NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet

Release Notes

RN-08476-001_v24.11 | July 2025

Table of Contents

Chapter 1. NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet Overview

The NVIDIA® Deep Learning SDK accelerates widely used deep learning frameworks such as Apache MXNet.

Apache MXNet is a deep learning framework that is designed for efficiency and flexibility. It allows you to mix the flavors of symbolic programming and imperative programming to maximize efficiency and productivity. At its core is a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. A graph optimization layer on top of the scheduler makes symbolic execution fast and memory efficient. The library is portable and lightweight, and it scales to multiple NVIDIA GPUs and multiple machines.

More than a deep learning project, Apache MXNet is a collection of blueprints and guidelines that are used to build deep learning systems and provide interesting insights about deep learning systems for hackers.

In the container, go to the `/workspace/README.md` directory for information about customizing your Optimized Deep Learning Framework image. For more information about Apache MXNet, including tutorials, documentation, and examples, see:

- Apache MXNet website
- Apache MXNet project

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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet

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 which runs the software.
- You define the runtime resources of the container by including 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.

   ```
   docker pull nvcr.io/nvidia/mxnet:24.01-py3
   ```

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

   Ensure that the pull process successfully completes before you proceed 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/
     mxnet:<xx.xx>-py3
     ```

   - 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/mxnet:<xx.xx>-
     py3
     ```

   To run the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, import it as a Python module:

   ```
   $ python
    Python 3.5.2 (default, Nov 23 2017, 16:37:01)
    [GCC 5.4.0 20160609] on linux
    Type "help", "copyright", "credits" or "license" for more information.
    >>> import mxnet as mx
    >>> a = mx.nd.ones((2,3), mx.gpu())
    >>> print((a*2).asnumpy())
    [[ 2.  2.  2.]
     [ 2.  2.  2.]]
   ```

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

> **Note:** To share data between ranks, NVIDIA Collective Communications Library (NCCL) might require shared system memory for IPC and pinned (page-locked) system memory resources, so you might need to increase your operating system's limits on these resources. Refer to your system's documentation for more information.
>
> In particular, Docker containers default to limited shared and pinned memory resources. When using NCCL in a container, we recommend 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
> ```

Chapter 4. Release 24.11

The Apache MXNet container has been deprecated in DLFW releases.

Chapter 5. Release 24.10

The Apache MXNet container has been deprecated in DLFW releases.

Chapter 6. Release 24.09

The Apache MXNet container has been deprecated in DLFW releases.

Chapter 7. Release 24.08

The Apache MXNet container has been deprecated in DLFW releases.

Chapter 8. Release 24.07

There is no Apache MXNet container in DLFW release 24.07.

Chapter 9. Release 24.06

This NVIDIA Optimized Deep Learning Framework container release is intended for use on the NVIDIA Hopper Architecture H100 GPU and on previous generation GPUs like A100, V100 and T4 etc, and with the latest NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.5.0.23
- NVIDIA cuBLAS 12.5.2.13
- cuDNN 9.1.0.70
- NCCL 2.21.5
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.19
- Horovod 0.28
- NVIDIA TensorRT™ 10.1.0.27
- NVIDIA DALI® 1.38
- nvImageCodec 0.2.0.7
- Nsight Compute 2024.2.0.16
- Nsight Systems 2024.2.3.38
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1

- JupyterLab 2.2.10, with updated mistune 2.0.4

Driver Requirements

Release 24.06 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.

GPU Requirements

Release 24.06 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.06 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- After the 24.06 release, Apache MXNet will no longer be released in NVIDIA Optimized Deep Learning Framework containers.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.06	22.04	NVIDIA CUDA 12.5.0.23		TensorRT 10.1.0.27
24.05		NVIDIA CUDA 12.4.1	1.9.1	TensorRT 10.0.1.6
24.04				TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01				TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- Users that run the pytest unittests found under `/opt/mxnet/tests` will notice some failing tests because the public download site `data.mxnet.io` for pre-trained models has moved.

Chapter 10. Release 24.05

This NVIDIA Optimized Deep Learning Framework container release is intended for use on the NVIDIA Hopper Architecture H100 GPU and on previous generation GPUs like A100, V100 and T4 etc, and with the latest NVIDIA CUDA® 12 and NVIDIA cuDNN 9 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.4.1
- NVIDIA cuBLAS 12.4.5.8
- cuDNN 9.1.0.70
- NCCL 2.21.5
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.19
- Horovod 0.28
- NVIDIA TensorRT™ 10.0.1.6
- NVIDIA DALI® 1.37.1
- nvImageCodec 0.2.0.7
- Nsight Compute 2024.1.1.4
- Nsight Systems 2024.2.1.106
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1

- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 24.05 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.05 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- After the 24.06 release, Apache MXNet will no longer be released in NVIDIA Optimized Deep Learning Framework containers.
- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.05	22.04	NVIDIA CUDA 12.4.1	1.9.1	TensorRT 10.0.1.6
24.04				TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41		
24.02		NVIDIA CUDA 12.3.2		
24.01				TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the [Training With Mixed Precision Guide].

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- Users that run the pytest unittests found under `/opt/mxnet/tests` will notice some failing tests because the public download site `data.mxnet.io` for pre-trained models has moved.

Chapter 11. Release 24.04

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.4.1
- NVIDIA cuBLAS 12.4.5.68
- cuDNN 9.1.0.70
- NCCL 2.21.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.18
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.3
- NVIDIA DALI® 1.36
- nvImageCodec 0.2.0.7
- Nsight Compute 2024.1.1.4
- Nsight Systems 2024.2.1.106

- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 24.04 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.04 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.04	22.04	NVIDIA CUDA 12.4.1	1.9.1	TensorRT 8.6.3
24.03		NVIDIA CUDA 12.4.0.41	1.9.1	
24.02		NVIDIA CUDA 12.3.2		
24.01				TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
22.08		NVIDIA CUDA 11.7.1		TensorRT 8.4.2.4

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- Users that run the pytest unittests found under `/opt/mxnet/tests` will notice some failing tests because the public download site `data.mxnet.io` for pre-trained models has moved.

Chapter 12. Release 24.03

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.4.0.41
- NVIDIA cuBLAS 12.4.2.65
- cuDNN 9.0.0.306
- NCCL 2.20
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.18
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.3
- NVIDIA DALI® 1.35
- Nsight Compute 2024.1.0.13
- Nsight Systems 2024.2.1.38
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 24.03 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.03 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.03	22.04	NVIDIA CUDA 12.4.0.41	1.9.1	TensorRT 8.6.3
24.02		NVIDIA CUDA 12.3.2		
24.01				TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	Apache MXNet	TensorRT
22.05		NVIDIA CUDA 11.7.0	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from	
19.12				TensorRT 6.0.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.11			September 4, 2019	
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/`

`AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- Users that run the pytest unittests found under `/opt/mxnet/tests` will notice some failing tests because the public download site `data.mxnet.io` for pre-trained models has moved.

Chapter 13. Release 24.02

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuDNN 9.0.0.306
- NCCL 2.19.4
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16rc4
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.3
- NVIDIA DALI® 1.34
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1.97
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 24.02 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.02 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.02	22.04	NVIDIA CUDA 12.3.2	1.9.1	TensorRT 8.6.3
24.01			1.9.1	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- Users that run the pytest unittests found under `/opt/mxnet/tests` will notice some failing tests because the public download site `data.mxnet.io` for pre-trained models has moved.

Chapter 14. Release 24.01

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuDNN 8.9.7.29
- NCCL 2.19.4
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16rc4
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.33
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1.97
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 24.01 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 24.01 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
24.01	22.04	NVIDIA CUDA 12.3.2	1.9.1	TensorRT 8.6.1.6
23.12				
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 15. Release 23.12

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.3.2
- NVIDIA cuBLAS 12.3.4.1
- cuDNN 8.9.7.29
- NCCL 2.19.3
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.32
- Nsight Compute 2023.3.1.1
- Nsight Systems 2023.4.1
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10, with updated mistune 2.0.4

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.

GPU Requirements

Release 23.12 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.12 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.12	22.04	NVIDIA CUDA 12.3.2	1.9.1	TensorRT 8.6.1.6
23.11		NVIDIA CUDA 12.3.0		
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 16. Release 23.11

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.3.0
- NVIDIA cuBLAS 12.3.2.1
- cuDNN 8.9.6
- NCCL 2.19.3
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.31.0
- Nsight Compute 2023.3.0.12
- Nsight Systems 2023.3.1.92
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10

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.

GPU Requirements

Release 23.11 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.11 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.11	22.04	NVIDIA CUDA 12.3.0	1.9.1	TensorRT 8.6.1.6
23.10		NVIDIA CUDA 12.2.2		
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 17. Release 23.10

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.2.2
- NVIDIA cuBLAS 12.2.5.6
- cuDNN 8.9.5
- NCCL 2.19.3
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.30.0
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.3.1.92
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10

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.

GPU Requirements

Release 23.10 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.10 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.10	22.04	NVIDIA CUDA 12.2.2	1.9.1	TensorRT 8.6.1.6
23.09		NVIDIA CUDA 12.2.1		
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 18. Release 23.09

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.2.1
- NVIDIA cuBLAS 12.2.5.6
- cuDNN 8.9.5
- NCCL 2.18.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.16
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.29.0
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.3.1.92
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10

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.

GPU Requirements

Release 23.09 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.08 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.09	22.04	NVIDIA CUDA 12.2.1	1.9.1	TensorRT 8.6.1.6
23.08				
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 19. Release 23.08

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.2.1
- NVIDIA cuBLAS 12.2.5
- cuDNN 8.9.4
- NCCL 2.18.3
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.15
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.28.0
- Nsight Compute 2023.2.1.3
- Nsight Systems 2023.2.3.1001
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10

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.

GPU Requirements

Release 23.08 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.08 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.08	22.04	NVIDIA CUDA 12.2.1	1.9.1	TensorRT 8.6.1.6
23.07		NVIDIA CUDA 12.1.1		
23.06				
23.05				TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 20. Release 23.07

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuDNN 8.9.3
- NCCL 2.18.3
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.15
- Horovod 0.28
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.270
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.2.3.1001
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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.

