL 2.5.6
- Latest version of TensorBoard. We now provide:
    - `19.11-tf1-py2` and `19.11-tf1-py3` include version 1.15.0+nv
    - `19.11-tf2-py3` includes version 2.0.1
- Latest version of OpenSeq2Seq at commit a81babd
    - Included only in `19.11-tf1-py2` and `19.11-tf1-py3`
- Latest version of Nsight Compute 2019.5.0
- Latest version of Nsight Systems 2019.5.2
- Latest version of DLProf 19.11
    - Included only in `19.11-tf1-py2` and `19.11-tf1-py3`
- Latest versions of Jupyter Client 5.3.4 and Jupyter Core 4.6.1
- Support added for the fast cuDNN CTC loss function via `nn.ctc_loss` (for `19.11-tf2-py3`) or `nn.ctc_loss_v2` (for `19.11-tf1-py3`). To enable it, define the `export TF_CUDNN_CTC_LOSS=1`environment variable.
- Ubuntu 18.04 with October 2019 updates

Announcements

- We will stop support for Python 2.7 in a future TensorFlow container release.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- New converters were added. Refer to the Supported Operators section in the *Accelerating Inference In TensorFlow With TensorRT User Guide* for the list of new converters.
- TensorFlow 2.0:
    - A new API is introduced for TF-TRT in TensorFlow 2.0. This new API can only be used in TensorFlow 2.0. Refer to the User Guide for more information about the new API.

- Introduced a new API method (`converter.build()`) for optimizing TensorRT engines during graph optimization. Previously, the optimization during preprocessing (before deployment) was possible by using `is_dynamic_op=False`.
- `converter.convert()` no longer returns a `tf.function`. Now, the function must be accessed from the saved model. This encapsulates the function in the converter for better safety.
- The `converter.calibrate()` method has been removed. To trigger calibration, a `calibration_input_fn` should be provided to `converter.convert()`.

Deprecated Features

- The old API of TF-TRT is deprecated. It still works in TensorFlow 1.14 and 1.15, however, it is removed in TensorFlow 2.0. The old API is a Python function named `create_inference_graph` which is now replaced by the Python class `TrtGraphConverter` in TensorFlow 1.x and `TrtGraphConverterV2` in TensorFlow 2.0 with a number of methods. Refer to TF-TRT User Guide for more information about the API and how to use it.

Known Issues

- We have observed a regression in the performance of certain TF-TRT benchmarks in TensorFlow 1.15 including image classification models with precision INT8. We are still investigating this. Since 19.11 comes with a new version of TensorFlow (1.15), which includes a lot of changes in the TensorFlow backend, it's very possible that the regression is caused by a change in the TensorFlow backend.
- CUDA 10.2 and NCCL 2.5.x libraries require slightly more device memory than previous releases. As a result, some models that ran previously may exhaust device memory.
- TensorRT INT8 calibration algorithm (see the TF-TRT User Guide for more information about how to use INT8) is very slow for certain models such as NASNet and Inception. We are working on optimizing the calibration algorithm in TensorRT.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. We are investigating the issue.
- The following sentence that appears in the log of TensorRT 6.0 can be safely ignored. This will be removed in the future releases of TensorRT.

```
Calling isShapeTensor before the entire network is constructed may result in an
 inaccurate result.
```

- The following warning is issued when the method `build()` from the API is not called. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- The following warning is issued because internally TensorFlow calls the TensorRT optimizer for certain objects unnecessarily. This warning can be ignored.

```
OP_REQUIRES failed at trt_engine_resource_ops.cc:183 : Not found: Container TF-TRT
 does not exist. (Could not find resource: TF-TRT/TRTEngineOp_...
```

- We have seen failures when using INT8 calibration (post-training) within the same process that does FP32/FP16 conversion. We recommend to use separate processes for different precisions until this issue gets resolved.
- We have seen failures when calling the TensorRT optimizer on models that are already optimized by TensorRT. This issue will be fixed in a future release.
- In case you import nets from models/slim, you might see the following error:

  ```
  AttributeError: module 'tensorflow_core.contrib' has no attribute 'tensorrt'
  ```

  Changing the order of imports can fix the issue. Therefore, import TensorRT before importing nets as follows:

  ```
  import tensorflow.contrib.tensorrt as trt
  import nets.nets_factory
  ```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit()`, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For backward compatibility with previous container releases, AMP can also be enabled for `tf.train` optimizers by defining the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is

concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues in the 19.11 release for NCF inference with XLA and VGG16 training without XLA; these benchmarks have performance that is lower than expected.
- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- Automatic Mixed Precision (AMP) does not support the Keras `LearningRateScheduler` in the 19.08 release. A fix will be included in a future release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx in the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 67. TensorFlow Release 19.10

The NVIDIA container image of TensorFlow, release 19.10, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  Note: Container image `19.10-py2` contains Python 2.7; `19.10-py3` contains Python 3.6.

- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.4
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- Horovod 0.18.1
- OpenMPI 3.1.4
- TensorBoard 1.14.0+nv
- MLNX_OFED
- OpenSeq2Seq at commit 2e0b1d8
- TensorRT 6.0.1
- DALI 0.14.0 Beta
- DLProf 19.10
- Nsight Compute 2019.4.0
- Nsight Systems 2019.5.1
- Tensor Core optimized example:
  - U-Net Medical

    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.3
    - Jupyter Core 4.5.0
    - Jupyter Notebook 6.0.1
    - JupyterLab 1.0.2
    - JupyterLab Server 1.0.0
    - Jupyter-TensorBoard

Driver Requirements

Release 19.10 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+ or 410. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.10 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.10 is based on TensorFlow 1.14.0.
- Latest version of NVIDIA cuDNN 7.6.4
- Latest version of Horovod 0.18.1
- Latest version of DALI 0.14.0 Beta
- Latest version of DLProf 19.10
- Latest versions of Nsight Systems 2019.5.1
- Latest versions of Jupyter Client 5.3.3

- Dilated convolutions will now be evaluated using cuDNN by default.
- Automatic Mixed Precision will correctly handle `TensorList` ops.
- Automatic Mixed Precision can now evaluate softmax and activation functions in FP16.
- Ubuntu 18.04 with September 2019 updates

Announcements

We will stop support for Python 2.7 in a future TensorFlow container release.

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Deprecated Features

- The old API of TF-TRT is deprecated. It still works in TensorFlow 1.14 and 1.15, however, it is removed in TensorFlow 2.0. The old API is a Python function named `create_inference_graph` which is now replaced by the Python class `TrtGraphConverter` with a number of methods. Refer to TF-TRT User Guide for more information about the API and how to use it.

Known Issues

- TensorRT INT8 calibration algorithm (see the TF-TRT User Guide for more information about how to use INT8) is very slow for certain models such as NASNet and Inception. We are working on optimizing the calibration algorithm in TensorRT.
- The pip package of TensorFlow 1.14 released by Google is missing TensorRT. This will be fixed in the next release of TensorFlow by Google. In the meantime, you can use the more recent versions of TensorFlow pip packages released by Google (1.15 and 2.0) or the NVIDIA container for TensorFlow.
- The accuracy of Faster RCNN with the backbone ResNet-50 using TensorRT6.0 INT8 calibration is lower than expected. We are investigating the issue.
- The following sentence that appears in the log of TensorRT 6.0 can be safely ignored. This will be removed in the future releases of TensorRT.

  ```
  Calling isShapeTensor before the entire network is constructed may result in an
   inaccurate result.
  ```

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using an optimizer from `tf.train` or `tf.keras.optimizers` for both `compute_gradients()` and `apply_gradients()` operations (for example, by calling `optimizer.minimize()` or `model.fit()`, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`.

For more information on this function, see the TensorFlow documentation here.

For backward compatibility with previous container releases, AMP can also be enabled for `tf.train` optimizers by defining the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

The tensor core examples provided in GitHub focus on achieving the best performance and convergence by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-

art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are known issues regarding TF-TRT INT8 accuracy issues. See the *Accelerating Inference In TensorFlow With TensorRT (TF-TRT)* section above for more information.
- There is a known performance regression in TensorFlow 1.14.0 affecting a variety of models. Affected models include GNMT, SSD, and NCF. Performance regressions can be as high as 20% compared to TensorFlow 1.13.1 in the 19.06 release.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.

- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- Automatic Mixed Precision (AMP) does not support the Keras `LearningRateScheduler` in the 19.08 release. A fix will be included in a future release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx in the 19.10 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 68. TensorFlow Release 19.09

The NVIDIA container image of TensorFlow, release 19.09, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  Note: Container image `19.09-py2` contains Python 2.7; `19.09-py3` contains Python 3.6.

- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.3
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- Horovod 0.18.0
- OpenMPI 3.1.4
- TensorBoard 1.14.0+nv
- MLNX_OFED
- OpenSeq2Seq at commit 2e0b1d8
- TensorRT 6.0.1
- DALI 0.13.0 Beta
- DLProf 19.09
- Nsight Compute 2019.4.0
- Nsight Systems 2019.4.2
- Tensor Core optimized example:
  - U-Net Medical

- SSD320 v1.2
- Neural Collaborative Filtering (NCF)
- BERT
- U-Net Industrial Defect Segmentation
- GNMT v2
- ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0
  - Jupyter Notebook 6.0.1
  - JupyterLab 1.0.2
  - JupyterLab Server 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.09 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 396, 384.111+ or 410. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.09 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.09 is based on TensorFlow 1.14.0.
- Latest version of NVIDIA cuDNN 7.6.3
- Latest version of Horovod 0.18.0
- Latest version of TensorRT 6.0.1
- Latest version of DALI 0.13.0 Beta
- Latest version of DLProf 19.09
- Latest versions of Nsight Compute 2019.4.0 and Nsight Systems 2019.4.2

- Latest version of Jupyter Notebook 6.0.1
- Ubuntu 18.04 with August 2019 updates

Announcements

We will stop support for Python 2.7 in a future TensorFlow container release.

