TensorFlow Wheel

Release Notes

v | July 2025

Table of Contents

Chapter 1. Overview

TensorFlow Wheel

This TensorFlow Wheel release is intended for use on the NVIDIA Ampere Architecture GPU, NVIDIA Turing Architecture GPUs, NVIDIA Volta Architecture GPUs, and NVIDIA Pascal Architecture GPU.

Chapter 2. TensorFlow Wheel Platform

This TensorFlow Wheel release is intended for use on the NVIDIA Ampere Architecture GPU, NVIDIA Turing Architecture GPUs, NVIDIA Volta Architecture GPUs, and NVIDIA Pascal Architecture GPU.

2.1. TensorFlow Wheel Release 23.03

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 1. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=12.1
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=12.0
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=12.0
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	12.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=12.1
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.8
NVIDIA CUDA cuFFT	nvidia-cufft	>=11.0
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=12.0
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.23.0+nv23.03

NVIDIA Product		Version
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.23.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.27.0+nv23.03
NVIDIA CUDA NCCL	nvidia-nccl	>=2.17
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.23.0 + nv23.03
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.5

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), 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.

Software Requirements

The 23.03 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 23.03 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 23.03
- TensorFlow Wheel Release 23.02

- TensorFlow Wheel Release 23.01
- TensorFlow Wheel Release 22.12
- TensorFlow Wheel Release 22.11
- TensorFlow Wheel Release 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is

concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Reverting to the old optimizers (available now under **tf.keras.optimizers.legacy**, e.g. tf.keras.optimizers.legacy.Adam instead of tf.keras.optimizers.Adam) resolves the errors. We also have an in-flight Horovod PR 3822 that fixes more cases.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash. This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

2.2. TensorFlow Wheel Release 23.02

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 2. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=12.0
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=12.0
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=12.0
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	12.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=12.0
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.7
NVIDIA CUDA cuFFT	nvidia-cufft	>=11.0
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=12.0
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.22.0+nv23.02
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.22.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.26.1+nv23.01

NVIDIA Product		Version
NVIDIA CUDA NCCL	nvidia-nccl	>=2.16
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.22.0 + nv23.02
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.5

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), 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.

Software Requirements

The 23.02 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 23.02 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 23.02
- TensorFlow Wheel Release 23.01
- TensorFlow Wheel Release 22.12
- TensorFlow Wheel Release 22.11
- TensorFlow Wheel Release 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08

- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the

paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- In 23.02 containers, certain cuDNN cases that use runtime compilation via NVRTC, particularly on ARM SBSA systems, can fail with the following:

  ```
  CUDNN_STATUS_RUNTIME_PREREQUISITE_MISSING
  ```

  A workaround for this situation is to export the following:

  ```
  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-11/lib64
  ```

  This will be fixed in the next release.
- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Reverting to the old optimizers (available now under **tf.keras.optimizers.legacy**, e.g. tf.keras.optimizers.legacy.Adam instead of tf.keras.optimizers.Adam) resolves the errors. We also have an in-flight Horovod PR 3822 that fixes more cases.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash. This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

2.3. TensorFlow Wheel Release 23.01

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 3. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=12.0
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=12.0
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=12.0
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=12.0
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.7
NVIDIA CUDA cuFFT	nvidia-cufft	>=11.0
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=12.0
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.21.0+nv23.01
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.21.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.26.1+nv23.01

NVIDIA Product		Version
NVIDIA CUDA NCCL	nvidia-nccl	>=2.16
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv23.01
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.5

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), 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.

Software Requirements

The 23.01 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 23.01 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 23.01
- TensorFlow Wheel Release 22.12
- TensorFlow Wheel Release 22.11
- TensorFlow Wheel Release 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07

- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the

paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Due to a dependency issue, `pip install nvidia-tensorflow[horovod]` may pick up an older version of cuBLAS unless `pip install nvidia-cublas-cu11~=11.8.0` is issued first.
- Note that if you wish to make modifications to the source and rebuild TensorFlow, starting from Container Release 22.10 (TensorFlow 2.10) you will need a C++ 17-compatible compiler.
- The default set of Keras optimizers are not currently compatible with Horovod, see github issues [1], [2]. Reverting to the old optimizers (available now under **tf.keras.optimizers.legacy**, e.g. tf.keras.optimizers.legacy.Adam instead of tf.keras.optimizers.Adam) resolves the errors. We also have an in-flight Horovod PR 3822 that fixes more cases.
- Some DLRM models may regress by 10-40%. We are currently investigating.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash. This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- On H100 NVLink systems using 2 GPUs for training, certain communication patterns can trigger a corner-case bug that manifests either as a hang or as an "illegal instruction" exception. A workaround for this case is to set the environment variable `NCCL_PROTO=^LL128`. This issue will be addressed in an upcoming release.
- Within the TF1 container on T4 GPUs, the MaskRCNN model may fail with either the low accuracy or illegal memory access. The root cause is under investigation and will be fixed in a future release.

2.4. TensorFlow Wheel Release 22.12

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 4. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.11
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.8
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.8
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.8
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.7
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.9
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.20.0+nv22.12
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.20.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.26.1+nv22.12

NVIDIA Product		Version
NVIDIA CUDA NCCL	nvidia-nccl	>=2.15
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv22.12
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.5

Driver Requirements

Release 22.12 is based on CUDA 11.8, 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), 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.

Software Requirements

The 22.12 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.12 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 22.12
- TensorFlow Wheel Release 22.11
- TensorFlow Wheel Release 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06

- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed

precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash. This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

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

2.5. TensorFlow Wheel Release 22.11

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 5. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.11
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.8
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.8
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.8
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.7
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.9
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.18.0+nv22.11
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.18.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.26.1+nv22.11
NVIDIA CUDA NCCL	nvidia-nccl	>=2.15
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv22.11
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.5

Driver Requirements

Release 22.11 is based on CUDA 11.8, 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), 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.

Software Requirements

The 22.11 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.11 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 22.11
- TensorFlow Wheel Release 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09

- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the

specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Certain models using RELU activation may exhibit extreme (and easily noticeable) performance regressions. We have root-cased this to a cuDNN issue and will release the fix in 22.12.

- Certain models may crash with an out-of-memory error. We are investigating and will fix in 22.12.
- A known performance regression of up to 50% affects some efficientnet models. The regression is inherited from upstream tensorflow and is still under investigation. It will be fixed in a subsequent release.
- The TF-TRT native segment fallback has a known issue that causes a crash. This issue occurs when you use TF-TRT to convert a model with a subgraph that is then converted to TensorRT, but the conversion fails to build. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

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

2.6. TensorFlow Wheel Release 22.10

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 6. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.11
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.8
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.8
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.8
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.6
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.9
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.18.0+nv22.10
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.18.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.25.0+nv22.10
NVIDIA CUDA NCCL	nvidia-nccl	>=2.15
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv22.10
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.5

Driver Requirements

Release 22.10 is based on CUDA 11.8, 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), 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.

Software Requirements

The 22.10 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.10 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 22.10
- TensorFlow Wheel Release 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11

- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D

image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Some multi-GPU TF2 models (e.g. EfficientNet) may crash with a segmentation fault. As a potential workaround, try increasing the host memory limit from the default of 64GB, by setting the environment variable `TF_GPU_HOST_MEM_LIMIT_IN_MB=131072`, which is MBs.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

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

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

2.7. TensorFlow Wheel Release 22.09

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 7. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.11
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.8
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.8
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.8
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.6
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.9
NVIDIA CUDA cuRAND	nvidia-curand	>=10.3
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7

NVIDIA Product		Version
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.17.0+nv22.09
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.17.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.25.0+nv22.09
NVIDIA CUDA NCCL	nvidia-nccl	>=2.15
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv22.09
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.5

Driver Requirements

Release 22.09 is based on CUDA 11.8, 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), 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.

Software Requirements

The 22.09 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.09 is based on 1.15.5.
- We introduced a new environment variable `TF_GRAPPLER_GRAPH_DEF_PATH` to output the Graphdef files before and after the TF grappler optimizations (For more information about the grappler optimizations, see the TensorFlow graph optimization

with Grappler). In checking the optimized operation graph during the TF runtime, users can specify the following:

```
TF_GRAPPLER_GRAPH_DEF_PATH=/path/to/graphdef
```

- We provided a visualization tool to convert (and compare) the given Graphdef files by `graphdef2pydot`, which was preinstalled in the 22.09 container. For more information about usage, see `graphdef2pydot -h`.