GPU Requirements

Release 23.07 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.06 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.07	22.04	NVIDIA CUDA 12.1.1	1.9.1	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				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 21. Release 23.06

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuDNN 8.9.2
- NCCL 2.18.1
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 39.0
- OpenMPI 4.1.4+
- OpenUCX 1.15.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.15
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.6.1.6
- NVIDIA DALI® 1.26.0
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.2.3.1001
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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.

GPU Requirements

Release 23.06 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.06 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers are no longer tested on Pascal GPU architectures.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.06	22.04	NVIDIA CUDA 12.1.1	1.9.1	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				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 22. Release 23.05

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 22.04 including Python 3.10
- NVIDIA CUDA 12.1.1
- NVIDIA cuBLAS 12.1.3.1
- cuDNN 8.9.1.23
- NCCL 2.18.1
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.14
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.6.1.2
- NVIDIA DALI® 1.25.0
- Nsight Compute 2023.1.1.4
- Nsight Systems 2023.2
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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.

GPU Requirements

Release 23.05 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.04 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 23.06 release, the NVIDIA Optimized Deep Learning Framework containers will no longer be tested on Pascal GPU architectures.
- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.05	22.04	NVIDIA CUDA 12.1.1	1.9.1	TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 23. Release 23.04

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 12.1.0
- NVIDIA cuBLAS 12.1.3
- cuDNN 8.9.0
- NCCL 2.17.1
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.6.1
- NVIDIA DALI® 1.23.0
- Nsight Compute 2023.1.0.15
- Nsight Systems 2023.1.1.127
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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.

GPU Requirements

Release 23.04 supports CUDA compute capability 6.0 and later. This corresponds to GPUs in the NVIDIA Pascal™, NVIDIA Volta™, NVIDIA Turing™, NVIDIA Ampere architecture, NVIDIA Hopper™, and NVIDIA Ada Lovelace architecture 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.04 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.04	20.04	NVIDIA CUDA 12.1.0	1.9.1	TensorRT 8.6.1
23.03				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		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 24. Release 23.03

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 12.1.0
- NVIDIA cuBLAS from CUDA 12.1.0
- cuDNN 8.8.1.3
- NCCL 2.17.1
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.5.3
- NVIDIA DALI® 1.23.0
- Nsight Compute 2023.1.0.15
- Nsight Systems 2023.1.1.127
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.03 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.03	20.04	NVIDIA CUDA 12.1.0		TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1	1.9.1	
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest

convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 25. Release 23.02

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 12.0.1
- NVIDIA cuBLAS from CUDA 12.0.1
- cuDNN 8.7.0
- NCCL 2.16.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.5.3
- NVIDIA DALI® 1.22.0
- Nsight Compute 2022.4.1.6
- Nsight Systems 2022.5.1.93
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.02 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.02	20.04	NVIDIA CUDA 12.0.1	1.9.1	TensorRT 8.5.3
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0		TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 26. Release 23.01

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 12.0.1
- NVIDIA cuBLAS from CUDA 12.0.1
- cuDNN 8.7.0
- NCCL 2.16.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.5.2.2
- NVIDIA DALI® 1.21.0
- Nsight Compute 2022.4.1.6
- Nsight Systems 2022.5.1
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 23.01 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
23.01	20.04	NVIDIA CUDA 12.0.1		TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	1.9.1	TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 27. Release 22.12

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuDNN 8.7.0
- NCCL 2.15.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.13
- Horovod 0.26.1
- NVIDIA TensorRT™ 8.5.1
- NVIDIA DALI® 1.20.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.12 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.12	20.04	NVIDIA CUDA 11.8.0	1.9.1	TensorRT 8.5.1
22.11				
22.10				TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.

Chapter 28. Release 22.11

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuDNN 8.7.0
- NCCL 2.15.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.4+
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.2tp1
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.5.1
- NVIDIA DALI® 1.18.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.10 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>Apache MXNet</th><th>TensorRT</th></tr>
<tr><td>22.11</td><td rowspan="17">20.04</td><td rowspan="3">NVIDIA CUDA 11.8.0</td><td rowspan="6">1.9.1</td><td>TensorRT 8.5.1</td></tr>
<tr><td>22.10</td><td rowspan="2">TensorRT 8.5 EA</td></tr>
<tr><td>22.09</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="5">1.9.0.rc6</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>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">TensorRT 8.0.1</td></tr>
<tr><td>21.07</td><td>NVIDIA CUDA 11.4.0</td><td>1.9.0.rc3</td></tr>
<tr><td>21.06</td><td>NVIDIA CUDA 11.3.1</td><td>1.9.0.rc2</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">1.8.0</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>1.8.0.rc2</td><td>7.2.2.3+cuda11.1.0.024</td></tr>
</table>

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- This container has by default the environment variable setting `MXNET_CUDNN_AUTOTUNE_DEFAULT=2`, which enables the selection of the fastest convolution op implementation from a list of cuDNN-supplied candidates at runtime. This setting may lead in rare model training scripts to long start-up times or hangs, in which case the recommended remedy is to turn off auto-tuning via `MXNET_CUDNN_AUTOTUNING_DEFAULT=0`.
- This container includes an experimental version of the group BatchNorm operator that uses the nvshmem library for inter-GPU communication when enabled in Python scripts via `mx.cuda_utils.nvshmem_init(mpi4py.COMM_WORLD)`. Users should be aware that this feature uses just-in-time link-time-optimization (JIT LTO), a preview capability of the CUDA 11 drivers R470 through R520 that will be migrated to the runtime as of the CUDA 12 drivers R525 or later. As a consequence, the nvshmem group BatchNorm function of this container requires a R470 through R520 bare-metal driver to function, though this restriction should be lifted in a future release.

Chapter 29. Release 22.10

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuDNN 8.6.0.163
- NCCL 2.15.5
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.5a1
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.2tp1
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.5.0.12
- NVIDIA DALI® 1.18.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.4.2.1
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.10 is based on 1.9.1.

  Apache MXNet, formerly an effort undergoing incubation at The Apache Software Foundation (ASF), has graduated to become a full-status active project of the ASF.

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.10	20.04	NVIDIA CUDA 11.8.0	1.9.1	TensorRT 8.5 EA
22.09				
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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 30. Release 22.09

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.8.0
- NVIDIA cuBLAS 11.11.3.6
- cuDNN 8.6.0.163
- NCCL 2.15.1 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.2rc4
- OpenUCX 1.14.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.12.1a0
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.5.0.12
- NVIDIA DALI® 1.17.0
- Nsight Compute 2022.3.0.22
- Nsight Systems 2022.3.1.43
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- JupyterLab 2.2.10 with updated mistune 2.0.4

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.09 is based on 1.9.1.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been

removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.09	20.04	NVIDIA CUDA 11.8.0	1.9.1	TensorRT 8.5 EA
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	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>Apache MXNet</th><th>TensorRT</th></tr>
<tr><td>20.11</td><td rowspan="14">18.04</td><td rowspan="2">NVIDIA CUDA 11.1.0</td><td></td><td rowspan="2">TensorRT 7.2.1</td></tr>
<tr><td>20.10</td><td rowspan="2">1.7.0</td></tr>
<tr><td>20.09</td><td rowspan="2">NVIDIA CUDA 11.0.3</td><td rowspan="3">TensorRT 7.1.3</td></tr>
<tr><td>20.08</td><td rowspan="4">1.6.0</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>TensorRT 7.1.2</td></tr>
<tr><td>20.03</td><td rowspan="4">NVIDIA CUDA 10.2.89</td><td rowspan="3">TensorRT 7.0.0</td></tr>
<tr><td>20.02</td><td>1.6.0.rc2</td></tr>
<tr><td>20.01</td><td rowspan="3">1.5.1 commit c98184806 from September 4, 2019</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></tr>
<tr><td>19.09</td><td>1.5.0 commit 006486af3 from August 28, 2019</td></tr>
<tr><td>19.08</td><td>1.5.0 commit 75a9e187d from June 27, 2019</td><td>TensorRT 5.1.5</td></tr>
</table>

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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 31. Release 22.08

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.7.1
- NVIDIA cuBLAS 11.10.3.66
- cuDNN 8.5.0.96
- NCCL 2.12.12 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.2rc4
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.4.2.4
- NVIDIA DALI® 1.16.0
- Nsight Compute 2022.2.1.0
- Nsight Systems 2022.1.3.18
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.08 is based on 1.9.1.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Starting with the 22.07 release, the NVIDIA Optimized Deep Learning Framework containers are available for the Arm SBSA platform. For example, pulling the Docker image nvcr.io/nvidia/mxnet:22.07-py3 on an Arm SBSA machine will automatically fetch the Arm-specific image.

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

  The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>Apache MXNet</th><th>TensorRT</th></tr>
<tr><td>22.08</td><td rowspan="13">20.04</td><td>NVIDIA CUDA 11.7.1</td><td rowspan="3">1.9.1</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="5">1.9.0.rc6</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>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">TensorRT 8.0.1.6</td></tr>
<tr><td>21.07</td><td>NVIDIA CUDA 11.4.0</td><td>1.9.0.rc3</td></tr>
<tr><td>21.06</td><td>NVIDIA CUDA 11.3.1</td><td>1.9.0.rc2</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">1.8.0</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	Apache MXNet	TensorRT
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Chapter 32. Release 22.07

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.7. Update 1 Preview
- NVIDIA cuBLAS 11.10.3.66
- cuDNN 8.4.1
- NCCL 2.12.12 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.2rc4
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.4.1
- NVIDIA DALI® 1.15.0
- Nsight Compute 2022.2.1.0
- Nsight Systems 2022.1.3.3
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.07 is based on 1.9.1.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.07	20.04	NVIDIA CUDA 11.7 Update 1 Preview	1.9.1	TensorRT 8.4.1
22.06				TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Chapter 33. Release 22.06

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.1. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.7. Update 1 Preview
- NVIDIA cuBLAS 11.10.3.66
- cuDNN 8.4.1
- NCCL 2.12.12 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.2rc4
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.2.5
- NVIDIA DALI® 1.14.0
- Nsight Compute 2022.1.1.2
- Nsight Systems 2022.1.3.3
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.06 is based on 1.9.1.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.06	20.04	NVIDIA CUDA 11.7 Update 1 Preview	1.9.1	TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	1.9.0.rc6	
22.04		NVIDIA CUDA 11.6.2		TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1		TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2		TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Chapter 34. Release 22.05

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc6. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.7.0
- NVIDIA cuBLAS 11.10.1.25
- cuDNN 8.4.0.27
- NCCL 2.12.10 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.2rc4
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.2.5
- NVIDIA DALI® 1.13.0
- Nsight Compute 2022.1.1.2
- Nsight Systems 2022.1.3.3
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.05 is based on 1.9.0rc6.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.05	20.04	NVIDIA CUDA 11.7.0		TensorRT 8.2.5
22.04		NVIDIA CUDA 11.6.2	1.9.0rc6	TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1	1.9.0rc6	TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2	1.9.0.rc6	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Fixed Issues

The following issue was fixed in this release:

- Fixed an issue from release 22.03 where segmentation faults are seen at the beginning of model training on A100.