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Deprecated Features

- The old API of TF-TRT is deprecated. It still works in TensorFlow 1.14, however, it may be removed in TensorFlow 2.0. The old API is a Python function named `create_inference_graph` which is not replaced by the Python class `TrtGraphConverter` with a number of methods. Refer to TF-TRT User Guide for more information about the API and how to use it.

Known Issues

- Precision mode in the TF-TRT API is a string with one of the following values: `FP32`, `FP16` or `INT8`. In TensorFlow 1.13, these strings were supported in lowercase, however, in TensorFlow 1.14 only uppercase is supported.
- INT8 calibration (see the TF-TRT User Guide for more information about how to use INT8) is a very slow process that can take 1 hour depending on the model. We are working on optimizing this algorithm in TensorRT.
- The pip package of TensorFlow 1.14 released by Google is missing TensorRT. This will be fixed in the next release of TensorFlow by Google. In the meantime, you can use the NVIDIA container for TensorFlow.

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.train.Optimizer` or `tf.keras.optimizers.Optimizer` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`. For backward compatibility with AMP in previous containers, AMP can also be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) by setting the following flag in the `tf.session` config:

```
config.graph_options.rewrite_options.auto_mixed_precision=1
```

Or equivalently for backward compatibility with AMP in previous NGC containers, by setting the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100

GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is a known performance regression in TensorFlow 1.14.0 affecting a variety of models. Affected models include GNMT, SSD, and UNet. Performance regressions can be as high as 20% compared to TensorFlow 1.13.1 in the 19.06 release.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- In previous containers, `libtensorflow_framework.so` was available in the `/usr/local/lib/tensorflow` directory. This was redundant with the libs installed

with the TensorFlow pip package. To find the TensorFlow `lib` directory, use `tf.sysconfig.get_lib()`.

- Automatic Mixed Precision (AMP) does not support the Keras `LearningRateScheduler` in the 19.08 release. A fix will be included in a future release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- There is a known issue when your NVIDIA driver release is older than 418.xx in the 19.09 release, the Nsight Systems profiling tool (for example, the `nsys`) might cause `CUDA runtime API error`. A fix will be included in a future release.

Chapter 69. TensorFlow Release 19.08

The NVIDIA container image of TensorFlow, release 19.08, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `19.08-py2` contains Python 2.7; `19.08-py3` contains Python 3.6.

- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.2
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- Horovod 0.16.2
- OpenMPI 3.1.4
- TensorBoard 1.14.0+nv
- MLNX_OFED +4.0
- OpenSeq2Seq at commit 2e0b1d8
- TensorRT 5.1.5
- DALI 0.12.0 Beta
- DLProf 19.08
- Nsight Compute 10.1.168
- Nsight Systems 2019.3.7.9
- Tensor Core optimized example:
  - U-Net Medical

  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)
  - BERT
  - U-Net Industrial Defect Segmentation
  - GNMT v2
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0
  - Jupyter Notebook 6.0.0
  - JupyterLab 1.0.2
  - JupyterLab Server 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.08 is based on NVIDIA CUDA 10.1.243, which requires NVIDIA Driver release 418.87. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.08 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.08 is based on TensorFlow 1.14.0.
- Latest version of NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- Latest version of NVIDIA cuDNN 7.6.2
- Latest version of NVIDIA NCCL 2.4.8
- Latest version of OpenMPI 3.1.4
- Latest commit of OpenSeq2Seq
- Latest version of Nsight Systems 2019.3.7.9

- Latest version of DALI 0.12.0 Beta
- Latest version of MLNX_OFED +4.0
- Latest version of DLProf 19.08
- Latest version of Jupyter Notebook 6.0.0
- Ubuntu 18.04 with July 2019 updates

Announcements

We will stop support for Python 2.7 in a future TensorFlow container release.

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Migrated TensorRT conversion sources from the `contrib` directory to the `compiler` directory in preparation for TensorFlow 2.0. The Python code can be found at `//tensorflow/python/compiler/tensorrt`.
- Added a user friendly `TrtGraphConverter` API for TensorRT conversion.
- Expanded support for TensorFlow operators in TensorRT conversion (for example, `Gather`, `Slice`, `Pack`, `Unpack`, `ArgMin`, `ArgMax`, `DepthSpaceShuffle`). Refer to the TF-TRT User Guide for a complete list of supported operators.
- Support added for TensorFlow operator `CombinedNonMaxSuppression` in TensorRT conversion which significantly accelerates SSD object detection models.
- Integrated TensorRT 5.1.5 into TensorFlow. See the TensorRT 5.1.5 Release Notes for a full list of new features.

Deprecated Features

- The old API of TF-TRT is deprecated. It still works in TensorFlow 1.14, however, it may be removed in TensorFlow 2.0. The old API is a Python function named `create_inference_graph` which is not replaced by the Python class `TrtGraphConverter` with a number of methods. Refer to TF-TRT User Guide for more information about the API and how to use it.

Known Issues

- Precision mode in the TF-TRT API is a string with one of the following values: `FP32`, `FP16` or `INT8`. In TensorFlow 1.13, these strings were supported in lowercase, however, in TensorFlow 1.14 only uppercase is supported.
- INT8 calibration (see the TF-TRT User Guide for more information about how to use INT8) is a very slow process that can take 1 hour depending on the model. We are working on optimizing this algorithm in TensorRT.

- The pip package of TensorFlow 1.14 released by Google is missing TensorRT. This will be fixed in the next release of TensorFlow by Google. In the meantime, you can use the NVIDIA container for TensorFlow.

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.train.Optimizer or tf.keras.optimizers.Optimizer` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`. For backward compatibility with AMP in previous containers, AMP can also be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) by setting the following flag in the `tf.session` config:

```
config.graph_options.rewrite_options.auto_mixed_precision=1
```

Or equivalently for backward compatibility with AMP in previous NGC containers, by setting the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is a known performance regression in TensorFlow 1.14.0 affecting a variety of models. Affected models include GNMT, SSD, and UNet. Performance regressions can be as high as 20% compared to TensorFlow 1.13.1 in the 19.06 release.
- For BERT Large training with the 19.08 release on Tesla V100 boards with 16 GB memory, performance with batch size 3 per GPU is lower than expected; batch size 2 per GPU may be a better choice for this model on these GPUs with the 19.08 release. 32 GB GPUs are not affected.
- TensorBoard has a bug in its IPv6 support which can result in the following error: `Tensorboard could not bind to unsupported address family ::`. To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- In previous containers, `libtensorflow_framework.so` was available in the `/usr/local/lib/tensorflow` directory. This was redundant with the libs installed with the TensorFlow pip package. To find the TensorFlow `lib` directory, use `tf.sysconfig.get_lib()`.
- Automatic Mixed Precision (AMP) does not support the Keras `LearningRateScheduler` in the 19.08 release. A fix will be included in a future release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.

Chapter 70. TensorFlow Release 19.07

The NVIDIA container image of TensorFlow, release 19.07, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 18.04

  > **Note:** Container image `19.07-py2` contains Python 2.7; `19.07-py3` contains Python 3.6.

- NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- NVIDIA cuDNN 7.6.1
- NVIDIA NCCL 2.4.7 (optimized for NVLink™)
- Horovod 0.16.2
- OpenMPI 3.1.3
- TensorBoard 1.14.0+nv
- MLNX_OFED +3.4
- OpenSeq2Seq at commit 27346d1
- TensorRT 5.1.5
- DALI 0.11.0 Beta
- DLProf 19.07
- Nsight Compute 10.1.168
- Nsight Systems 2019.3.6.30
- Tensor Core optimized example:
  - U-Net Medical

- SSD320 v1.2
- Neural Collaborative Filtering (NCF)
- BERT
- U-Net Industrial Defect Segmentation
- GNMT v2
- ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0
  - Jupyter Notebook 5.7.8
  - JupyterLab 1.0.1
  - JupyterLab Server 1.0.0
  - Jupyter-TensorBoard

Driver Requirements

Release 19.07 is based on NVIDIA CUDA 10.1.168, which requires NVIDIA Driver release 418.67. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.07 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.07 is based on TensorFlow 1.14.0.
- Automatic Mixed Precision updated with latest upstream changes (see below)
- Latest version of Nsight Systems 2019.3.6.30
- Latest version of Python 3.6
- Latest version TensorBoard 1.14.0 with additional updates from NVIDIA
- Latest version of NVIDIA cuDNN 7.6.1
- Latest versions of Jupyter Client 5.3.1, Jupyter Core 4.5.0, JupyterLab 1.0.1 and JupyterLab Server 1.0.0, including Jupyter-TensorBoard integration.