NVIDIA TensorFlow Wheel Versions

- 22.09
- TensorFlow Wheel Release 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- We have introduced another feature to be able to switch to the channel-last layout (NHWC) for harnessing the power of Tensor Core math. If you observed a performance regression in TF-TRT models, consider enabling the environment variable via export `TF_ENABLE_LAYOUT_NHWC=1` to check whether it helps regain the lost performance.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

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

2.8. TensorFlow Wheel Release 22.08

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 8. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.10
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.7
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.7
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.7
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.5
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.16.0+nv22.08
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.16.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.25.0+nv22.08
NVIDIA CUDA NCCL	nvidia-nccl	>=2.12
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	==1.16.0 + nv22.08
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5

NVIDIA Product		Version
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.5

Driver Requirements

Release 22.08 is based on CUDA 11.7 Update 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.

Software Requirements

The 22.08 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.08 is based on 1.15.5.
- We introduced a new environment variable `TF_ENABLE_LAYOUT_NHWC` to enforce the NHWC layout at runtime. In some models with fp32 on Ampere GPUs, users may obtain better performance when specifying `TF_ENABLE_LAYOUT_NHWC=1`, which can better utilize the TF32 tensor cores.

NVIDIA TensorFlow Wheel Versions

- 22.08
- TensorFlow Wheel Release 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01

- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

  ```
  export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
  ```
- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.
- There is a known performance regression in XLA that can cause performance regressions of up to 55% when training certain models such as EfficientNet with XLA enabled. The root cause is under investigation and will be fixed in a future release.
- TF-TRT 22.07 may fail to build TensorRT engines for HF BERT or HF BART, which may manifest as large performance regressions. Please revert back to the previous version 22.06 if you see a TF-TRT warning stating that Myelin graph could not be created or see a substantial performance regression.

2.9. TensorFlow Wheel Release 22.07

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 9. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.10
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.7
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.7
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.7
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.4
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.15.0+nv22.07
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.15.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.24.3+nv22.07
NVIDIA CUDA NCCL	nvidia-nccl	>=2.12
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.5

Driver Requirements

Release 22.07 is based on NVIDIA CUDA 11.7, which requires NVIDIA Driver release 515 or later.

Software Requirements

The 22.07 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.07 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 22.07
- TensorFlow Wheel Release 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

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

2.10. TensorFlow Wheel Release 22.06

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 10. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.10
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.7
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.7
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.7
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.4
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.4
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.14.0+nv22.06
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.14.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.24.3+nv22.06
NVIDIA CUDA NCCL	nvidia-nccl	>=2.12
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.5

Driver Requirements

Release 22.06 is based on NVIDIA CUDA 11.7, which requires NVIDIA Driver release 515 or later.

Software Requirements

The 22.06 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.06 is based on 1.15.5.
- Added support for NHWC TF32 2D convolutions in XLA.
- TensorFlow 2.9 improves the functionality of `prefetch_to_device` to allow for concurrent kernel execution and data transfer. To make use of this feature, ensure that your dataset pipeline ends by applying the `prefetch_to_device` operation as follows.

```
dataset = dataset.batch(batch_size=1024)
…
dataset = dataset.apply(tf.data.experimental.prefetch_to_device('/gpu:0'))
```

NVIDIA TensorFlow Wheel Versions

- 22.06
- TensorFlow Wheel Release 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05

- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model

is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

```
cannot allocate memory in static TLS block
```

The workaround is to run the following command:

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

- IO dominated CNN models, such as AlexNet and ResNet50 see a ~10% performance reduction on some platforms. The regression is under investigation and will be fixed in a future release.
- In some configurations, the UNet3D model on A100 fails to initialize CUDNN due to an OOM. This can be fixed by increasing the GPU memory carveout with the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB=2000`.

2.11. TensorFlow Wheel Release 22.05

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 11. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.10
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.7
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.7
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.7
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.4
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.3
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.13.0+nv22.05
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.13.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.24.2+nv22.05
NVIDIA CUDA NCCL	nvidia-nccl	>=2.12

NVIDIA Product		Version
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.4.2

Driver Requirements

Release 22.05 is based on NVIDIA CUDA 11.7, which requires NVIDIA Driver release 515 or later.