  Users no longer need to set `MXNET_CUDNN_HEUR_MODE=0` in the environment as a workaround.

Chapter 35. Release 22.04

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc6. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.6.2
- cuBLAS 11.9.3.115
- cuDNN 8.4.0.27
- NCCL 2.12.10 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.24.2, with enhancements
- NVIDIA TensorRT™ 8.2.4.2
- NVIDIA DALI® 1.12.0
- Nsight Compute 2022.1.1.2
- Nsight Systems 2022.2.1.31-5fe97ab
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 22.04 is based on NVIDIA CUDA 11.6.2, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, 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 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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.04 is based on 1.9.0rc6.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.04	20.04	NVIDIA CUDA 11.6.2	1.9.0rc6	TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1	1.9.0rc6	TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2	1.9.0.rc6	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the [Training With Mixed Precision Guide].

Fixed Issues

The following issue was fixed in this release:

- Fixed an issue from release 22.03 where segmentation faults are seen at the beginning of model training on A100.

  Users no longer need to set `MXNET_CUDNN_HEUR_MODE=0` in the environment as a workaround.

Chapter 36. Release 22.03

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU and on previous generation GPUs like V100 and T4, and with the latest NVIDIA CUDA® 11 and NVIDIA cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc6. It is prebuilt and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.6.1
- cuBLAS 11.8.1.74
- cuDNN 8.3.3.40
- NCCL 2.12.9 (optimized for NVIDIA NVLink®)
- Amazon Labs Sockeye sequence-to-sequence framework 2.3.14 (for machine translation)
- rdma-core 32.1
- OpenMPI 4.1.1rc1
- OpenUCX 1.12.0
- GDRCopy 2.3
- NVIDIA HPC-X 2.10
- Horovod 0.23.0, with enhancements
- NVIDIA TensorRT™ 8.2.3
- NVIDIA DALI® 1.11.1
- Nsight Compute 2022.1.1.2
- Nsight Systems 2021.5.2.53
- GluonCV Toolkit 0.7

- GluonNLP Toolkit 0.8.1
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.10
  - JupyterLab Server 1.0.6
  - Jupyter-TensorBoard

Driver Requirements

Release 22.03 is based on NVIDIA CUDA 11.6.1, which requires NVIDIA Driver release 470 or later. However, if you are running on Data Center GPUs (formerly Tesla), for example, T4, 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 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. 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 22.03 is based on 1.9.0rc6.

  Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`).

The models can instead be obtained from Github or the NGC. Some python packages, included in previous containers to support these example models, have also been removed. Depending on their specific use cases, you might need to add some packages that were previously preinstalled.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
22.03	20.04	NVIDIA CUDA 11.6.1	1.9.0rc6	TensorRT 8.2.3
21.09		NVIDIA CUDA 11.4.2	1.9.0.rc6	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on Volta and NVIDIA Turing, so 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:

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a higher accuracy.

  This model script is available on GitHub and one NGC.

Automatic Mixed Precision

Training deep learning networks is a computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware and software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision, such as FP16, to use the Tensor Cores that are available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

Automatic Mixed Precision (AMP) automatically applies the FP16 training guidelines by using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

In the container, the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial is located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory, shows you how to get started with mixed precision training using AMP for Apache MXNet, by using the SSD network example from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- Segmentation faults are possible with low probability at the beginning of model training on A100.

  This issue will be fixed in a future release. To eliminate this issue, users are advised to set the `MXNET_CUDNN_HEUR_MODE=0` environment variable before launching their training script.
- During model training, the **'nvJPEG error (10): <unknown error>'** warning message might be displayed.

  This message refers to an issue about parsing image metadata and can be safely ignored.

Chapter 37. Release 21.09

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU, as well as on previous generation GPUs like V100 and T4, and with the latest CUDA 11 and cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc6. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- Horovod 0.22.1, with enhancements
- rdma-core 36.0
- OpenMPI 4.1.1+
- OpenUCX 1.11.0rc1
- GDRCopy 2.3
- NVIDIA HPC-X 2.9
- Nsight Systems 2021.3.1.57
- TensorRT 8.0.3
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- DALI 1.5.0

- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.9
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.09 is based on 1.9.0.rc6. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- As of the previous (21.08) release, the implementation of the Convolution and Deconvolution operators is based on a direct use of the cuDNN v8 API, rather than the cuDNN v7 legacy API. This approach can provide enhanced performance by autotuning over an expanded set of conv/deconv ‘engine configs’ as compared to the restricted set of ‘algo numbers’ in prior releases. Users should note that in this 21.09 release:
  - The time spent in autotuning may be longer than seen in 21.08 and earlier releases.

- The potential issue of the 21.08 release, where a model's GPU global memory use may have increased due to the new autotuning process and selections, has been corrected.

Announcements

- After release 21.09, future releases of the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container will be paused for approximately 6 months.
- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- 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 Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.09	20.04	NVIDIA CUDA 11.4.2	1.9.0.rc6	TensorRT 8.0.3
21.08		NVIDIA CUDA 11.4.1		TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 38. Release 21.08

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU, as well as on previous generation GPUs like V100 and T4, and with the latest CUDA 11 and cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc6. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- Horovod 0.22.1, with enhancements
- 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
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- DALI 1.4.0

- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 2.2.9
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.08 is based on 1.9.0.rc6. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- The implementation of the Convolution and Deconvolution operators is now based on a direct use of the cuDNN v8 API, rather than the cuDNN v7 legacy API. This approach can provide enhanced performance by autotuning over an expanded set of conv/deconv ‘engine configs’ as compared to the restricted set of ‘algo numbers’ in prior releases. Users should note:
  - The following Convolution and Deconvolution operator parameters, generally used for testing purposes, are no longer supported: `cudnn_algo_fwd`, `cudnn_algo_bwd_data`, `cudnn_algo_bwd_filter`, `cudnn_algo_fwd_prec`, `cudnn_algo_bwd_prec`, and `cudnn_algo_verbose`.

- The preferred conv/deconv engine config chosen by the autotuning process can be displayed by setting the environment variable `MXNET_CUDNN_ALGO_VERBOSE_LEVEL=1`, while the full autotuning results are available by setting `MXNET_CUDNN_ALGO_VERBOSE_LEVEL=2`.
- In some cases, compared to previous releases, a model's GPU global memory use may increase due to the new autotuning process and selections. Users are advised to set `MXNET_CUDNN_AUTOTUNE_DEFAULT=0` in such cases.

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.
- 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 Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.08	20.04	NVIDIA CUDA 11.4.1	1.9.0.rc6	TensorRT 8.0.1.6
21.07		NVIDIA CUDA 11.4.0	1.9.0.rc3	
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10	18.04	NVIDIA CUDA 11.1.0	1.7.0	TensorRT 7.2.1
20.09	18.04	NVIDIA CUDA 11.0.3	1.7.0	TensorRT 7.1.3
20.08	18.04	NVIDIA CUDA 11.0.3	1.6.0	TensorRT 7.1.3
20.07	18.04	NVIDIA CUDA 11.0.194	1.6.0	TensorRT 7.1.3
20.06	18.04	NVIDIA CUDA 11.0.167	1.6.0	TensorRT 7.1.2
20.03	18.04	NVIDIA CUDA 10.2.89	1.6.0	TensorRT 7.0.0
20.02	18.04	NVIDIA CUDA 10.2.89	1.6.0.rc2	TensorRT 7.0.0
20.01	18.04	NVIDIA CUDA 10.2.89	1.5.1 commit c98184806 from September 4, 2019	TensorRT 7.0.0
19.12 19.11	18.04	NVIDIA CUDA 10.2.89	1.5.1 commit c98184806 from September 4, 2019	TensorRT 6.0.1
19.10	18.04	NVIDIA CUDA 10.1.243	1.5.1 commit c98184806 from September 4, 2019	TensorRT 6.0.1
19.09	18.04	NVIDIA CUDA 10.1.243	1.5.0 commit 006486af3 from August 28, 2019	TensorRT 6.0.1
19.08	18.04	NVIDIA CUDA 10.1.243	1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 39. Release 21.07

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU, as well as on previous generation GPUs like V100 and T4, and with the latest CUDA 11 and cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc3. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- Horovod 0.22.1, with enhancements
- 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
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1

- DALI 1.3.0
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.6
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.07 is based on 1.9.0.rc3. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.07	20.04	NVIDIA CUDA 11.4.0	1.9.0.rc3	TensorRT 8.0.1.6
21.06		NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05		NVIDIA CUDA 11.3.0	1.8.0	
21.04				
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- 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.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 40. Release 21.06

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU, as well as on previous generation GPUs like V100 and T4, and with the latest CUDA 11 and cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.9.0.rc2. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- 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 2021.1.3.14
- TensorRT 7.2.3.4
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1

- DALI 1.2.0
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.6
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.05 is based on 1.9.0.rc2. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- A new Gluon operator `mxnet.gluon.nn.SpatialParallelConv3D` was added that implements 3D convolution of features-last batchsize-1 inputs spatially distributed across multiple GPUs--each one of n GPUs holds an input of shape [1, D, H, W, C] and the convolution is performed as if it were done on the full [1, n*D, H, W, C] input tensor.
- A new Gluon operator `mxnet.gluon.nn.SpatialParallelSplit` was added that implements a spatial 1:n split of a tensor across multiple GPUs. For n GPUs, the

output for an input of shape [1, n*D, H, W, C] is [1,D,H,W,C], where each GPU holds a different part of the input.