- Latest version of DLProf 19.07
- Latest version of DALI 0.11.0 Beta
- Latest version of Ubuntu 18.04

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Migrated TensorRT conversion sources from the `contrib` directory to the `compiler` directory in preparation for TensorFlow 2.0. The Python code can be found at `//tensorflow/python/compiler/tensorrt`.
- Added a user friendly `TrtGraphConverter` API for TensorRT conversion.
- Expanded support for TensorFlow operators in TensorRT conversion (for example, `Gather`, `Slice`, `Pack`, `Unpack`, `ArgMin`, `ArgMax`, `DepthSpaceShuffle`). Refer to the TF-TRT User Guide for a complete list of supported operators.
- Support added for TensorFlow operator `CombinedNonMaxSuppression` in TensorRT conversion which significantly accelerates SSD object detection models.
- Integrated TensorRT 5.1.5 into TensorFlow. See the TensorRT 5.1.5 Release Notes for a full list of new features.

Deprecated Features

- The old API of TF-TRT is deprecated. It still works in TensorFlow 1.14, however, it may be removed in TensorFlow 2.0. The old API is a Python function named `create_inference_graph` which is not replaced by the Python class `TrtGraphConverter` with a number of methods. Refer to TF-TRT User Guide for more information about the API and how to use it

Known Issues

- Precision mode in the TF-TRT API is a string with one of the following values: `FP32`, `FP16` or `INT8`. In TensorFlow 1.13, these strings were supported in lowercase, however, in TensorFlow 1.14 only uppercase is supported.
- INT8 calibration (see the TF-TRT User Guide for more information about how to use INT8) is a very slow process that can take 1 hour depending on the model. We are working on optimizing this algorithm in TensorRT.
- The pip package of TensorFlow 1.14 released by Google is missing TensorRT. This will be fixed in the next release of TensorFlow by Google. In the meantime, you can use the NVIDIA container for TensorFlow.

Announcements

We will stop support for Python 2.7 in a future TensorFlow container release. Once support has ended, the TensorFlow container will contain only one version of Python.

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.train.Optimizer or tf.keras.optimizers.Optimizer` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by wrapping the optimizer with `tf.train.experimental.enable_mixed_precision_graph_rewrite()`. For backward compatibility with AMP in previous containers, AMP can also be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) with the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration.

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository.

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy.
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007.
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, tensor cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training.

Known Issues

- There is a known performance regression in TensorFlow 1.14.0 affecting a variety of models. Affected models include GNMT, SSD, and UNet. Performance regressions can be as high as 20% compared to TensorFlow 1.13.1 in the 19.06 release.
- There is an issue in TensorFlow 1.14 that increases the GPU memory footprint of certain models such as BERT. As a result, training may need to be performed with a reduced batch size.

- TensorBoard has a bug in its IPv6 support which can result in the following error:
  `Tensorboard could not bind to unsupported address family ::.`
  To workaround this error, pass the `--host <IP>` flag when starting TensorBoard.
- In previous containers, `libtensorflow_framework.so` was available in the `/usr/local/lib/tensorflow` directory. This was redundant with the libs installed with the TensorFlow pip package. To find the TensorFlow `lib` directory, use `tf.sysconfig.get_lib()`.
- Automatic Mixed Precision (AMP) does not support the Keras `LearningRateScheduler` in the 19.07 release. A fix will be included in the 19.08 release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- Using `TF_ENABLE_NHWC=1` might cause memory leak (OOM) if `FusedBatchNormV3` is explicitly used. By default, `tf.nn.fused_batch_norm()` uses `FusedBatchNorm` and `FusedBatchNormV2`. The `FusedBatchNormV3` is set to be available after November 11th, 2019. A fix will be included in the 19.08 release.

Chapter 71. TensorFlow Release 19.06

The NVIDIA container image of TensorFlow, release 19.06, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `19.06-py2` contains Python 2.7; `19.06-py3` contains Python 3.5.

- NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- NVIDIA cuDNN 7.6.0
- NVIDIA NCCL 2.4.7 (optimized for NVLink™)
- Horovod 0.16.2
- OpenMPI 3.1.3
- TensorBoard 1.13.1+nv
- MLNX_OFED 3.4
- OpenSeq2Seq at commit 27346d1
- TensorRT 5.1.5
- DALI 0.10.0 Beta
- DLProf 19.06
- Nsight Compute 10.1.168
- Nsight Systems 2019.3.1.94
- Tensor Core optimized example:
  - U-Net Medical

    - SSD320 v1.2
    - Neural Collaborative Filtering (NCF)
    - BERT
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.2.4
    - Jupyter Core 4.4.0
    - Jupyter Notebook 5.7.8
    - JupyterLab 0.35.6
    - JupyterLab Server 0.2.0

Driver Requirements

Release 19.06 is based on NVIDIA CUDA 10.1.168, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.06 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.06 is based on TensorFlow 1.13.1.
- Latest version of NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- Latest version of NVIDIA NCCL 2.4.7
- Latest version of DALI 0.10.0 Beta
- Latest version of JupyterLab 0.35.6
- Latest version of Horovod 0.16.2
- Latest version of Nsight Compute 10.1.168
- Latest OpenSeq2Seq at commit 27346d1

- Added DLProf 19.06 software. Deep Learning Profiler (DLProf) is a tool for profiling deep learning models to help data scientists understand and improve performance of their models visually via TensorBoard or by analyzing text reports.
- Determinism - Setting the environment variable `TF_CUDNN_DETERMINISM=1` forces the selection of deterministic cuDNN convolution and max-pooling algorithms. When this is enabled, the algorithm selection procedure itself is also deterministic.

  Alternatively, setting `TF_DETERMINISTIC_OPS=1` has the same effect and additionally makes any bias addition that is based on `tf.nn.bias_add()` (for example, in Keras layers) operate deterministically on GPU. If you set `TF_DETERMINISTIC_OPS=1` then there is no need to also set `TF_CUDNN_DETERMINISM=1`.

  Selecting these deterministic options may reduce performance.
- Ubuntu 16.04 with May 2019 updates (see Announcements)

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Integrated TensorRT 5.1.5 into TensorFlow. See the TensorRT 5.1.5 Release Notes for a full list of new features.
- Improved examples at GitHub: TF-TRT, including README files, build scripts, benchmark mode, ResNet models from TensorFlow official model zoo, etc...

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.Optimizer()` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) with the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration.
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository.
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy.
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007.
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is

in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, tensor cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training.

Announcements

In the next release, we will no longer support Ubuntu 16.04. Release 19.07 will instead support Ubuntu 18.04.

Known Issues

- There is a known performance regression with TensorFlow 1.13.1 for some networks when run with small batch sizes. As a workaround, increase the batch size.
- The AMP preview implementation is not compatible with Distributed Strategies. We recommend using Horovod for parallel training with AMP.
- AMP is not compatible with models the use ResourceVariables for the global_step passed to the `tf.train.Optimizer.apply_gradients`. This will be fixed in the 19.07 NGC release.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.
- Support for CUDNN float32 Tensor Op Math mode first introduced in the 18.09 release is now deprecated in favor of Automatic Mixed Precision. It is scheduled to be removed after the 19.11 release.
- DLProf and Nsight Systems in the container will not work with GPU drivers newer than r418.
- There is a known issue when running the 19.06 TensorFlow container on a DGX-2 (or other systems having more than 8 GPUs) with RHEL 7.x (as opposed to Ubuntu) as the operating system. The known issue is that in some circumstances you will be shown the following message:

```
E tensorflow/stream_executor/cuda/cuda_driver.cc:300] failed call to cuInit:
 CUDA_ERROR_OPERATING_SYSTEM: OS call failed or operation not supported on this OS
```

Chapter 72. TensorFlow Release 19.05

The NVIDIA container image of TensorFlow, release 19.05, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `19.05-py2` contains Python 2.7; `19.05-py3` contains Python 3.5.

- NVIDIA CUDA 10.1 Update 1 including cuBLAS 10.1 Update 1
- NVIDIA cuDNN 7.6.0
- NVIDIA NCCL 2.4.6 (optimized for NVLink™)
- Horovod 0.16.1
- OpenMPI 3.1.3
- TensorBoard 1.13.1
- MLNX_OFED 3.4
- OpenSeq2Seq at commit 6e8835f
- TensorRT 5.1.5
- DALI 0.9.1 Beta
- Nsight Compute 10.1.163
- Nsight Systems 2019.3.1.94
- Tensor Core optimized example:
  - U-Net Medical
  - SSD320 v1.2

    - Neural Collaborative Filtering (NCF)
    - Bert
    - U-Net Industrial Defect Segmentation
    - GNMT v2
    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.2.4
    - Jupyter Core 4.4.0
    - JupyterLab 0.35.4
    - JupyterLab Server 0.2.0

Driver Requirements

Release 19.05 is based on CUDA 10.1 Update 1, which requires NVIDIA Driver release 418.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.05 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.05 is based on TensorFlow 1.13.1.
- Latest version of NVIDIA CUDA 10.1 Update 1 including cuBLAS 10.1 Update 1
- Latest version of NVIDIA cuDNN 7.6.0
- Latest version of TensorRT 5.1.5
- Latest version of DALI 0.9.1 Beta
- Latest version of Nsight Compute 10.1.163
- Added the U-Net Medical Tensor Core example
- Added the NHWC plumbing to remove unnecessary format conversions between NHWC and NCHW. This feature is disabled by default, but can be enabled by setting the environment variable `TF_ENABLE_NHWC=1`.
- Ubuntu 16.04 with April 2019 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Integrated TensorRT 5.1.5 into TensorFlow. See the TensorRT 5.1.5 Release Notes for a full list of new features.
- Improved examples at GitHub: TF-TRT, including README files, build scripts, benchmark mode, ResNet models from TensorFlow official model zoo, etc...