Software Requirements

The 22.05 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.05 is based on 1.15.5.
- Fixed segfault in SparseToDense when validate_indices if false for both TF1 and TF2.
- Fixed XLA device indexing issue in TF2 that caused out-of-memory errors when using Horovod to distribute work to multiple GPUs.
- Removed unneeded copies when saving resource variables. This lowers the effective memory footprint for models with large layers (e.g., embedding layers in recommender models).
- Optimized depthwise convolution backprop filter kernel, providing speedups between 10 and 100x over previous implementation.

NVIDIA TensorFlow Wheel Versions

- 22.05
- TensorFlow Wheel Release 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12

- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides

product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

  ```
  cannot allocate memory in static TLS block
  ```

  The workaround is to run the following command:

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

2.12. TensorFlow Wheel Release 22.04

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 12. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.9
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.6
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.6
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.6
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.4
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.3
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.12.0+nv22.04
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.12.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.24.2+nv22.04

NVIDIA Product		Version
NVIDIA CUDA NCCL	nvidia-nccl	>=2.12
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.4.2

Driver Requirements

Release 22.04 is based on NVIDIA CUDA 11.6.2, which requires NVIDIA Driver release 510 or later.

Software Requirements

The 22.04 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.04 is based on 1.15.5.
- Container sizes were reduced by removing redundant PTX code sections.
- Fixed the race condition in the cuDNN heuristics lookup that might sometimes lead to segmentation faults.
- TF2 added cuTENSOR support for the einsum single label case.
- Fixed pooling operations to support tensors with dimensions that exceed the 32-bit integer indexing.

NVIDIA TensorFlow Wheel Versions

- 22.04
- TensorFlow Wheel Release 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11

- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data

for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT

but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.

- A known issue affects aarch64 libgomp, which might sometimes cause the following error:

```
cannot allocate memory in static TLS block
```

The workaround is to run the following command:

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

2.13. TensorFlow Wheel Release 22.03

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 13. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.8
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.6
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.6
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.6
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.3
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.3
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.11.1+nv22.03
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.11.1
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.23.0+nv22.03
NVIDIA CUDA NCCL	nvidia-nccl	>=2.11

NVIDIA Product		Version
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2

Driver Requirements

Release 22.03 is based on NVIDIA CUDA 11.6.1, which requires NVIDIA Driver release 510 or later.

Software Requirements

The 22.03 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.03 is based on 1.15.5.
- The following vulnerabilities were patched in this release of TensorFlow 1.15.5:

  CVE-2020-10531, CVE-2021-41197, CVE-2021-41206, CVE-2021-41208, CVE-2022-21725, CVE-2022-21728, CVE-2022-21729, CVE-2022-21730, CVE-2022-21731, CVE-2022-21732, CVE-2022-21733, CVE-2022-21734, CVE-2022-21735, CVE-2022-21736, CVE-2022-21737, CVE-2022-21739, CVE-2022-21741, CVE-2022-23557, CVE-2022-23558, CVE-2022-23559, CVE-2022-23562, CVE-2022-23563, CVE-2022-23564, CVE-2022-23565, CVE-2022-23566, CVE-2022-23567, CVE-2022-23568, CVE-2022-23569, CVE-2022-23571, CVE-2022-23573, CVE-2022-23575, CVE-2022-23576, CVE-2022-23577,CVE-2022-23578, CVE-2022-23579, CVE-2022-23580, CVE-2022-23581, CVE-2022-23582, CVE-2022-23583, CVE-2022-23584, CVE-2022-23585, CVE-2022-23586, CVE-2022-23588, CVE-2022-23589, and CVE-2022-23591.

- Fixed a bug in the XLA convolution autotuner that appeared in the 22.01-tf2 release which sometimes caused **Failed to determine best cudnn convolution algorithm: RESOURCE_EXHAUSTED** errors.

- TensorFlow 1.15 has been patched for compatibility with numpy 1.21.1, and the numpy version has been updated to that version.

  With older numpy releases, certain matrix operations resulted in NaN and Inf values on ARM SBSA.

NVIDIA TensorFlow Wheel Versions

- 22.03
- TensorFlow Wheel Release 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1

convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.

2.14. TensorFlow Wheel Release 22.02

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 14. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.8
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.6
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.6
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.6
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.3

NVIDIA Product		Version
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.3
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.10.0+nv22.02
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.10.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.23.0+nv22.02
NVIDIA CUDA NCCL	nvidia-nccl	>=2.11
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2

Driver Requirements

Release 22.02 is based on NVIDIA CUDA 11.6.0, which requires NVIDIA Driver release 510 or later.