- A new Gluon operator `mxnet.gluon.nn.SpatialParallelAllgather` was added that implements a n:1 spatial allgather of a tensor from multiple GPUs. For n GPUs, the output for an input of shape [1, D, H, W, C] is [1, n*D, H, W, C], so that every GPU holds the full tensor.
- Ubuntu 20.04 with May 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.06	20.04	NVIDIA CUDA 11.3.1	1.9.0.rc2	TensorRT 7.2.3.4
21.05	20.04	NVIDIA CUDA 11.3.0	1.8.0	TensorRT 7.2.3.4
21.04	20.04	NVIDIA CUDA 11.3.0	1.8.0	TensorRT 7.2.3.4
21.03	20.04	NVIDIA CUDA 11.2.1	1.8.0	TensorRT 7.2.2.3
21.02	20.04	NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12	20.04	NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0.rc0	TensorRT 7.2.1
20.10	18.04	NVIDIA CUDA 11.1.0	1.7.0	TensorRT 7.2.1
20.09	18.04	NVIDIA CUDA 11.0.3	1.7.0	TensorRT 7.1.3
20.08	18.04	NVIDIA CUDA 11.0.3	1.6.0	TensorRT 7.1.3
20.07	18.04	NVIDIA CUDA 11.0.194	1.6.0	TensorRT 7.1.3
20.06	18.04	NVIDIA CUDA 11.0.167	1.6.0	TensorRT 7.1.2

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 41. Release 21.05

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, container release is intended for use on the NVIDIA Ampere Architecture A100 GPU, as well as on previous generation GPUs like V100 and T4, and with the latest CUDA 11 and cuDNN 8 libraries. The container image is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.8.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- 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
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1

- DALI 1.0.0
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.6
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.05 is based on 1.8.0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- A new Gluon operator `mxnet.gluon.nn.SpatialParallelConv3D` was added that implements 3D convolution of features-last batchsize-1 inputs spatially distributed across multiple GPUs--each one of n GPUs holds an input of shape [1, D, H, W, C] and the convolution is performed as if it were done on the full [1, n*D, H, W, C] input tensor.
- A new Gluon operator `mxnet.gluon.nn.SpatialParallelSplit` was added that implements a spatial 1:n split of a tensor across multiple GPUs. For n GPUs, the

output for an input of shape [1, n*D, H, W, C] is [1,D,H,W,C], where each GPU holds a different part of the input.

- A new Gluon operator `mxnet.gluon.nn.SpatialParallelAllgather` was added that implements a n:1 spatial allgather of a tensor from multiple GPUs. For n GPUs, the output for an input of shape [1, D, H, W, C] is [1, n*D, H, W, C], so that every GPU holds the full tensor.
- Ubuntu 20.04 with April 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.05	20.04	NVIDIA CUDA 11.3.0	1.8.0	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	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12				TensorRT 6.0.1
19.11				
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 42. Release 21.04

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 21.04, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon recently released Apache MXNet version 1.8.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- Horovod 0.21.2
- 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
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- DALI 1.0.0
- Tensor Core optimized example:

- ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.04 is based on 1.8.0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- An option was added to use the new execution engine (enabled by setting the environment variables MXNET_ASYNC_GPU_ENGINE and HOROVOD_ENABLE_ASYNC_COMPLETION to 1) which potentially improves the execution speed. The improvement is especially visible for the imperative models as well as large scale training of models using Horovod.
- Ubuntu 20.04 with March 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.04	20.04	NVIDIA CUDA 11.3.0	1.8.0	TensorRT 7.2.3.4
21.03		NVIDIA CUDA 11.2.1		TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 43. Release 21.03

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 21.03, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon recently released Apache MXNet version 1.8.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including Python 3.8
- NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.1026
- NVIDIA cuDNN 8.1.1
- NVIDIA NCCL 2.8.4 (optimized for NVLink™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- MLNX_OFED 5.1
- OpenMPI 4.0.5
- Horovod 0.21.2
- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- 7.2.2.3+cuda11.1.0.024
- DALI 0.31.0
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1

- Jupyter Notebook 6.0.3
- JupyterLab 1.2.6
- JupyterLab Server 1.0.6
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.03 is based on 1.8.0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- An expanded functionality InstanceNorm operator that adds fp64 and fp16 data I/O capability to the existing fp32 support. Also an experimental InstanceNormV2 operator with additional parameters: axis (for NHWC support), act_type (for integrated relu) and xbuf_group/xbuf_ptr (for multi-GPU operation).
- Improved the performance of the softmax, log_softmax and softmin operators, especially for cases with small or odd number of elements per row.
- Improved the performance of the BatchNorm operator when used with fp16 and axis=-1 options.
- Improved performance of multiple operators when operating on large input.
- Latest version of NVIDIA CUDA 11.2.1 including cuBLAS 11.4.1.1026
- Latest version of NVIDIA cuDNN 8.1.1
- Latest version of Horovod 0.21.2
- Latest version of DALI 0.31.0

- Ubuntu 20.04 with February 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.03	20.04	NVIDIA CUDA 11.2.1	1.8.0	TensorRT 7.2.2.3
21.02		NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 44. Release 21.02

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 21.02, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.8.0.rc2. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- MLNX_OFED 5.1
- OpenMPI 4.0.5
- Horovod 0.21.0 with enhancements
- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- 7.2.2.3+cuda11.1.0.024
- DALI 0.29
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1

- Jupyter Notebook 6.0.3
- JupyterLab 1.2.6
- JupyterLab Server 1.0.6
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 21.02 is based on 1.8.0.rc2. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- Improved performance of bias addition and gradient for Convolution (Conv*D in Gluon) and FullyConnected (Dense in Gluon) layers.
- Added 1D and 3D features-last support (NWC and NDHWC layouts) for Convolution and Deconvolution (Conv*D, Conv*DTranspose in Gluon).
- Latest version of NVIDIA CUDA 11.2.0 including cuBLAS 11.3.1
- Latest version of NVIDIA cuDNN 8.1.0
- Latest versionb of NVIDIA NCCL 2.8.4
- Latest version of 7.2.2.3+cuda11.1.0.024
- Latest version of Horovod 0.21.0
- Latest version of Nsight Compute 2020.3.0.18
- Latest version of Nsight Systems 2020.4.3.7
- Latest version of DALI 0.29
- Ubuntu 20.04 with January 2021 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
21.02	20.04	NVIDIA CUDA 11.2.0	1.8.0.rc2	7.2.2.3+cuda11.1.0.024
20.12		NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 45. Release 21.01

The NVIDIA container image release for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet 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 46. Release 20.12

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.12, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.8.0.rc0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 20.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- MLNX_OFED 5.1
- OpenMPI 4.0.5
- Horovod 0.20.2 with enhancements
- Nsight Compute 2020.2.1.8
- Nsight Systems 2020.3.4.32
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- TensorRT 7.2.2
- DALI 0.28
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1

- Jupyter Notebook 6.0.3
- JupyterLab 1.2.6
- JupyterLab Server 1.0.6
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.12 is based on 1.8.0.rc0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- Latest version of NVIDIA CUDA 11.1.1 including cuBLAS 11.3.0
- Latest version of NVIDIA cuDNN 8.0.5
- Latest versionb of NVIDIA NCCL 2.8.3
- Latest version of TensorRT 7.2.2
- Latest version of DALI 0.28
- Ubuntu 20.04 with November 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.12	20.04	NVIDIA CUDA 11.1.1	1.8.0.rc0	TensorRT 7.2.2
20.11	18.04	NVIDIA CUDA 11.1.0		TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- The 1D NCHW Deconvolution operator may produce incorrect output under these very narrow constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias and an output fed into an Addoperator. Also, the 1D NCHW Convolution operator may produce an incorrect data gradient under the constraints: input_features=1, output_features=1,

input_size=output_size, stride=1, cross-correlation mode, no bias, an input fed into some other operator and MXNET_EXEC_ENABLE_ADDTO=1 set in the environment. This will be fixed in a future release.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 47. Release 20.11

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.11, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.8.0.rc0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- Amazon Labs Sockeye sequence-to-sequence framework 2.2.3 (for machine translation)
- MLNX_OFED 5.1
- OpenMPI 4.0.5
- Horovod 0.20.2 with enhancements
- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- TensorRT 7.2.1
- DALI 0.27
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1

- Jupyter Notebook 6.0.3
- JupyterLab 1.2.6
- JupyterLab Server 1.0.6
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.11 is based on 1.8.0.rc0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- The environment variable controlling usage of CUDA graphs was changed to align with the upstream Apache MXNet 1.8.0 and is now MXNET_ENABLE_CUDA_GRAPHS (in the previous container releases it was MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH).
- Support for CUDA graphs was added to MXNet-TensorRT integration.
- The speed of applying pointwise fusion to a graph was greatly improved.
- Support for CUDA enhanced compatibility was added.
- Latest version of NVIDIA NCCL 2.8.2
- Latest version of TensorRT 7.2.1
- Latest version of DALI 0.27
- Ubuntu 18.04 with October 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.11	18.04	NVIDIA CUDA 11.1.0	1.8.0.rc0	TensorRT 7.2.1
20.10			1.7.0	
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- The 1D NCHW Deconvolution operator may produce incorrect output under these very narrow constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias and an output

fed into an Addoperator. Also, the 1D NCHW Convolution operator may produce an incorrect data gradient under the constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias, an input fed into some other operator and MXNET_EXEC_ENABLE_ADDTO=1 set in the environment. This will be fixed in a future release.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 48. Release 20.10

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.10, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.7.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

> **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 2.1.16 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.6
- Horovod 0.19.1 with enhancements
- Nsight Compute 2020.2.0.18
- Nsight Systems 2020.3.4.32
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- TensorRT 7.2.1
- DALI 0.26
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.10 is based on 1.7.0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- Increased performance of computing gradient of convolutions with bias.
- Latest version of NVIDIA CUDA 11.1.0
- Latest version of cuBLAS 11.2.1
- Latest version of NVIDIA cuDNN 8.0.4
- Latest version of TensorRT 7.2.1
- Latest version of DALI 0.26
- Latest version of Nsight Compute 2020.2.0.18
- Latest version of Nsight Systems 2020.3.4.32
- Latest version of Amazon Labs Sockeye sequence-to-sequence framework 2.1.16