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.Optimizer()` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) with the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration.

- An implementation of the SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository.
- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.
- An implementation of the Bert model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy.
- An implementation of the U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007.
- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.
- An implementation of the ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, tensor cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training.

Known Issues

- There is a known performance regression with TensorFlow 1.13.1 for some networks when run with small batch sizes. As a workaround, increase the batch size.
- The AMP preview implementation is not compatible with Distributed Strategies. We recommend using Horovod for parallel training with AMP.
- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.

Chapter 73. TensorFlow Release 19.04

The NVIDIA container image of TensorFlow, release 19.04, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `19.04-py2` contains Python 2.7; `19.04-py3` contains Python 3.5.

- NVIDIA CUDA 10.1.105 including cuBLAS 10.1.0.105
- NVIDIA cuDNN 7.5.0
- NVIDIA NCCL 2.4.6 (optimized for NVLink™)
- Horovod 0.16.1
- OpenMPI 3.1.3
- TensorBoard 1.13.1
- MLNX_OFED 3.4
- OpenSeq2Seq at commit 6e8835f
- TensorRT 5.1.2
- DALI 0.8.1 Beta
- Nsight Compute 10.1.105
- Nsight Systems 2019.3.1.8
- Tensor Core optimized example:
  - SSD320 v1.2
  - Neural Collaborative Filtering (NCF)

- Bert
- U-Net Industrial Defect Segmentation
- GNMT v2
- ResNet-50 v1.5

- Jupyter and JupyterLab:
  - Jupyter Client 5.2.4
  - Jupyter Core 4.4.0
  - JupyterLab 0.35.4
  - JupyterLab Server 0.2.0

Driver Requirements

Release 19.04 is based on CUDA 10.1, which requires NVIDIA Driver release 418.xx.x +. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.04 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.04 is based on TensorFlow 1.13.1.
- Added the GNMT v2, U-Net Industrial Defect Segmentation, Bert, Neural Collaborative Filtering (NCF), and SSD320 v1.2 Tensor Core examples
- Latest version of NVIDIA NCCL 2.4.6
- Latest version of cuBLAS 10.1.0.105
- Latest version of DALI 0.8.1 Beta
- Latest version of Nsight Systems 2019.3.1.8
- Latest version of Horovod 0.16.1
- Improved stability for auto-tuning of fastest convolutional algorithms.
- Ubuntu 16.04 with March 2019 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Integrated TensorRT 5.1.2 RC into TensorFlow. See the TensorRT 5.1.2 RC Release Notes for a full list of new features.
- Improved examples at GitHub: TF-TRT, including README files, build scripts, benchmark mode, ResNet models from TensorFlow official model zoo, etc...

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.Optimizer()` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) with the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository.

- An implementation of the Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item.
- An implementation of the Bert model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy.
- An implementation of the U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007.
- An implementation of the GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep.
- An implementation of the ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, tensor cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training.

Known Issues

- There is a known performance regression with TensorFlow 1.13.1 for some networks when run with small batch sizes. As a workaround, increase the batch size.
- The AMP preview implementation is not compatible with Distributed Strategies. We recommend using Horovod for parallel training with AMP.

- A known issue in TensorFlow results in the error `Cannot take the length of Shape with unknown rank` when training variable sized images with the Keras `model.fit` API. Details are provided here and a fix will be available in a future release.

Chapter 74. TensorFlow Release 19.03

The NVIDIA container image of TensorFlow, release 19.03, is available on NGC.

Contents of the TensorFlow container

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `19.03-py2` contains Python 2.7; `19.03-py3` contains Python 3.5.

- NVIDIA CUDA 10.1.105 including cuBLAS 10.1.105
- NVIDIA cuDNN 7.5.0
- NVIDIA NCCL 2.4.3 (optimized for NVLink™)
- Horovod 0.16.0
- OpenMPI 3.1.3
- TensorBoard 1.13.1
- MLNX_OFED 3.4
- OpenSeq2Seq at commit 6e8835f
- TensorRT 5.1.2
- DALI 0.7 Beta
- Nsight Compute 10.1.105
- Nsight Systems 10.1.105
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:

- Jupyter Client 5.2.4
- Jupyter Core 4.4.0
- JupyterLab 0.35.4
- JupyterLab Server 0.2.0

Driver Requirements

Release 19.03 is based on CUDA 10.1, which requires NVIDIA Driver release 418.xx+. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384.111+ or 410. The CUDA driver's compatibility package only supports particular drivers. For a complete list of supported drivers, see the CUDA Application Compatibility topic. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.03 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.03 is based on TensorFlow 1.13.1.
- Latest version of NVIDIA CUDA 10.1.105 including cuBLAS 10.1.105
- Latest version of NVIDIA cuDNN 7.5.0
- Latest version of NVIDIA NCCL 2.4.3
- Latest version of DALI 0.7 Beta
- Latest version of TensorRT 5.1.2
- Latest version of Horovod 0.16.0
- Latest version of TensorBoard 1.13.1
- Added the ResNet-50 v1.5 Tensor Core example
- Added Nsight Compute 10.1.105 and Nsight Systems 10.1.105 software
- Added support for TensorFlow Automatic Mixed Precision (TF-AMP); see below for more information.
- Ubuntu 16.04 with February 2019 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Integrated TensorRT 5.1.2 RC into TensorFlow. See the TensorRT 5.1.2 RC Release Notes for a full list of new features.
- Improved examples at GitHub: TF-TRT, including README files, build scripts, benchmark mode, ResNet models from TensorFlow official model zoo, etc...

Announcements

TensorRT 3.x is not longer supported, therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 5.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Automatic Mixed Precision (AMP)

Automatic mixed precision converts certain float32 operations to operate in float16 which can run much faster on Tensor Cores. Automatic mixed precision is built on two components:

- a loss scaling optimizer
- graph rewriter

For models already using a `tf.Optimizer()` for both `compute_gradients()` and `apply_gradients()` operations, automatic mixed precision can be enabled by defining the following environment variable before calling the usual float32 training script:

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

Models implementing their own optimizers can use the graph rewriter on its own (while implementing loss scaling manually) with the following environment variable:

```
export TF_ENABLE_AUTO_MIXED_PRECISION_GRAPH_REWRITE=1
```

For more information about how to access and enable Automatic mixed precision for TensorFlow, see Automatic Mixed Precision Training In TensorFlow from the TensorFlow User Guide, along with Training With Mixed Precision.

Tensor Core Examples

These examples focus on achieving the best performance and convergence from NVIDIA Volta Tensor Cores by using the latest deep learning example networks for training.

Each example model trains with mixed precision Tensor Cores on Volta, therefore you can get results much faster than training without tensor cores. This model is tested against each NGC monthly container release to ensure consistent accuracy and performance over time. This container includes the following tensor core examples.

- An implementation of the ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for tensor cores (mixed precision) training.

Known Issues

- There is a known performance regression with TensorFlow 1.13.1 for some networks when run with small batch sizes. As a workaround, increase the batch size.
- The AMP preview implementation is not compatible with Distributed Strategies. We recommend using Horovod for parallel training with AMP.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 75. TensorFlow Release 19.02

The NVIDIA container image of TensorFlow, release 19.02, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `19.02-py2` contains Python 2.7; `19.02-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.2
- NVIDIA Collective Communications Library (NCCL) 2.3.7 (optimized for NVLink™)
- Horovod 0.15.1
- OpenMPI 3.1.3
- TensorBoard 1.12.2
- MLNX_OFED 3.4
- OpenSeq2Seq v18.12 at commit 59c70e7
- TensorRT 5.0.2
- DALI 0.6.1 Beta
- Jupyter and JupyterLab:
  - Jupyter Client 5.2.4
  - Jupyter Core 4.4.0
  - JupyterLab 0.35.4

- JupyterLab Server 0.2.0

Driver Requirements

Release 19.02 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.02 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.02 is based on TensorFlow 1.13.0-rc0.
- Latest version of DALI 0.6.1 Beta
- Latest version of TensorBoard 1.12.2
- Added Jupyter and JupyterLab software in our packaged container.
- Latest version of jupyter_client 5.2.4
- Latest version of jupyter_core 4.4.0
- Ubuntu 16.04 with January 2019 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- The following operators can now be converted from TensorFlow to TensorRT: `ExpandDims`, `Reshape`, `Sigmoid`, `Sqrt`, `Square`, `Squeeze`, `StridedSlice` and `Tanh`. For more information, see Supported Ops.
- You can manually insert quantization ranges (generated during quantization-aware training) to the graph, and then TF-TRT can use them during INT8 inference. That means calibration is not required with this feature. For more information, see INT8 Quantization.

Deprecated Features

- Support for TensorRT 3 has been removed.