Software Requirements

The 22.02 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.02 is based on 1.15.5.
- For TF2 added CudnnMHA Keras op to expose CUDNN's optimized multi-head attention implementation.
- Fixed segmentation fault when VLOG logging was enabled in TF1.
- Updated TF-TRT with latest upstream changes.

- Fixed bug in TF2 where CUDNN's fused batched norm grad kernels could be called when `training = false`.
- Extended autotuning over CUDNN fallback engines. This change may increase the execution time of the first few iterations, but can result in substantially better engines being chosen during later iterations.

NVIDIA TensorFlow Wheel Versions

- 22.02
- TensorFlow Wheel Release 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1

convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.

2.15. TensorFlow Wheel Release 22.01

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 15. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.8
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.6
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.6
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.6
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.3

NVIDIA Product		Version
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.7
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.3
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.9.0+nv22.01
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.9.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.23.0+nv22.01
NVIDIA CUDA NCCL	nvidia-nccl	>=2.11
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	==1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2

Driver Requirements

Release 22.01 is based on NVIDIA CUDA 11.6.0, which requires NVIDIA Driver release 510 or later.

Software Requirements

The 22.01 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 22.01 is based on 1.15.5.
- Fixed circular dependency in monolithic builds to address github issue 21.
- TF-TRT respects enable_tensor_float_32_execution Python API.
- TF-TRT supports Structured Sparsity on NVIDIA Ampere architecture GPUs. This can be enabled by passing `enable_sparse_compute=True` to `TrtGraphConverterV2`.

NVIDIA TensorFlow Wheel Versions

- 22.01
- TensorFlow Wheel Release 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the

existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- Debugging with TF_CPP_MIN_VLOG_LEVEL=3 can result in a segmentation while auto-tuning convolution algorithms. This will be fixed in the 22.02 release.

2.16. TensorFlow Wheel Release 21.12

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 16. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	>=11.7.3.1
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	>=11.5.57
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	>=11.5.50
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	>=11.5.50
NVIDIA CUDA cuDNN	nvidia-cudnn	>=8.3.1.22
NVIDIA CUDA cuFFT	nvidia-cufft	>=10.6.0.54
NVIDIA CUDA cuRAND	nvidia-curand	>=10.2.6.48

NVIDIA Product		Version
NVIDIA CUDA cuSOLVER	nvidia-cusolver	>=11.2.1.48
NVIDIA CUDA cuSPARSE	nvidia-cusparse	>=11.7.0.31
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	==1.8.0+nv21.12
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	==1.8.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	==0.22.1+nv21.12
NVIDIA CUDA NCCL	nvidia-nccl	==2.11.4
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	<1.16.0,>=1.15.0
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	==8.2.1.8

Driver Requirements

Release 21.12 is based on NVIDIA CUDA 11.5.0, which requires NVIDIA Driver release 495 or later.

Software Requirements

The 21.12 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 21.12 is based on 1.15.5.
- The environment variable `TF_DISABLE_REDUCED_PRECISION_REDUCTION=1` can now be set to disable intermediate reductions in lower precision than the requested math type.
- Patched the following CVEs in TensorFlow 1.15.5: CVE-2021-29571, CVE-2021-29592, CVE-2021-29601, CVE-2021-29608, CVE-2021-29609, CVE-2021-29613, CVE-2021-22876, CVE-2021-22897, CVE-2021-22898, CVE-2021-22901, CVE-2021-37636, CVE-2021-37640, CVE-2021-37642, CVE-2021-37644, CVE-2021-37646, CVE-2021-37653, CVE-2021-37660, CVE-2021-37661,

CVE-2021-37668, CVE-2021-37669, CVE-2021-37670, CVE-2021-37672, CVE-2021-37673, CVE-2021-37674, CVE-2021-37675, CVE-2021-37684, CVE-2021-37686, CVE-2021-37690, CVE-2021-37691, CVE-2021-41195, CVE-2021-41196, CVE-2021-41197, CVE-2021-41198, CVE-2021-41199, CVE-2021-41200, CVE-2021-41201, CVE-2021-41202, CVE-2021-41203, CVE-2021-41204, CVE-2021-41206, CVE-2021-41207, CVE-2021-41208, CVE-2021-41213, CVE-2021-41215, CVE-2021-41216, CVE-2021-41217, CVE-2021-41218, CVE-2021-41219, CVE-2021-41221, CVE-2021-41222, CVE-2021-41223, CVE-2021-41224, CVE-2021-41225, CVE-2021-41228, CVE-2021-22922, CVE-2021-22923, CVE-2021-22924, CVE-2021-22925, CVE-2021-22926