- Latest version of GluonCV Toolkit 0.7
- Ubuntu 18.04 with September 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.10	18.04	NVIDIA CUDA 11.1.0	1.7.0	TensorRT 7.2.1
20.09		NVIDIA CUDA 11.0.3		TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of

parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

- The 1D NCHW Deconvolution operator may produce incorrect output under these very narrow constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias and an output fed into an Addoperator. Also, the 1D NCHW Convolution operator may produce an incorrect data gradient under the constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias, an input fed into some other operator and MXNET_EXEC_ENABLE_ADDTO=1 set in the environment. This will be fixed in a future release.
- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 49. Release 20.09

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.09, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.7.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

> **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 2.1.16 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.6
- Horovod 0.19.1 with enhancements
- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- GluonCV Toolkit 0.7
- GluonNLP Toolkit 0.8.1
- TensorRT 7.1.3
- DALI 0.25.1
- Tensor Core optimized example:

- ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.6
  - JupyterLab Server 1.0.6
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.09 is based on 1.7.0. Apache MXNet is an effort undergoing incubation at The Apache Software Foundation (ASF).
- Increased performance of computing gradient of convolutions with bias.
- Latest version of NVIDIA cuDNN 8.0.4
- Latest version of DALI 0.25.1
- Latest version of Amazon Labs Sockeye sequence-to-sequence framework 2.1.16
- Latest version of GluonCV Toolkit 0.7
- Ubuntu 18.04 with August 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.09	18.04	NVIDIA CUDA 11.0.3	1.7.0	TensorRT 7.1.3
20.08			1.6.0	
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- The 1D NCHW Deconvolution operator may produce incorrect output under these very narrow constraints: input_features=1, output_features=1, input_size=output_size, stride=1, cross-correlation mode, no bias and an output fed into an Addoperator. Also, the 1D NCHW Convolution operator may produce an incorrect data gradient under the constraints: input_features=1, output_features=1,

input_size=output_size, stride=1, cross-correlation mode, no bias, an input fed into some other operator and MXNET_EXEC_ENABLE_ADDTO=1 set in the environment. This will be fixed in a future release.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 50. Release 20.08

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.08, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.6.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.6
- Horovod 0.19.1 with enhancements
- Nsight Compute 2020.1.2.4
- Nsight Systems 2020.3.2.6
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.1.3
- DALI 0.24
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.08 is based on Apache MXNet 1.6.0
- Latest version of NVIDIA CUDA 11.0.3 including cuBLAS 11.2.0
- Latest version of NVIDIA cuDNN 8.0.2
- Latest version of DALI 0.24
- Latest version of NVIDIA NCCL 2.7.8
- Latest version of Nsight Compute 2020.1.2.4
- Latest version of Nsight Systems 2020.3.2.6
- Ubuntu 18.04 with July 2020 updates
- Decreased GPU memory consumption.
- Improved performance when using AMP with layout optimization for networks using the BatchNorm operator.

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.08	18.04	NVIDIA CUDA 11.0.3	1.6.0	TensorRT 7.1.3
20.07		NVIDIA CUDA 11.0.194		
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo where for some values of sequence length the inference may fail with the error `CUDA Driver: operation failed due to a previous error during capture`. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export`

`MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 51. Release 20.07

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.07, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.6.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.6
- Horovod 0.19.1 with enhancements
- Nsight Compute 2020.1.1.8
- Nsight Systems 2020.3.2.6
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.1.3
- DALI 0.23
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.07 is based on Apache MXNet 1.6.0
- Latest version of NVIDIA CUDA 11.0.194 including cuBLAS 11.1.0
- Latest version of DALI 0.23
- Latest version of NVIDIA NCCL 2.7.6
- Latest version of Nsight Compute 2020.1.1.8
- Latest version of Nsight Systems 2020.3.2.6
- Latest version of TensorRT 7.1.3
- Ubuntu 18.04 with June 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA Optimized Deep Learning Framework Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.07	18.04	NVIDIA CUDA 11.0.194	1.6.0	TensorRT 7.1.3
20.06		NVIDIA CUDA 11.0.167		TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo where for some values of sequence length the inference may fail with the error `CUDA Driver: operation failed due to a previous error during capture`. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.

Chapter 52. Release 20.06

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.06, is available on NGC.

Contents of the MXNet container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, which is based upon Apache MXNet version 1.6.0. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.6
- Horovod 0.19.1 with enhancements
- Nsight Compute 2020.1.0.33
- Nsight Systems 2020.2.5.8
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.1.2
- DALI 0.22
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 6.0.0
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - Jupyter-TensorBoard

Driver Requirements

Release 20.06 is based on NVIDIA CUDA 11.0., 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.06 is based on MXNet 1.6.0
- Latest version of NVIDIA CUDA 11.0.167 including cuBLAS 11.1.0
- Latest version of NVIDIA cuDNN 8.0.1
- Latest version of DALI 0.22
- Latest version of NVIDIA NCCL 2.7.5
- Latest version of Nsight Compute 2020.1.0.33
- Latest version of Nsight Systems 2020.2.5.8
- Latest version of TensorRT 7.1.2
- Integrated latest NVIDIA Deep Learning SDK to support NVIDIA A100 using CUDA 11 and cuDNN 8
- Performance improvement of the NHWC FP16 BatchNorm operation

- Performance improvement of broadcast operations, such as `broadcast_add`
- Enabled support of the `MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH` in Gluon models hybridized using `hybridize(static_alloc=True,static_shape=True)` option
- Enabled pointwise fusion to be applied to more Gluon models
- Ubuntu 18.04 with May 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA MXNet Container Versions

The following table shows what versions of Ubuntu, CUDA, Apache MXNet, and TensorRT are supported in each of the NVIDIA containers for the Optimized Deep Learning Framework. For older container versions, refer to the [Frameworks Support Matrix].

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.06	18.04	NVIDIA CUDA 11.0.167	1.6.0	TensorRT 7.1.2
20.03		NVIDIA CUDA 10.2.89		TensorRT 7.0.0
20.02			1.6.0.rc2	
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo where for some values of sequence length the inference may fail with the error `CUDA Driver: operation failed due to a previous error during capture`. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export`

`MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

- Performance regression (up to 40%) was noticed for ResNet50 mixed precision training with small batch sizes, such as 16 or 32 on V100 compared to 20.03. This will be addressed in a future release.
- Performance regression (up to 6%) was noticed for ResNet50 mixed precision training with batch size 128 and 256 on T4 compared to the 20.03. This will be addressed in a future release.
- There is a known performance regression of up to 15% compared to the 20.03 release in the BERT QA demo with batch size of 1 on V100. This will be addressed in a future release.

Chapter 53. Release 20.03

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.03, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.19.0
- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.0.0
- DALI 0.19.0
- Tensor Core optimized example:

- ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 6.0.0
  - Jupyter Core 4.6.1
  - Jupyter Notebook 6.0.3
  - JupyterLab 1.2.6
  - JupyterLab Server 1.0.6
  - 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.03 is based on Apache MXNet 1.6.0
- Increased performance of `BatchNorm` operator when the `bn_group` parameter is set to more than 1 and enabled grouping for 8 and 16 GPUs in a single node
- Increased performance of the `ArgSort` operator
- Exposed cudnn control parameters such as `cudnn_algo_fwd`, previously only available in `mx.sym.Convolution` to Gluon's `gluon.nn.Conv[1D,2D,3D]`
- Increased performance of the `elementwise` operations such as `elementwise_add`, most notably for FP16 precision
- The latest version of DALI 0.19.0
- Ubuntu 18.04 with February 2020 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA MXNet Container Versions

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

<table>
<tr><th>Container Version</th><th>Ubuntu</th><th>CUDA Toolkit</th><th>Apache MXNet</th><th>TensorRT</th></tr>
<tr><td>20.03</td><td rowspan="7">18.04
16.04</td><td rowspan="4">NVIDIA CUDA 10.2.89</td><td>1.6.0</td><td rowspan="3">TensorRT 7.0.0</td></tr>
<tr><td>20.02</td><td>1.6.0.rc2</td></tr>
<tr><td>20.01</td><td rowspan="3">1.5.1 commit c98184806 from September 4, 2019</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></tr>
<tr><td>19.09</td><td>1.5.0 commit 006486af3 from August 28, 2019</td></tr>
<tr><td>19.08</td><td>1.5.0 commit 75a9e187d from June 27, 2019</td><td>TensorRT 5.1.5</td></tr>
</table>

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play—it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 54. Release 20.02

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.02, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of the NVIDIA Optimized Deep Learning Framework in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.19.0
- Nsight Compute 2019.5.0
- Nsight Systems 2020.1.1
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.0.0
- DALI 0.18.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.3
    - JupyterLab 1.2.6
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.02 is based on Apache MXNet 1.6.0.rc2 and includes upstream commits up through commit c98184806 from September 4, 2019.
- Latest version of Nsight Systems 2020.1.1
- Latest version of Horovod 0.19.0
- Latst version of DALI 0.18.0 Beta
- Latest version of Jupyter Notebook 6.0.3, JupyterLab 1.2.6
- Ubuntu 18.04 with January 2020 updates
- A new option was added for Automatic Mixed Precision: layout optimization. It can be triggered by specifying `layout_optimization=True` option in the call to `amp.init(layout_optimization=True)`. Setting this option might significantly increase throughput of training and inference of convolutional neural networks.

- Significantly improved performance of the BatchNorm operator when `use_global_stats=True` is set.