Announcements

TensorRT 3.x is not longer supported, therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 5.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

- Horovod and XLA cannot be used together due to a known issue in upstream TensorFlow. We expect this to be resolved in an upcoming release.
- There is a known performance regression with TensorFlow 1.13.0-rc0 for some networks when run with small batch sizes. As a workaround, increase the batch size.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 76. TensorFlow Release 19.01

The NVIDIA container image of TensorFlow, release 19.01, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `19.01-py2` contains Python 2.7; `19.01-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.2
- NCCL 2.3.7 (optimized for NVLink™)
- Horovod 0.15.1
- OpenMPI 3.1.3
- TensorBoard 1.12.0
- MLNX_OFED 3.4
- OpenSeq2Seq v18.12 at commit 59c70e7
- TensorRT 5.0.2
- DALI 0.6 Beta

Driver Requirements

Release 19.01 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you

may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 19.01 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 19.01 is based on TensorFlow 1.12.0.
- Latest version of DALI 0.6 Beta
- Latest version of NVIDIA cuDNN 7.4.2
- Latest version of OpenMPI 3.1.3
- Ubuntu 16.04 with December 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Deprecated Features

- The image-classification examples were moved from `/opt/tensorflow/nvidia-examples/inference/image-classification/scripts` to https://github.com/tensorflow/tensorrt/tree/master/tftrt/examples/image-classification.
- The `check_accuracy.py` script, used to check whether the accuracy generated by the example matches with the expectation, was removed from the example. Refer to the published accuracy numbers to verify whether your generated accuracy numbers match with the expectation.

Announcements

Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

- Horovod and XLA cannot be used together due to a known issue in upstream TensorFlow. We expect this to be resolved in an upcoming release.
- If using or upgrading to a 3-part-version driver, for example, a driver that takes the format of `xxx.yy.zz`, you will receive a `Failed to detect NVIDIA driver version.` message. This is due to a known bug in the entry point script's parsing of 3-part driver versions. This message is non-fatal and can be ignored. This will be fixed in the 19.04 release.

Chapter 77. TensorFlow Release 18.12

The NVIDIA container image of TensorFlow, release 18.12, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `18.12-py2` contains Python 2.7; `18.12-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.1
- NCCL 2.3.7 (optimized for NVLink™)
- Horovod 0.15.1
- OpenMPI 3.1.2
- TensorBoard 1.12.0
- MLNX_OFED 3.4
- OpenSeq2Seq v18.12 at commit 59c70e7
- TensorRT 5.0.2
- DALI 0.5.0 Beta

Driver Requirements

Release 18.12 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you

may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

GPU Requirements

Release 18.12 supports CUDA compute capability 6.0 and higher. This corresponds to GPUs in the Pascal, Volta, and Turing families. Specifically, for a list of GPUs that this compute capability corresponds to, see CUDA GPUs. For additional support details, see Deep Learning Frameworks Support Matrix.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.12 is based on TensorFlow 1.12.0.
- Latest version of DALI 0.5.0 Beta.
- OpenSeq2Seq's custom CTC decoder is now pre-built in the container.
- The `tensorflow.contrib.nccl` module has been moved into core as `tensorflow.python.ops.nccl_ops`. User scripts may need to be updated accordingly. No changes are required for Horovod users. For an example of using Horovod, refer to the `nvidia-examples/cnn/` directory.
- Inference image classification examples have been removed from the container and are now available at: GitHub: TensorFlow/TensorRT Integration.
- Ubuntu 16.04 with November 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Deprecated Features

- The image-classification examples were moved from `/opt/tensorflow/nvidia-examples/inference/image-classification/scripts` to https://github.com/tensorflow/tensorrt/tree/master/tftrt/examples/image-classification.
- The `check_accuracy.py` script, used to check whether the accuracy generated by the example matches with the expectation, was removed from the example. Refer to the published accuracy numbers to verify whether your generated accuracy numbers match with the expectation.

Announcements

Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

- OpenSeq2Seq is only supported in the Python 3 container.
- Horovod and XLA cannot be used together due to a known issue in upstream TensorFlow. We expect this to be resolved in an upcoming release.

Chapter 78. TensorFlow Release 18.11

The NVIDIA container image of TensorFlow, release 18.11, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `18.11-py2` contains Python 2.7; `18.11-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.1
- NCCL 2.3.7 (optimized for NVLink™)
- Horovod 0.15.1
- OpenMPI 3.1.2
- TensorBoard 1.12.0
- MLNX_OFED 3.4
- OpenSeq2Seq v18.11 at commit 4b95346
- TensorRT 5.0.2
- DALI 0.4.1 Beta

Driver Requirements

Release 18.11 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you

may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.11 is based on TensorFlow 1.12.0-rc2.
- Latest version of Horovod 0.15.1.
- Latest version of NCCL 2.3.7.
- Latest version of NVIDIA cuDNN 7.4.1.
- Latest version of TensorRT 5.0.2
- Latest version of DALI 0.4.1 Beta.
- Bug fixes and improvements for TensorFlow-TensorRT (TF-TRT) integration.
- Added an object detection example to `workspace/nvidia-examples/inference/object-detection`.
- Ubuntu 16.04 with October 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Added support for dilated convolution.
- Fixed a bug in the `Identity` op.
- Fixed a bug in the `Relu6` op.
- Support added to allow empty const tensor.
- Added object detection example to `nvidia-examples/inference`.

Known Issues

- In the TF-TRT API, the `minimum_segment_size` argument default value is 3. In the image classification examples under `nvidia-examples/inference`, we define a command line argument for `minimum_segment_size` which has its own default value. In 18.10, the default value was `7` and in 18.11 we changed it to `2`. Smaller values for this argument would cause to convert more TensorFlow nodes to TensorRT which typically should improve the performance, however, we have observed cases where the performance gets worse. In particular, Resnet-50 with smaller batch sizes gets slower with `minimum_segment_size=2` comparing to `minimum_segment_size=7`.

Announcements

Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

OpenSeq2Seq is only supported in the Python 3 container.

Chapter 79. TensorFlow Release 18.10

The NVIDIA container image of TensorFlow, release 18.10, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

> **Note:** Container image `18.10-py2` contains Python 2.7; `18.10-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.4.0
- NCCL 2.3.6 (optimized for NVLink™)
- Horovod 0.13.10
- OpenMPI 3.1.2
- TensorBoard 1.10.0
- MLNX_OFED 3.4
- OpenSeq2Seq v18.10 at commit 655eb65
- TensorRT 5.0.0 RC
- DALI 0.4 Beta

Driver Requirements

Release 18.10 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you

may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.10 is based on TensorFlow 1.10.0.
- Latest version of NCCL 2.3.6.
- Latest version of DALI 0.4 Beta
- Latest version of OpenMPI 3.1.2
- Fixed a bug in the ResNet example script when using `NHWC` data format.
- Fixed several issues when accelerating inference in TensorFlow with TensorRT including support for ReLu6, Identity, and dilated convolutions.
- Ubuntu 16.04 with September 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- New examples at `nvidia-examples/tftrt` with good accuracy and performance.
- Built TF-TRT with TensorRT 5.0.0 which introduces the new TensorRT APIs into TF-TRT.
- Added support for the TensorFlow operator RELU6 (using `Relu6(x) = min(Relu(x), 6)`).
- Made improvements in the image classification example, such as bug fixes and using the `dynamic_op` feature.

Limitations

- Not all the new TensorRT 5.0.0 features are supported yet in TF-TRT including INT8 quantization ranges and the plugins registry.
- We have only tested image classification models with TF-TRT including the ones we have provided in our examples inside the container (`nvidia-examples/tftrt`). This means object detection, translation (convolutional and recurrent based) are not yet supported due to either functionality or performance limitations.
- TF-TRT has an implementation of optimizing the TensorFlow graph by specifying appropriate TensorFlow session arguments without using the Python TF-TRT API (`create_inference_graph`), however, we have not thoroughly tested this functionality yet, therefore, we don’t support it.

Known Issues

- Running inference with batch sizes larger than the maximum batch size is not supported by TensorRT.
- Due to certain logs (errors or warnings) of TF-TRT, they could be misleading and point to the TensorRT graph as broken while it's not. It is recommended to check whether there is any TensorRT op in the graph (the type of op is `TRTEngineOp`). If there is not TensorRT ops in the graph, that means no conversion has happened and the inference should fall back to the native TensorFlow. Currently, the best way to verify whether a frozen graph resulting from the conversion is not broken is to run inference on it and check the accuracy of the results.
- There are operators that are not supported by either TensorRT or the conversion algorithm. The convertor is supposed to skip these ops but this skip may not happen properly due to bugs. One way to get around this problem is to increase the value of the `minimum_segment_size` parameter and hope that the subgraphs that contain those ops are too small and remain out of the conversion.
- We have observed functionality problems in optimizing:
  - NASNet models with TF-TRT in FP16 precision mode.
  - ResNet, MobileNet, and NASNet models with TF-TRT in INT8 precision mode.

  > **Note:** TF-TRT cannot optimize certain models such as ResNet in INT8 precision mode because of a lacking feature in TensorRT regarding the dimensionality of tensors. Usually, increasing the value of `minimum_segment_size` is a workaround by removing those unsupported dimensions out of the TensorRT sub-graph.
- TF-TRT doesn't work with TensorFlow Lite due to a TensorRT bug that causes Flatbuffer symbols to be exposed. This means you cannot import both `tf.contrib.tensorrt` and `tf.lite` in the same process.
- We have observed a bit low accuracy on image classification models with TF-TRT on Jetson AGX Xavier.
- INT8 calibration on `mobilenet_v1` and `mobilenet_v2` using TF-TRT fails if the calibration dataset has only one element.