NVIDIA TensorFlow Wheel Versions

- 21.12
- TensorFlow Wheel Release 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is

concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known performance regression affecting UNet Medical 3D model training by up to 23%. This will be addressed in a future release.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- TensorFlow Wheel release 21.12 has a known corruption issue in its NVTX profiling markers when using the CUPTI library from CUDA Toolkit version 11.5. An updated CUPTI build, numbered 11.5.57 or higher, in CUDA 11.5 Update 1 will address this issue.

2.17. TensorFlow Wheel Release 21.11

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 17. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.7.3.1
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.5.*
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.5.50
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.5.50
NVIDIA CUDA cuDNN	nvidia-cudnn	8.3.0.96
NVIDIA CUDA cuFFT	nvidia-cufft	10.6.0.54
NVIDIA CUDA cuRAND	nvidia-curand	10.2.6.48
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.2.1.48
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.7.0.31
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.7.0+nv21.11
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.7.0
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.1+nv21.11
NVIDIA CUDA NCCL	nvidia-nccl	2.11.4
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.11
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	8.0.3.4

Driver Requirements

Release 21.11 is based on NVIDIA CUDA 11.5.0, which requires NVIDIA Driver release 495 or later.

Software Requirements

The 21.11 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel release 21.11 is based on 1.15.5.
- Patched CVE-2021-37663 in TensorFlow 1.

NVIDIA TensorFlow Wheel Versions

- 21.11
- TensorFlow Wheel Release 21.10
- TensorFlow Wheel Release 21.09
- TensorFlow Wheel Release 21.08
- TensorFlow Wheel Release 21.07
- TensorFlow Wheel Release 21.06
- TensorFlow Wheel Release 21.05
- TensorFlow Wheel Release 21.04
- TensorFlow Wheel Release 21.03
- TensorFlow Wheel Release 21.02
- TensorFlow Wheel Release 21.01
- TensorFlow Wheel Release 20.12
- TensorFlow Wheel Release 20.11
- TensorFlow Wheel Release 20.10
- TensorFlow Wheel Release 20.09
- TensorFlow Wheel Release 20.08
- TensorFlow Wheel Release 20.07
- TensorFlow Wheel Release 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the

paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- A CUDNN performance regression can cause slowdowns of up to 15% in certain ResNet models. This will be fixed in a future release.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 includes a feature that works around this issue, but TF1 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- TF-TRT native segment fallback has a known issue causing a crash. This will occur when using TF-TRT to convert a model with a subgraph that is converted to TensorRT but fails to build at runtime. Instead of falling back to native TensorFlow TF-TRT will crash. Using `export TF_TRT_OP_DENYLIST="ProblematicOp"` can help to prevent conversion of an OP causing a native segment fallback.
- TensorFlow Wheel release 21.11 has a known corruption issue in its NVTX profiling markers when using the CUPTI library from CUDA Toolkit version 11.5. An updated CUPTI build, numbered 11.5.57 or higher, in CUDA 11.5 Update 1 will address this issue.

2.18. TensorFlow Wheel Release 21.10

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 18. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.6.5.2
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.4.120

NVIDIA Product		Version
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.4.120
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.4.108
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.4.15
NVIDIA CUDA cuFFT	nvidia-cufft	10.5.2.100
NVIDIA CUDA cuRAND	nvidia-curand	10.2.5.120
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.2.0.120
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.6.0.120
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.6.0+nv21.10
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.6.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.6.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2021.3.2.12
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.1+nv21.10
NVIDIA CUDA NCCL	nvidia-nccl	2.11.4
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.6.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.10
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	8.0.3.4

Driver Requirements

Release 21.10 is based on NVIDIA CUDA 11.4.2 + cuBLAS 11.6.5.2, which requires NVIDIA Driver release 470 or later. However, if you are running on Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 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.