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA MXNet Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.02	18.04 16.04	NVIDIA CUDA 10.2.89	1.6.0.rc2	TensorRT 7.0.0
20.01			1.5.1 commit c98184806 from September 4, 2019	
19.12 19.11				TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 55. Release 20.01

The NVIDIA container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 20.01, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.18.1
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 7.0.0
- DALI 0.17.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.2
    - JupyterLab 1.2.4
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 20.01 is based on MXNet 1.5.1 and includes upstream commits up through commit c98184806 from September 4, 2019.
- Latest version of TensorRT 7.0.0
- Latest version of DALI 0.17.0 Beta
- Latest version of JupyterLab 1.2.4
- Ubuntu 18.04 with December 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

NVIDIA MXNet Container Versions

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

Container Version	Ubuntu	CUDA Toolkit	Apache MXNet	TensorRT
20.01	18.04	NVIDIA CUDA 10.2.89	1.5.1 commit c98184806 from September 4, 2019	TensorRT 7.0.0
19.12 19.11	16.04			TensorRT 6.0.1
19.10		NVIDIA CUDA 10.1.243		
19.09			1.5.0 commit 006486af3 from August 28, 2019	
19.08			1.5.0 commit 75a9e187d from June 27, 2019	TensorRT 5.1.5

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo where for some values of sequence length the inference may fail with the error `CUDA Driver: operation failed due to a previous error during capture`. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 56. Release 19.12

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.12, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

> **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.18.1
- Nsight Compute 2019.5.0
- Nsight Systems 2019.6.1
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 6.0.1
- DALI 0.16.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.2
    - JupyterLab 1.2.3
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.12 is based on Apache MXNet 1.5.1 and includes upstream commits up through commit c98184806 from September 4, 2019.
- The compilation speed of fused pointwise kernels in many models was improved.
- Latest version of DALI 0.16.0 Beta
- Latest version of Nsight Systems 2019.6.1
- Latest version of JupyterLab 1.2.3
- Ubuntu 18.04 with November 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of

parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 57. Release 19.11

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.11, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.18.1
- Nsight Compute 2019.5.0
- Nsight Systems 2019.5.2
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 6.0.1
- DALI 0.15.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.4
    - Jupyter Core 4.6.1
    - Jupyter Notebook 6.0.2
    - JupyterLab 1.2.2
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.11 is based on MXNet 1.5.1 and includes upstream commits up through commit c98184806 from September 4, 2019.
- Improved performance of the BERT QA inference demo, especially for T4 GPU on large batch sizes.
- Improved performance of `box_nms` operator.
- Improved performance of backward pass when using `Embedding` operator.
- Improved performance of `argmax` operator.
- Improved performance of PointWise fusion when interacting with scalars.
- Latest version of Python 3.6
- 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 Nsight Compute 2019.5.0
- Latest version of Nsight Systems 2019.5.2
- Latest version of DALI 0.15.0 Beta
- Latest versions of Jupyter Client 5.3.4, Jupyter Core 4.6.1, JupyterLab 1.2.2, and Jupyter Notebook 6.0.2
- Ubuntu 18.04 with October 2019 updates

Announcements

- Deep learning framework containers 19.11 and later include experimental support for Singularity v3.0.

Tensor Core Examples

The tensor core examples provided in GitHub and NVIDIA GPU Cloud (NGC) focus on achieving the best performance and convergence from NVIDIA Volta tensor cores by using the latest deep learning example networks and model scripts for training.

Each example model trains with mixed precision Tensor Cores on Volta and Turing, 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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- There is a known performance regression for Apache MXNet Inception V3 and Alexnet training with multi-GPU configurations. This will be fixed in 19.12.
- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 58. Release 19.10

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.10, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including Python 3.5
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.4
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.18.1
- Nsight Compute 2019.4.0
- Nsight Systems 2019.5.1
- GluonCV Toolkit 0.5
- GluonNLP Toolkit 0.8.1
- TensorRT 6.0.1
- DALI 0.14.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.3
    - Jupyter Core 4.5.0
    - Jupyter Notebook 6.0.1
    - JupyterLab 1.1.4
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.10 is based on Apache MXNet 1.5.1 and includes upstream commits up through commit c98184806 from September 4, 2019.
- Latest version of NVIDIA cuDNN 7.6.4
- Latest version of Nsight Systems 2019.5.1
- Latest version of DALI 0.14.0 Beta
- Latest version of GluonCV Toolkit 0.5
- Latest versions of Jupyter Client 5.3.3 and JupyterLab 1.1.4
- Ubuntu 18.04 with September 2019 updates

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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training

speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 59. Release 19.09

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.09, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including Python 3.5
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.3
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED
- OpenMPI 3.1.4
- Horovod 0.18.1
- Nsight Compute 2019.4.0
- Nsight Systems 2019.4.2
- GluonCV Toolkit 0.4
- GluonNLP Toolkit 0.8.1
- TensorRT 6.0.1
- DALI 0.13.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.1
    - Jupyter Core 4.5.0
    - Jupyter Notebook 6.0.1
    - JupyterLab 1.1.1
    - JupyterLab Server 1.0.6
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.09 is based on Apache MXNet 1.5.0 and includes upstream commits up through commit 006486af3 from August 28, 2019.
- Significantly improved BERT pretraining performance when using GluonNLP.
- Improved performance of backward computation of softmax operator.
- Improved performance of backward computation of FullyConnected operator when using bias.
- Added FP16 support for TopK operator.
- Added automatic fusion of operators for BatchNorm + activation, `BatchnNormAddRelu` and `NormConvolution`.
- Latest version of NVIDIA cuDNN 7.6.3

- Latest version of Horovod 0.18.1
- Latest version of GluonNLP Toolkit 0.8.1
- Latest version of TensorRT 6.0.1
- Latest version of DALI 0.13.0 Beta
- Latest versions of Nsight Compute 2019.4.0 and Nsight Systems 2019.4.
- Latest versions of Jupyter Notebook 6.0.1, JupyterLab 1.1.1, and JupyterLab Server 1.0.6.
- Ubuntu 18.04 with August 2019 updates

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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- There is a known performance issue with the `resnet50v1.5` MXNet 19.09 script on T4 and Titan RTX GPUs. The `NHWC_BATCHNORM_LAUNCH_MARGIN` environment variable contains a sub-optimal value. As a workaround, remove line 78 (`os.environ['NHWC_BATCHNORM_LAUNCH_MARGIN'] = "64"`) from the `/opt/mxnet/nvidia-examples/resnet50v1.5/runner` file.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 60. Release 19.08

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.08, is available on NGC.

Contents of the Optimized Deep Learning Frameworkcontainer

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including Python 3.5
- NVIDIA CUDA 10.1.243 including cuBLAS 10.2.1.243
- NVIDIA cuDNN 7.6.2
- NVIDIA NCCL 2.4.8 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

> **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED +4.0
- OpenMPI 3.1.4
- Horovod 0.16.4
- Nsight Compute 10.1.168
- Nsight Systems 2019.3.7.9
- TensorRT 5.1.5
- GluonCV Toolkit 0.4
- GluonNLP Toolkit 0.7.1
- DALI 0.12.0 Beta
- Tensor Core optimized example:

    - ResNet-50 v1.5
- Jupyter and JupyterLab:
    - Jupyter Client 5.3.1
    - Jupyter Core 4.5.0
    - Jupyter Notebook 6.0.0
    - JupyterLab 1.0.5
    - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.08 is based on Apache MXNet 1.5.0 and includes upstream commits up through commit 75a9e187d from June 27, 2019.
- 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 Horovod 0.16.4
- Added GluonCV Toolkit 0.4 and GluonNLP Toolkit 0.7.1 to the container.
- Latest version of DALI 0.12.0 Beta
- Latest version of OpenMPI 3.1.4
- Latest version of Nsight Systems 2019.3.7.9
- Latest version of MLNX_OFED +4.0

- Latest versions of Jupyter Notebook 6.0.0 and JupyterLab 1.0.5
- Included support for length parameter in softmax (PR 15159)
- Improved performance of forward computation of the `softmax` operator
- Improved performance of forward computation of the `FullyConnected` operator when using bias.
- Added a demo for fast BERT QA inference using Apache MXNet in `/workspace/examples/gluon/bert_inference`.
- Improved latency of inference done with networks using the `Dropout` operator.
- Added experimental support for CUDA Graphs inside Apache MXNet guarded by the `MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH`environment variable.
- Ubuntu 18.04 with July 2019 updates

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.

- The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- There is a known issue in the BERT QA demo, where for some values of sequence length the inference may fail with `CUDA Driver: operation failed due to a previous error during capture`error. To test those values of sequence length, change the line `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=1` to `export MXNET_EXPERIMENTAL_ENABLE_CUDA_GRAPH=0` in the `test_bert_inference` script inside the demo directory.

Chapter 61. Release 19.07

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.07, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 18.04 including Python 3.5
- NVIDIA CUDA 10.1.168 including cuBLAS 10.2.0.168
- NVIDIA cuDNN 7.6.1
- NVIDIA NCCL 2.4.7 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.99 (for machine translation)
- MLNX_OFED +3.4
- OpenMPI 3.1.3
- Horovod 0.16.2
- TensorRT 5.1.5
- DALI 0.11.0 Beta
- Tensor Core optimized example:
  - ResNet-50 v1.5
- Jupyter and JupyterLab:
  - Jupyter Client 5.3.1
  - Jupyter Core 4.5.0

- Jupyter Notebook 5.7.8
- JupyterLab 1.0.0
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.07 is based on Apache MXNet 1.5.0.rc2 and includes upstream commits up through commit 75a9e187d from June 27, 2019.
- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.07 now uses OpenBLAS instead of Atlas.
- Pointwise operators fusion was introduced to improve performance of training and inference and is controlled by the `MXNET_USE_FUSION` environment variable (on by default).
- Latest version of NVIDIA cuDNN 7.6.1
- Latest version of MLNX_OFED +3.4
- Latest versions of Jupyter Client 5.3.1, Jupyter Core 4.5.0, JupyterLab 1.0.0 and JupyterLab Server 1.0.0, including Jupyter-TensorBoard integration.
- Latest version of DALI 0.11.0 Beta
- Latest version of Amazon Labs Sockeye sequence-to-sequence framework 1.18.99
- Latest version of Ubuntu 18.04

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.

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training

speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

- There is a known issue with the pointwise fusion when calculating the gradient of the `erf` function. When training a network containing the `erf` operator, set the `MXNET_USE_FUSION=0` environment variable.