Announcements

Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the **Note** in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

OpenSeq2Seq is only supported in the Python 3 container.

Chapter 80. TensorFlow Release 18.09

The NVIDIA container image of TensorFlow, release 18.09, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `18.09-py2` contains Python 2.7; `18.09-py3` contains Python 3.5.

- NVIDIA CUDA 10.0.130 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0.130
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.3.0
- NCCL 2.3.4 (optimized for NVLink™)
- Horovod™ 0.13.10
- OpenMPI 3.0.0
- TensorBoard 1.10.0
- MLNX_OFED 3.4
- OpenSeq2Seq v18.09 at commit 694a230
- TensorRT 5.0.0 RC
- DALI 0.2 Beta

Driver Requirements

Release 18.09 is based on CUDA 10, which requires NVIDIA Driver release 410.xx. However, if you are running on Tesla (Tesla V100, Tesla P4, Tesla P40, or Tesla P100), you

may use NVIDIA driver release 384. For more information, see CUDA Compatibility and Upgrades.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.09 is based on TensorFlow 1.10.0.
- Latest version of cuDNN 7.3.0.
- Latest version of CUDA 10.0.130 which includes support for DGX-2, Turing, and Jetson Xavier.
- Latest version of cuBLAS 10.0.130.
- Latest version of NCCL 2.3.4.
- Latest version of TensorRT 5.0.0 RC.
- Latest version of TensorBoard 1.10.0.
- Latest version of DALI 0.2 Beta
- Added support for CUDNN float32 Tensor Op Math mode, which enables float32 models to use Tensor Cores on supported hardware, at the cost of reduced precision. This is disabled by default, but can be enabled by setting the environment variables `TF_ENABLE_CUDNN_TENSOR_OP_MATH_FP32=1` (for convolutions) or `TF_ENABLE_CUDNN_RNN_TENSOR_OP_MATH_FP32=1` (for RNNs that use the `cudnn_rnn` op). This feature is currently considered experimental.
- Renamed the existing environment variable `TF_ENABLE_TENSOR_OP_MATH_FP32` to `TF_ENABLE_CUBLAS_TENSOR_OP_MATH_FP32`.

  > **Note:** When using any of the `TF_ENABLE_*_TENSOR_OP_MATH_FP32` environment variables, it is recommended that models also use *loss scaling* to avoid numerical issues during training. For more information about loss scaling, see Training With Mixed Precision.

- Enhanced `tf.contrib.layers.layer_norm` by adding a `use_fused_batch_norm` parameter that improves performance. This parameter is disabled by default, but can be enabled by setting it to `True`.
- Ubuntu 16.04 with August 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- New examples at `nvidia-examples/tftrt` with good accuracy and performance.
- Built TF-TRT with TensorRT 5.0.0 which introduces the new TensorRT APIs into TF-TRT.

Limitations

- Not all the new TensorRT 5.0.0 features are supported yet in TF-TRT including INT8 quantization ranges and the plugins registry.
- We have only tested image classification models with TF-TRT including the ones we have provided in our examples inside the container (`nvidia-examples/tftrt`). This means object detection, translation (convolutional and recurrent based) are not yet supported due to either functionality or performance limitations.
- TF-TRT has an implementation of optimizing the TensorFlow graph by specifying appropriate TensorFlow session arguments without using the Python TF-TRT API (`create_inference_graph`), however, we have not thoroughly tested this functionality yet, therefore, we don't support it.
- TF-TRT has an implementation of the dynamic conversion of a TensorFlow graph, but we have not thoroughly tested this functionality yet, therefore, we don't support it.

Known Issues

- Running inference with batch sizes larger than the maximum batch size is not supported by TensorRT.
- Due to certain logs (errors or warnings) of TF-TRT, they could be misleading and point to the TensorRT graph as broken while it's not. It is recommended to check whether there is any TensorRT op in the graph (the type of op is `TRTEngineOp`). If there is not TensorRT ops in the graph, that means no conversion has happened and the inference should fall back to the native TensorFlow. Currently, the best way to verify whether a frozen graph resulting from the conversion is not broken is to run inference on it and check the accuracy of the results.
- There are operators that are not supported by either TensorRT or the conversion algorithm. The convertor is supposed to skip these ops but this skip may not happen properly due to bugs. One way to get around this problem is to increase the value of the `minimum_segment_size` parameter and hope that the subgraphs that contain those ops are too small and remain out of the conversion.
- We have observed functionality problems in optimizing:
  - NASNet models with TF-TRT in FP16 precision mode.
  - ResNet, MobileNet, and NASNet models with TF-TRT in INT8 precision mode.

> **Note:** TF-TRT cannot optimize certain models such as ResNet in INT8 precision mode because of a lacking feature in TensorRT regarding the dimensionality of tensors. Usually, increasing the value of `minimum_segment_size` is a workaround by removing those unsupported dimensions out of the TensorRT sub-graph.

- TF-TRT doesn’t work with TensorFlow Lite due to a TensorRT bug that causes Flatbuffer symbols to be exposed. This means you cannot import both `tf.contrib.tensorrt` and `tf.lite` in the same process.
- We have observed a bit low accuracy on image classification models with TF-TRT on Jetson AGX Xavier.
- INT8 calibration on `mobilenet_v1` and `mobilenet_v2` using TF-TRT fails if the calibration dataset has only one element.

Announcements

Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the **Note** in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

- OpenSeq2Seq is only supported in the Python 3 container.
- The `build_imagenet_data` scripts have a missing dependency on the `axel` application. This can be resolved by issuing the following command:

  ```
  apt-get update &&
  apt-get install axel
  ```

Chapter 81. TensorFlow Release 18.08

The NVIDIA container image of TensorFlow, release 18.08, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

> **Note:** Container image `18.08-py2` contains Python 2.7; `18.08-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.2.1
- NCCL 2.2.13 (optimized for NVLink™)
- Horovod™ 0.12.1
- OpenMPI™ 3.0.0
- TensorBoard 1.9.0
- MLNX_OFED 3.4
- OpenSeq2Seq v0.5 at commit 83e96551.
- TensorRT 4.0.1
- DALI 0.1.2 Beta

Driver Requirements

Release 18.08 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.08 is based on TensorFlow 1.9.0.
- Latest version of cuDNN 7.2.1.
- Latest version of DALI 0.1.2 Beta.
- Latest version of TensorBoard 1.9.0.
- Added experimental support for float16 data type in Horovod, allowing functions such as `all_reduce` to accept tensors in float16 precision. (This functionality is not yet integrated into multi-GPU training examples).
- Ubuntu 16.04 with July 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- TensorRT conversion has been integrated into optimization pass. The `tensorflow/contrib/tensorrt/test/test_tftrt.py` script has an example showing the use of optimization pass.

Limitations

- TensorRT conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.
- Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.
- Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.
- Current optimization pass does not support INT8 yet.

Known Issues

- Input tensors are required to have rank 4 for quantization mode (INT8 precision).

Announcements

Starting with the next major version of CUDA release, we will no longer provide updated Python 2 containers and will only update Python 3 containers.

Known Issues

- The DALI integrated ResNet-50 samples in the 18.08 NGC TensorFlow container has lower than expected accuracy and performance results. We are working to address the issue in the next release.
- There is a known performance regression in the inference benchmarks for ResNet-50. We haven't seen this regression in the inference benchmarks for VGG or training benchmarks for any network. The cause of the regression is still under investigation.

Chapter 82. TensorFlow Release 18.07

The NVIDIA container image of TensorFlow, release 18.07, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

> **Note:** Container image `18.07-py2` contains Python 2.7; `18.07-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.4
- NCCL 2.2.13 (optimized for NVLink™)
- Horovod™ 0.12.1
- OpenMPI™ 3.0.0
- TensorBoard 1.8.0
- MLNX_OFED 3.4
- OpenSeq2Seq v0.4 at commit 98ad236a.
- TensorRT 4.0.1
- DALI 0.1 Beta

Driver Requirements

Release 18.07 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.07 is based on TensorFlow 1.8.0.
- Added support for DALI 0.1 Beta.
- Latest version of CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.425.
- Ubuntu 16.04 with June 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Added TensorRT 4.0 API support with extended layer support. This support includes the FullyConnected layer and `BatchedMatMul` op.
- Resource management added, where memory allocation is uniformly managed by TensorFlow.
- Bug fixes and better error handling in conversion.

Limitations

- TensorRT conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.
- Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.
- Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.

Known Issues

- Input tensors are required to have rank 4 for quantization mode (INT8 precision).

Announcements

Starting with the next major version of CUDA release, we will no longer provide updated Python 2 containers and will only update Python 3 containers.

Known Issues

There are no known issues in this release.