Chapter 62. Release 19.06

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.06, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.16.2
- TensorRT 5.1.5
- DALI 0.10.0 Beta
- Tensor Core optimized example:
  - 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.06 is based on Apache MXNet 1.4.1. The image also contains upstream commits up through commit 01cf29d0a from May 26, 2019 and has much of the functionality from Apache MXNet 1.5.0.rc0.
- 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
- Ubuntu 16.04 with May 2019 updates (see Announcements)

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.

- ResNet50 v1.5 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have an increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations make training these new models a feasible task.

Most of the hardware and software training optimization opportunities involve exploiting lower precision like FP16 in order to utilize the Tensor Cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines that are needed to ensure proper model training.

That is where AMP (Automatic Mixed Precision) comes into play- it automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows how to get started with mixed precision training using AMP for Apache MXNet, using by example the SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Announcements

In the next release, we will no longer support Ubuntu 16.04. Release 19.07 will instead support Ubuntu 18.04.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 63. Release 19.05

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.05, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.16.1
- TensorRT 5.1.5
- DALI 0.9.1 Beta
- 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 19.05 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.05 is based on MXNet 1.4.0commit 87c7addcd from February 12, 2019.
- 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
- Updated to Horovod 0.16.1
- New experimental `NormalizedConvolution` operator (see below).
- Ubuntu 16.04 with April 2019 updates

`NormalizedConvolution` Operator

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet 19.05 includes a new operator (`NormalizedConvolution`) to improve training speeds of CNN's like ResNet-50. The `NormalizedConvolution` operator combines the functions of BatchNorm and Convolution into one operator to reduce data transfers to and from GPU global memory. For more information regarding its use through the ResNet-50 sample model script, see `./example/image-classification/symbols/resnet-v1b-normconv-fl.py`.

`NormalizedConvolution` is supported by new APIs of cuDNN v7.6, however, its Python API is experimental until it becomes incorporated into upstream Apache MXNet.

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 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations, make it a feasible task.

However, where most of the (both hardware and software) optimization opportunities exists is in exploiting lower precision (like FP16) to, for example, utilize tensor cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines.

That is where AMP (Automatic Mixed Precision) comes into play. It automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` inside this container, shows you how to get started with mixed precision training using AMP for Apache MXNet. As an example of a network we will use SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training

speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 64. Release 19.04

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.04, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.13.11
- TensorRT 5.1.2
- DALI 0.8.1 Beta
- 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 19.04 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.04 is based on Apache MXNet 1.4.0commit 87c7addcd from February 12, 2019.
- Latest version of NVIDIA NCCL 2.4.6
- Latest version of cuBLAS 10.1.0.105
- Latest version of DALI 0.8.1 Beta
- Added support for Apache MXNet Automatic Mixed Precision; see below for more information.
- Ubuntu 16.04 with March 2019 updates

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 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Automatic Mixed Precision (AMP)

Training deep learning networks is a very computationally intensive task. Novel model architectures tend to have increasing number of layers and parameters, which slows down training. Fortunately, new generations of training hardware as well as software optimizations, make it a feasible task.

However, where most of the (both hardware and software) optimization opportunities exists is in exploiting lower precision (like FP16) to, for example, utilize tensor cores available on new Volta and Turing GPUs. While training in FP16 showed great success in image classification tasks, other more complicated neural networks typically stayed in FP32 due to difficulties in applying the FP16 training guidelines.

That is where AMP (Automatic Mixed Precision) comes into play. It automatically applies the guidelines of FP16 training, using FP16 precision where it provides the most benefit, while conservatively keeping in full FP32 precision operations unsafe to do in FP16.

The NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet AMP tutorial, located in the `/opt/mxnet/nvidia-examples/AMP/AMP_tutorial.md` directory of this container, shows you how to get started with mixed precision training using AMP for Apache MXNet. As an example of a network we will use SSD network from GluonCV.

For more information about AMP, see the Training With Mixed Precision Guide.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 65. Release 19.03

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.03, is available on NGC.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.13.11
- TensorRT 5.1.2
- DALI 0.7 Beta
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.03 is based on Apache MXNet 1.4.0.
- 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
- Optimized NMS operator performance
- Ubuntu 16.04 with February 2019 updates

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 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- 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 66. Release 19.02

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.02, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.13.11
- TensorRT 5.0.2
- DALI 0.6.1 Beta
- Tensor Core optimized example:
  - ResNet50 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 19.02 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.02 is based on 1.4.0.rc2.
- Latest version of DALI 0.6.1 Beta
- 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
- Added an image classification example in Gluon.
- Multiple enhancements to Gluon training speed with models hybridized with `static_alloc=True` setting.
- Added Python bindings for NVTX and CUDA profiler in the `mxnet.cuda_utils` package.
- Ubuntu 16.04 with January 2019 updates

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. This container includes the following Tensor Core examples.

- An implementation of the ResNet50 model. The ResNet50 v1.5 model is a slightly modified version of the original ResNet50 v1 model that trains to a greater accuracy.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- 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 67. Release 19.01

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 19.01, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

> **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.61 (for machine translation)
- OpenMPI 3.1.3
- Horovod 0.15.1
- TensorRT 5.0.2
- DALI 0.6 Beta
- Tensor Core optimized example:
    - ResNet50 v1.5

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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 19.01 is based on 1.4.0.rc0.
- Latest version of DALI 0.6 Beta
- Latest version of NVIDIA cuDNN 7.4.2
- Latest version of Sockeye
- Latest version of OpenMPI 3.1.3
- Improvements to out-of-the-box Horovod performance
- Added ResNet50 v1.5 Tensor Core optimized example.
- Ubuntu 16.04 with December 2018 updates

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. This container includes the following Tensor Core examples.

- An implementation of the ResNet50 model. The ResNet50 v1 is a modified version of the original ResNet50 v1 model.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked under 13341.

- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.
- 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 68. Release 18.12

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.12, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.28 (for machine translation)
- OpenMPI 3.1.2
- Horovod 0.15.1
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.12 is based on 1.3.0, with all upstream changes from the Apache MXNet main branch up to and including PR 13069.
- Improved handling of float32 datatype in `examples/image-classification/train_imagenet_runner`.
- Enabled NVIDIA Tools Extension SDK (NVTX) instrumentation.
- Improved speed of metrics computation during training, especially in the case of using TopKAccuracy metric.
- Latest version of DALI 0.5.0 Beta.
- Ubuntu 16.04 with November 2018 updates

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release. Issue tracked on https://github.com/apache/incubator-mxnet/issue/13341.
- The default setting of the environment variable `MXNET_GPU_COPY_NTHREADS=1` in the container may not be optimal for all networks. Networks with a high ratio of parameters and computation, like AlexNet, may achieve greater multi-GPU training speeds with the setting `MXNET_GPU_COPY_NTHREADS=2`. Users are encouraged to try this setting for their own use case.

Chapter 69. Release 18.11

The container image for NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.11, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.28 (for machine translation)
- OpenMPI 3.1.2
- Horovod 0.15.1
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.11 is based on 1.3.0, with all upstream changes from the Apache MXNet main branch up to and including PR 12537.
- Added fused `BatchNormAddRelu` operator to the Apache MXNet Symbol package (accessible via `mx.sym.BatchNormAddRelu`), which performs BatchNorm operation on data, sums the result with a tensor and performs Relu activation on the result of the sum. Currently it is limited to FP16 data type and `NHWC` data layout.
- Added `MXNET_EXEC_ENABLE_ADDTO` environment variable, which when set to `1` increases performance for some networks.
- Increased performance of `Batchnorm` and `Batchnorm+Relu` operators in FP16 and `NHWC` data format.
- Added support for multi-node via Horovod integration. Currently you can use it by specifying `horovod` type of KVStore.
- Added `MXNET_UPDATE_ON_KVSTORE` environment variable, which controls whether to update parameters using KVStore (default is `1` for KVStore `device` and `0` for KVStore `horovod`).
- Added aggregation of SGD updates which increases performance when update on KVStore is disabled.
- Increased performance when training with small batch sizes.
- Fixed a bug that prevented matrix multiplications to overlap with other computation, which increases performance for some networks.
- Fixed an issue which prevented score function to respect not-full batches of data.
- Added `resnet-v1b` as possible network in the `train_imagenet_runner` script.
- 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.
- Ubuntu 16.04 with October 2018 updates.

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release.

- Apache MXNet ResNet50 regresses in FP32 performance. This issue should be fixed in a later release.

Chapter 70. Release 18.10

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.10, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including 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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.28 (for machine translation)
- OpenMPI 3.1.2
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.10 is based on 1.3.0, with all upstream changes from the Apache MXNet main branch up to and including PR 12537.
- Latest version of NCCL 2.3.6.
- Latest version of DALI 0.4 Beta.
- Added support for OpenMPI 3.1.2
- The known issue in the prior release regarding the variable maximum GPU global memory usage has been fixed. You should now see lower and stable global memory usage from run to run, and across GPUs in multi-GPU training.
- Ubuntu 16.04 with September 2018 updates

Known Issues

- The Apache MXNet KVStore GPU peer-to-peer communication tree discovery, as of release 18.09, is not compatible with DGX-1V. Only users that set the environment variable `MXNET_KVSTORE_USETREE=1` will experience issues, which will be resolved in a subsequent release.
- Apache MXNet ResNet50 regresses in FP32 performance. This issue should be fixed in a later release.

Chapter 71. Release 18.09

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.09, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04 including Python 3.5
- NVIDIA CUDA 10.0 including CUDA® Basic Linear Algebra Subroutines library™ (cuBLAS) 10.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.3.0
- NCCL 2.3.4 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.
- Amazon Labs Sockeye sequence-to-sequence framework 1.18.28 (for machine translation)
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.09 is based on 1.3.0, with all upstream changes from the Apache MXNet main branch up to the creation point of the v1.3.x branch (PR 12301), plus all substantive cherry-picks from main that were included in the v1.3.0 release.
- The demonstrator of mixed precision ResNet-50 training using the `NHWC` data layout has been expanded to work now on the Turing architecture in addition to Volta.
- Latest version of cuDNN 7.3.0.
- Latest version of CUDA 10.0 which includes support for DGX-2, Turing, and Jetson Xavier.
- Latest version of cuBLAS 10.0.
- Latest version of NCCL 2.3.4.
- Latest version of TensorRT 5.0.0 RC.
- Latest version of DALI 0.2 Beta.
- Ubuntu 16.04 with August 2018 updates

Known Issues

The multi-threaded nature of Apache MXNet model execution may result in a variable maximum usage of GPU global memory, as discussed in earlier release notes. Users that experience sporadic out-of-GPU-memory errors should experiment with setting the environment variable `MXNET_GPU_WORKER_NTHREADS=1` as a possible remedy. We anticipate the need for this experimentation will be removed in our next release.

Chapter 72. Release 18.08

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.08, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.28 (for machine translation)
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.08 is based on 1.2.0, with all upstream changes from the Apache MXNet main branch up to and including PR 11545.
- Latest version of cuDNN 7.2.1.
- Latest version of DALI 0.1.2 Beta.
- New demonstrator of increased mixed-precision ResNet-50 training speeds on Volta when processed end-to-end in the `NHWC` data layout. We are working to PR the code improvements to upstream Apache MXNet. To evaluate in the meantime, type `/opt/mxnet/examples/image_classification/train_imagenet_runner --batch-size N` . Substitute 256 for `N` on systems with GPUs having 32GB global memory (or 192 with 16GB GPUs) and prepare the imagenet database as directed in `nvidia-examples/imagenet_preparations`. Training images should be pre-resized to 480px shorter side and validation should be pre-resized to 256px shorter side. The script expects RecordIO files to be present in the `/data/imagenet/train-480-val-256-recordio/` directory.
- Ubuntu 16.04 with July 2018 updates

Announcements

Starting with the next major version of the CUDA release, we will no longer provide updated Python 2 containers and will only update Python 3 containers.

Known Issues

The multi-threaded nature of Apache MXNet model execution may result in a variable maximum usage of GPU global memory. Users that experience sporadic out-of-GPU-memory errors should experiment with setting the environment variable `MXNET_GPU_WORKER_NTHREADS=1` as a possible remedy. We anticipate the need for this experimentation will be removed in a subsequent release.

Chapter 73. Release 18.07

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.07, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.23 (for machine translation)
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.07 is based on 1.2.0, with all upstream changes from the Apache MXNet main branch up to and including PR 11302.
- 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

Announcements

Starting with the next major version of the CUDA release, we will no longer provide updated Python 2 containers and will only update Python 3 containers.

Known Issues

- Some of the unit tests available in `/opt/mxnet/tests/python/{gpu,unittest}/*.py` require the SciPy Python library. For those that want to run the unit tests, first install SciPy by `pip install scipy`. There is no longer a need to specifically request the 1.0 version of SciPy.
- The multi-threaded nature of Apache MXNet model execution may result in a variable maximum usage of GPU global memory. Users that experience sporadic out-of-GPU-memory errors should experiment with setting the environment variable `MXNET_GPU_WORKER_NTHREADS=1` as a possible remedy. We anticipate the need for this experimentation will be removed in a subsequent release.

Chapter 74. Release 18.06

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.06, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.22 (for machine translation)
- 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 Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.06 is based on 1.2.0. Specifically, container image 18.06 has merged all commits on upstream Apache MXNet main, up to the creation point of the v1.2.0 branch, and all commits on that branch up to the 1.2.0 tag.
- Container includes TensorRT 4.0.1
- TensorRT integration examples for in-framework inference can be found in `/workspace/examples/tensorrt-integration`. This includes a LeNet-5 unit test and a ResNet-50 example.
- Support added for DALI iterators.
- Ubuntu 16.04 with May 2018 updates

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

Some of the unit tests available in `/opt/mxnet/tests/python/{gpu,unittest}/*.py` require the SciPy Python library. For those that want to run the unit tests, first install the 1.0 version of SciPy by typing `pip install scipy==1.0`.

Note: The latest SciPy release, version 1.1, is not compatible with the unit tests.

Chapter 75. Release 18.05

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.05, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.13 (for machine translation)

Driver Requirements

Release 18.05 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This Optimized Deep Learning Framework release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.05 is based on Apache MXNet 1.1.0.
- For this month, no upstream merges as we work toward incorporating the upcoming Apache MXNet 1.2.0 release.
- Ubuntu 16.04 with April 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

Those wishing to run the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet unit tests under `/opt/mxnet/tests/python` should install SciPy using `pip install scipy==1.0`, as the recently available SciPy v1.1 is not compatible with all the unit tests. For more information, see Broken test_sparse_operator.test_sparse_mathematical_core with scipy 1.1.0.

Chapter 76. Release 18.04

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.04, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  Note: The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.18.1 (for machine translation)

Driver Requirements

Release 18.04 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.04 is based on NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet 1.1.0.
- For this month, no upstream merges as we develop a performant approach to Apache MXNet's recent operator refactoring to a stateless imperative style.
- ResNet-50 performance improvement based on the automatic fusion of `add_relu` and `copy_split` backward pass.
- 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 are no known issues in this release.

Chapter 77. Release 18.03

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.03, is available.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.17.4 (for machine translation)

Driver Requirements

Release 18.03 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.03 is based on Apache MXNet 1.1.0.
- Incorporated all upstream changes from the NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet main branch, specifically, PR 9749.
- Added compute-graph optimizations for improved ResNet performance.
- Latest version of cuBLAS 9.0.333
- Latest version of cuDNN 7.1.1
- 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 78. Release 18.02

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.02, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.02 is based on Apache MXNet 1.1.0.

Contents of Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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
- NCCL 2.1.2 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.17.0 (for machine translation)

Driver Requirements

Release 18.02 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- 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.

Chapter 79. Release 18.01

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 18.01, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 18.01 is based on Apache MXNet 1.0.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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
- NCCL 2.1.2 (optimized for NVLink™)
- ONNX exporter 0.1 for CNN classification models

  > **Note:** The ONNX exporter is being continuously improved. You can try the latest changes by pulling from the main branch.

- Amazon Labs Sockeye sequence-to-sequence framework 1.16.2 (for machine translation)

Driver Requirements

Release 18.01 is based on CUDA 9, which requires NVIDIA Driver release 384.xx.

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Addition of Python 3 package
- Enhanced-performance cuDNN-based batched 1D convolutions (merged to upstream)
- Added `MxNet-to-ONNX` exporter for classification of CNN models (tested with LeNet-5, ResNet-50, etc.).
- Added the Sockeye sequence-to-sequence framework, along with a German-to-English translation model, based on the WMT'15 dataset and translation task. This model's launch script should reproduce the OpenNMT reference model when trained until convergence.
- 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 17.12 container.

Chapter 80. Release 17.12

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.12, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.12 is based on Apache MXNet 1.0.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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
- 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Both the `nccl` and `nccl_allreduce` KVStore options now have the same, improved performance.
- Latest version of CUDA
- Latest version of cuDNN
- Latest version of NCCL
- Ubuntu 16.04 with November 2017 updates

Known Issues

There are no known issues in this release.

Chapter 81. Release 17.11

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.11, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.11 is based on Apache MXNet 0.12.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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
- 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Added Sockeye to the container, including NCCL `kvstore` option
- Enabled `mx.sym.batch_dot` to use FP16
- 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 82. Release 17.10

The NVIDIA container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.10, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.10 is based on Apache MXNet 0.11.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.3
- 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Mixed precision support for all optimizers
- New image input pipeline with faster speed and support for global shuffling after each epoch when used with `IndexedRecordIO` format.
- 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 83. Release 17.09

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.09, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.09 is based on Apache MXNet 0.11.0.rc3.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 9.0
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 7.0.2
- 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 NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Tensor Core support in convolutions, deconvolutions, and fully connected layers on Volta
- Support for mixed precision training with SGD optimizer
- Streamlined FP16 examples for image classification
- Optimized input pipeline for image processing
- 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 84. Release 17.07

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.07, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.07 is based on Apache MXNet 0.10.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

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
- NCCL 2.0.3 (optimized for NVLink™)

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Support for multi-precision SGD
- cuBLAS back-end for `FullyConnected` operation
- Ubuntu 16.04 with June 2017 updates

Known Issues

There are no known issues in this release.

Chapter 85. Release 17.06

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.06, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.06 is based on Apache MXNet 0.10.0.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NCCL 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Implemented double buffering in ResNet v1 example
- Ubuntu 16.04 with May 2017 updates

Known Issues

There are no known issues in this release.

Chapter 86. Release 17.05

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.05, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.05 is based on Apache MXNet 0.9.3a+.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.21
- NCCL 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Latest cuDNN release
- Improved IO pipeline for increased multi-GPU performance
- Optimized SGD weight update
- Added the `nccl_allreduce` option for gradient communication
- Added support for dilated deconvolutions
- Improved convolutional neural network (CNN) performance by removing unnecessary computations
- Added options to show the cuDNN algorithms that are chosen for convolutions
- Ubuntu 16.04 with April 2017 updates

Known Issues

There are no known issues in this release.

Chapter 87. Release 17.04

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.04, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.04 is based on Apache MXNet 0.9.3a+.

Contents of the Optimized Deep Learning Framework container

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.20
- NCCL 1.6.1 (optimized for NVLink™)

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Support for cuDNN accelerated dilated convolutions
- Ubuntu 16.04 with March 2017 updates

Known Issues

There are no known issues in this release.

Chapter 88. Release 17.03

The container image of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet, release 17.03, is available.

NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet container image version 17.03 is based on Apache MXNet 0.9.3.

Contents of the Optimized Deep Learning Framework

This container image contains the complete source of the version of NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet in `/opt/mxnet`. It is pre-built and installed to the Python path.

The container also includes the following:

- Ubuntu 16.04
- NVIDIA CUDA® 8.0.61
- NVIDIA CUDA® Deep Neural Network library™ (cuDNN) 6.0.20

Key Features and Enhancements

This NVIDIA Optimized Deep Learning Framework, powered by Apache MXNet release includes the following key features and enhancements.

- Ubuntu 16.04 with February 2017 updates
- Improved input pipeline for image processing
- Support for FP16 training of AlexNet
- Optimized embedding layer of CUDA kernels
- Optimized tensor broadcast and reduce CUDA kernels

Known Issues

There are no known issues in this release.

Notice

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