Chapter 83. TensorFlow Release 18.06

The NVIDIA container image of TensorFlow, release 18.06, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

> **Note:** Container image `18.06-py2` contains Python 2.7; `18.06-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.4
- NCCL 2.2.13 (optimized for NVLink™)
- Horovod™ 0.12.1
- OpenMPI™ 3.0.0
- TensorBoard 1.8.0
- MLNX_OFED 3.4
- OpenSeq2Seq v0.2 at commit a4f627e
- TensorRT 4.0.1

Driver Requirements

Release 18.06 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.06 is based on TensorFlow 1.8.0.
- Updated scripts and README in `nvidia-examples/cnn/` to use cleaner implementation with high-level TensorFlow APIs including Datasets, Layers, and Estimators. Multi-GPU support in these scripts is now provided exclusively using Horovod/MPI.
- Fixed incorrect network definition in `resnet18` and `resnet34` models in `nvidia-examples/cnn/`.
- Updated scripts and README in `nvidia-examples/build_imagenet_data/` to improve usability and ensure that the dataset is correctly downloaded and resized.
- Added support for TensorRT 4 features to TensorFlow-TensorRT integration.
- Includes integration with TensorRT 4.0.1
- Optimized CPU bilinear image resize kernel to improve performance of input pipeline.
- Ubuntu 16.04 with May 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

Key Features And Enhancements

- Added TensorRT 4.0 API support with extended layer support. This support includes the FullyConnected layer and `BatchedMatMul` op.
- Resource management added, where memory allocation is uniformly managed by TensorFlow.
- Bug fixes and better error handling in conversion.

Limitations

- TensorRT conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.
- Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.
- Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.

Known Issues

- Input tensors are required to have rank 4 for quantization mode (INT8 precision).

Announcements

Starting with the next major version of CUDA release, we will no longer provide updated Python 2 containers and will only update Python 3 containers.

Known Issues

There are no known issues in this release.

Chapter 84. TensorFlow Release 18.05

The NVIDIA container image of TensorFlow, release 18.05, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `18.05-py2` contains Python 2.7; `18.05-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.2
- NCCL 2.1.15 (optimized for NVLink™)
- Horovod™ 0.12.1
- OpenMPI™ 3.0.0
- TensorBoard 1.7.0
- MLNX_OFED 3.4
- OpenSeq2Seq v0.2

Driver Requirements

Release 18.05 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.05 is based on TensorFlow 1.7.0.
- For developers needing more visibility between network layer calls and CUDA kernel calls, we've added support for basic NVTX ranges to the TensorFlow executor. Nsight Systems or the NVIDIA Visual Profiler, with NVTX ranges, are able to display each TensorFlow op demarcated by an NVTX range named by the op. NVTX ranges are enabled by default but can be disabled by setting the environment variable `TF_DISABLE_NVTX_RANGES=1`.
- Optimized input pipeline in `nvcnn.py` and `nvcnn_hvd.py` by casting back to uint8 immediately after image preprocessing.
- Added OpenSeq2Seq v0.2 to the base container.
- Includes integration with TensorRT 3.0.4
- Ubuntu 16.04 with April 2018 updates

Accelerating Inference In TensorFlow With TensorRT (TF-TRT)

For step-by-step instructions on how to use TF-TRT, see Accelerating Inference In TensorFlow With TensorRT User Guide.

> **ATTENTION:**
>
> Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to the platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.
>
> For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Key Features And Enhancements

- TensorRT backend accelerates inference performance for frozen TensorFlow models.
- Automatic segmenter that recognizes TensorRT compatible subgraphs and converts them into TensorRT engines. TensorRT engines are wrapped with TensorFlow custom ops that moves the execution of the subgraph to TensorRT backend for optimized performance, while fall back to TensorFlow for non-TensorRT compatible ops.
- Supported networks are slim classification networks including ResNet, VGG, and Inception.
- Mixed precision and quantization are supported.

Limitations

- Conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.
- Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.
- Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.
- Resource management is not integrated, therefore, ensure you limit the memory claimed by TensorFlow in order for TensorRT to acquire the necessary resource. To limit the memory, use `setting per_process_gpu_memory_fraction to < 1.0` and pass it to session creation, for example:

```
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333) sess =
 tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
```

Deprecated Features

In the 18.05 container, you need to create a TensorFlow session with the `per_process_gpu_memory_fraction` option. With the resource management fully integrated, you no longer need to reserve GPU memory from TensorFlow. Therefore, the option is not necessary for mixed TensorFlow-TensorRT (TF-TRT) model.

Known Issues

The TensorRT engine only accepts input tensor with `rank == 4`.

Announcements

Starting with the next major version of CUDA release, we will no longer provide Python 2 containers and will only maintain Python 3 containers.

Known Issues

There are no known issues in this release.

Chapter 85. TensorFlow Release 18.04

The NVIDIA container image of TensorFlow, release 18.04, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `18.04-py2` contains Python 2.7; `18.04-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.1
- NCCL 2.1.15 (optimized for NVLink™)
- Horovod™ 0.11.3
- OpenMPI™ 3.0.0
- TensorBoard 0.4.0-rc1
- MLNX_OFED 3.4

Driver Requirements

Release 18.04 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.04 is based on TensorFlow 1.7.0.

- Added the Mellanox user-space InfiniBand driver to the container.
- Latest version of MLNX_OFED 3.4
- Added support for TensorRT integration in TensorFlow. For functionality details, see TensorRT Integration Speeds Up TensorFlow Inference and the example in the `nvidia-examples/tftrt` directory.
- Improved `nvidia_examples/nvcnn.py` and `nvcnn_hvd.py` to ensure ResNet-50 model converges correctly out of the box. See *Changelog* at the top of `nvidia_examples/nvcnn.py` for more details.
- Enabled Tensor Op math for cuDNN-based RNNs in FP16 precision. This is enabled by default, but can be disabled by setting the environment variable `TF_DISABLE_CUDNN_RNN_TENSOR_OP_MATH=1`.
- Includes integration with TensorRT 3.0.4
- Latest version of NCCL 2.1.15
- Ubuntu 16.04 with March 2018 updates

Announcements

Starting with the next major version of CUDA release, we will no longer provide Python 2 containers and will only maintain Python 3 containers.

Known Issues

There is a degraded performance for graph construction time of grouped convolutions. For more information, see Support for depthwise convolution by groups.

Chapter 86. TensorFlow Release 18.03

The NVIDIA container image of TensorFlow, release 18.03, is available.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  Note: Container image `18.03-py2` contains Python 2.7; `18.03-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 (see Errata section and 2.1) including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.333 (see section 2.3.1)
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.1.1
- NCCL 2.1.2 (optimized for NVLink™)
- Horovod™ 0.11.3
- OpenMPI™ 3.0.0
- TensorBoard 0.4.0-rc1

Driver Requirements

Release 18.03 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- TensorFlow container image version 18.03 is based on TensorFlow 1.4.0.
- Latest updates to OpenSeq2Seq module

- Latest version of cuBLAS 9.0.333
- Latest version of cuDNN 7.1.1
- Latest version of OpenMPI 3.0.0
- Latest version of Horovod 0.11.3
- Latest version of TensorBoard 0.4.0-rc1
- Ubuntu 16.04 with February 2018 updates

Announcements

Starting with the next major version of CUDA release, we will no longer provide Python 2 containers and will only maintain Python 3 containers.

Known Issues

There are no known issues in this release.

Chapter 87. TensorFlow Release 18.02

The NVIDIA container image of TensorFlow, release 18.02, is available.

TensorFlow container image version 18.02 is based on TensorFlow 1.4.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `18.02-py2` contains Python 2.7; `18.02-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 including:
  - CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.282 Patch 2 which is installed by default
  - cuBLAS 9.0.234 Patch 1 as a debian file. Installing Patch 1 by issuing the `dpkg -i /opt/cuda-cublas-9-0_9.0.234-1_amd64.deb` command is the workaround for the known issue described below.
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NVIDIA® Collective Communications Library ™ (NCCL) 2.1.2 (optimized for NVLink™)
- Horovod™ 0.11.2

Driver Requirements

Release 18.02 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Updated OpenSeq2Seq example to include latest bug fixes
- Latest version of cuBLAS
- Ubuntu 16.04 with January 2018 updates

Known Issues

- cuBLAS 9.0.282 regresses RNN seq2seq FP16 performance for a small subset of input sizes. This issue should be fixed in the next update. As a workaround, install cuBLAS 9.0.234 Patch 1 by issuing the `dpkg -i /opt/cuda-cublas-9-0_9.0.234-1_amd64.deb` command.
- The `broadcast` and `reduce` (but not `all_reduce`) functions in the `tf.contrib.nccl` module cause an error when executed as part of a graph. This issue should be fixed in the next update. The multi-GPU training example script `nvidia-examples/cnn/nvcnn.py` includes a workaround for the `nccl.broadcast` function so that the script still runs correctly.

  > **Note:** The Horovod example script `nvidia-examples/cnn/nvcnn_hvd.py` is not affected by this issue.

- Some Python 3 codes may encounter errors when handling text strings containing non-Latin characters. This can be fixed by setting an environment variable with the following command:

  ```
  $ export LC_ALL=C.UTF-8
  ```

  This issue should be fixed in the next update.

Chapter 88. TensorFlow Release 18.01

The NVIDIA container image of TensorFlow, release 18.01, is available.

TensorFlow container image version 18.01 is based on TensorFlow 1.4.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed as a system Python module.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04

  > **Note:** Container image `18.01-py2` contains Python 2.7; `18.01-py3` contains Python 3.5.

- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.282
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NVIDIA® Collective Communications Library ™ (NCCL) 2.1.2 (optimized for NVLink™)
- Horovod™ 0.11.2

Driver Requirements

Release 18.01 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Addition of Python 3 package
- Horovod is now pre-installed in the container
- Updated OpenSeq2Seq example to include latest bug fixes

- Latest version of cuBLAS
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with December 2017 updates

Known Issues

cuBLAS 9.0.282 regresses RNN seq2seq FP16 performance for a small subset of input sizes. As a workaround, revert back to the 11.12 container.

Chapter 89. TensorFlow Release 17.12

The NVIDIA container image of TensorFlow, release 17.12, is available.

TensorFlow container image version 17.12 is based on TensorFlow 1.4.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.234
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.5
- NVIDIA® Collective Communications Library ™ (NCCL) 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 17.12 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with November 2017 updates

Known Issues

A corner case of float16 reductions is known to give the wrong result of Maxwell and earlier architectures. This will be fixed in a future release.

Chapter 90. TensorFlow Release 17.11

The NVIDIA container image of TensorFlow, release 17.11, is available.

TensorFlow container image version 17.11 is based on TensorFlow 1.3.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA 9.0.176 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 9.0.234
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.4
- NVIDIA® Collective Communications Library ™ (NCCL) 2.1.2 (optimized for NVLink™)

Driver Requirements

Release 17.11 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Added support for float16 data type and Tensor Core math in batched matrix multiply operations.
- Added example script `nvidia-examples/cnn/nvcnn_hvd.py`, which demonstrates use of the Horovod library for multi-node training.
- Added `Dockerfile.horovod` demonstrating how to build a Docker container with the Horovod library and MPI support.

- Added OpenSeq2Seq example demonstrating sequence-to-sequence model training in `nvidia-examples/OpenSeq2Seq/`.
- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with October 2017 updates

Known Issues

There are no known issues in this release.

Chapter 91. TensorFlow Release 17.10

The NVIDIA container image of TensorFlow, release 17.10, is available.

TensorFlow container image version 17.10 is based on TensorFlow 1.3.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.3
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.5 (optimized for NVLink™)

Driver Requirements

Release 17.10 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Added PNG image support to `nvcnn.py`.
- Fixed issue with `batchnorm` op that broke backwards compatibility in the previous release.
- Renamed the `TF_ENABLE_TENSOR_OP_MATH (default=1)` environment variable to `TF_DISABLE_TENSOR_OP_MATH (default=0)`.
- Upgraded Bazel to version 0.5.4.
- Worked around hash mismatches in third-party source downloads.

- Enabled compilation flags `-march=sandybridge -mtune=broadwell`.
- Updated Eigen to the top of the tree and removed custom patches.
- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with September 2017 updates

Known Issues

There are no known issues in this release.

Chapter 92. TensorFlow Release 17.09

The NVIDIA container image of TensorFlow, release 17.09, is available.

TensorFlow container image version 17.09 is based on TensorFlow 1.3.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.2
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.5 (optimized for NVLink™)

Driver Requirements

Release 17.09 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Tensor Core operation support in TensorFlow is enabled by default on Volta for FP16 convolutions and matrix multiplies, which should give a speedup for FP16 models.
- Added experimental support for:
  - FP16 training in `nvidia-examples/cnn/nvcnn.py`
  - FP16 input/output in the fused batch normalization operation (`tf.nn.fused_batch_norm`)
  - Tensor Core operation in FP16 convolutions and matrix multiplications

        - Added the TF_ENABLE_TENSOR_OP_MATH parameter which enables and disables Tensor Core operation (defaults to enabled).
    - Tensor Core operation in FP32 matrix multiplications
        - Added the TF_ENABLE_TENSOR_OP_MATH_FP32 parameter which enables and disables Tensor Core operation for float32 matrix multiplications (defaults to disabled because it reduces precision).
- Increased the TF_AUTOTUNE_THRESHOLD parameter which improves auto-tune stability.
- Increased the CUDA_DEVICE_MAX_CONNECTIONS parameter which solves performance issues related to streams on Tesla K80 GPUs.
- Enhancements to `nvidia-examples/cnn/nvcnn.py`
    - Fixed a bug where the final layer was wrong when running in evaluation mode.
    - Changed `is_training` to a constant instead of a placeholder for better performance and reduced memory use.
    - Merged gradients for all layers into a single NCCL call for better performance.
    - Disabled use of XLA by default for better performance.
    - Disabled `zero_debias_moving_mean` in batch normalization operation.
- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with August 2017 updates

Known Issues

There are no known issues in this release.

Chapter 93. TensorFlow Release 17.07

The NVIDIA container image of TensorFlow, release 17.07, is available.

TensorFlow container image version 17.07 is based on TensorFlow 1.2.1.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61.2 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) Patch 2
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 2.0.3 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Small bug-fixes in evaluation mode of `nvidia-examples/cnn/nvcnn.py`
- Ubuntu 16.04 with June 2017 updates

Known Issues

There are no known issues in this release.

Chapter 94. TensorFlow Release 17.06

The NVIDIA container image of TensorFlow, release 17.06, is available.

TensorFlow container image version 17.06 is based on TensorFlow 1.1.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Ubuntu 16.04 with May 2017 updates

Known Issues

The `inception_v4` model, with a batch size of 64 per GPU, and with large input images or resolution (for example, 480 pixels on the shortest side), are seen to run out of memory. To work around this in TensorFlow 17.06, reduce the resolution or reduce the batch size to allow the model to fit.

Chapter 95. TensorFlow Release 17.05

The NVIDIA container image of TensorFlow, release 17.05, is available.

TensorFlow container image version 17.05 is based on TensorFlow 1.0.1.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Latest cuDNN release
- Ubuntu 16.04 with April 2017 updates

Known Issues

The `inception_v4` model, with a batch size of 64 per GPU, and with large input images or resolution (for example, 480 pixels on the shortest side), are seen to run out of memory. To work around this in TensorFlow 17.05, reduce the resolution or reduce the batch size to allow the model to fit.

Chapter 96. TensorFlow Release 17.04

The NVIDIA container image of TensorFlow, release 17.04, is available.

TensorFlow container image version 17.04 is based on TensorFlow 1.0.1.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

To achieve optimum TensorFlow performance, for image based training, the container includes a sample script that demonstrates efficient training of convolutional neural networks (CNNs). The sample script may need to be modified to fit your application.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.20
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- 2x improvement on 8GPUs; 1.5X on 4 GPUs
- Updated NCCL integration with support for NVLink
- Multi-GPU CNN examples that demonstrates efficient training of CNNs using NCCL
- XLA (Accelerated Linear Algebra) support enabled, allowing users to offload operations to TensorFlow experimental XLA back-end
- Ubuntu 16.04 with March 2017 updates

Known Issues

There are no known issues in this release.

Chapter 97. TensorFlow Release 17.03

The NVIDIA container image of TensorFlow, release 17.03, is available.

TensorFlow container image version 17.03 is based on TensorFlow 1.0.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.20
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Multi-GPU BigLSTM example that trains a recurrent neural network (RNN) to learn a language model
- Ubuntu 16.04 with February 2017 updates

Known Issues

There are no known issues in this release.

Chapter 98. TensorFlow Release 17.02

The NVIDIA container image of TensorFlow, release 17.02, is available.

TensorFlow container image version 17.02 is based on TensorFlow 0.12.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 14.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.13
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Fused image color adjustment kernels for improved preprocessing performance
- Ubuntu 14.04 with January 2017 updates

Known Issues

There are no known issues in this release.

Chapter 99. TensorFlow Release 17.01

The NVIDIA container image of TensorFlow, release 17.01, is available.

TensorFlow container image version 17.01 is based on TensorFlow 0.12.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 14.04
- NVIDIA CUDA® 8.0.54
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.10
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Ubuntu 14.04 with December 2016 updates

Known Issues

There are no known issues in this release.

Chapter 100. TensorFlow Release 16.12

The NVIDIA container image of TensorFlow, release 16.12, is available.

TensorFlow container image version 16.12 is based on TensorFlow 0.12.0.

Contents of TensorFlow

This container image contains the complete source of the version of NVIDIA TensorFlow in `/opt/tensorflow`. It is pre-built and installed into the `/usr/local/[bin,lib]` directories in the container image.

The container also includes the following:

- Ubuntu 14.04
- NVIDIA CUDA® 8.0.54
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.5
- NVIDIA® Collective Communications Library ™ (NCCL) 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This TensorFlow release includes the following key features and enhancements.

- Supports multi-GPU training
  - [BETA] NCCL integration for improved multi-GPU scaling

    Note: Requires explicit use by the model script.
- Supports recurrent neural networks
  - Support for cuDNN recurrent neural networks (RNN) layers

    Note: Requires explicit use by the model script.
- Better I/O throughput via `libjpeg-turbo`, fast iDCT decoding
- Support for the non-fused Winograd algorithm for improved convolution performance.

- TensorBoard; a data visualization toolkit
- Several built-in TensorFlow examples
- Ubuntu 14.04 with November 2016 updates

Known Issues

There are no known issues in this release.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

HDMI

OpenCL

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, DALI, DGX, DGX-1, DGX-2, DGX Station, DLProf, Jetson, Kepler, Maxwell, NCCL, Nsight Compute, Nsight Systems, NvCaffe, PerfWorks, Pascal, SDK Manager, Tegra, TensorRT, Triton Inference Server, Tesla, TF-TRT, and Volta are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
https://www.nvidia.com