Software Requirements

The 21.10 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.10 is based on 1.15.5.
- Improved handling of exp ops in XLA.
- Enable pointwise row vectorization for small rows in XLA.
- Integrate latest TF-TRT features for dynamic shape support.
- Gemm+bias+relu cublasLt based epilogue fusion in XLA. This feature can be enabled by setting the environment variable `TF_USE_CUBLASLT=1`.

NVIDIA TensorFlow Wheel Versions

- 21.10
- 21.09
- 21.08
- 21.07
- 21.06
- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12

- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Announcements

- Starting with the 21.10 release, a beta version of the TensorFlow 1 and 2 containers is available for the ARM SBSA platform. Pulling the Docker image `nvcr.io/nvidia/tensorflow:21.10-tf2-py3` on an ARM SBSA machine will automatically fetch the ARM-specific image.
- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). These models are still available on Github or as model-specific containers from the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on specific use cases, users may need to install some packages that were previously pre-installed.
- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is `slurm-wlm-basic-plugins`).
- The `nvtx-plugins` utility package pre-installed in previous releases has been removed. Users depending on nvtx-plugins can install it using:

```
pip install nvtx-plugins
```

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which require downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed

precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the `21.06-tf1` release. This will be fixed in a future release.
- The OpenSeq2Seq toolkit has been removed from the TensorFlow 1.x container.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 includes a feature that works around this issue, but TF1 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.

2.19. TensorFlow Wheel Release 21.09

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 19. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.6.1.51
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.4.120
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.4.120
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.4.108
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.4.15
NVIDIA CUDA cuFFT	nvidia-cufft	10.5.2.100
NVIDIA CUDA cuRAND	nvidia-curand	10.2.5.120
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.2.0.120
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.6.0.120

NVIDIA Product		Version
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.5.0+nv21.09
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.5.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.5.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2021.3.1.57
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.1+nv21.09
NVIDIA CUDA NCCL	nvidia-nccl	2.11.4
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.5.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.09
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	8.0.3.0

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 Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 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.

Software Requirements

The 21.09 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.09 is based on 1.15.5.
- The following vulnerabilities have been patched in the TensorFlow 1.x container: CVE-2021-37678, CVE-2021-37679, CVE-2021-37659, CVE-2021-37676, CVE-2021-37667, CVE-2021-37650, CVE-2021-37671, CVE-2021-37665, CVE-2021-37664, CVE-2021-37655, CVE-2021-37641, CVE-2021-37662, CVE-2021-37656, CVE-2021-37658, CVE-2021-37643, CVE-2021-37648, CVE-2021-37647, CVE-2021-37635, CVE-2021-37638, CVE-2021-37657, CVE-2021-37639, CVE-2021-37666, CVE-2021-37652, CVE-2021-37654, and CVE-2021-37651.
- CublasLT integration in native TensorFlow and XLA. Allows for more flexible matmul fusions in XLA.

NVIDIA TensorFlow Wheel Versions

- 21.09
- 21.08
- 21.07
- 21.06
- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07

- 20.06

Announcements

- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). These models are still available on Github or as model-specific containers from the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on specific use cases, users may need to install some packages that were previously pre-installed.

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100

GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Support for SLURM PMI2 is deprecated and will be removed after the 21.12 release. PMIX is supported by the container, but is not supported by default in SLURM. Users depending on SLURM integration may need to configure SLURM for PMIX in the base OS as appropriate to their OS distribution (for Ubuntu 20.04, the required package is slurm-wlm-basic-plugins).
- The `nvtx-plugins` utility package pre-installed in previous releases has been removed. Users depending on `nvtx-plugins` can install it as `pip install nvtx-plugins`.
- For TensorFlow 1.15, TF-TRT inference throughput may regress for certain models by up to 37% compared to the 21.06-tf1 release. This will be fixed in a future release.
- The OpenSeq2Seq toolkit has been removed from the TensorFlow 1.x container.

- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 21.08 includes a feature that works around this issue, but TF1 21.08 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.

2.20. TensorFlow Wheel Release 21.08

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 20. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.5.4.8
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.4.100
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.4.100
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.4.108
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.2.26
NVIDIA CUDA cuFFT	nvidia-cufft	10.5.1.100
NVIDIA CUDA cuRAND	nvidia-curand	10.2.5.100
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.2.0.100
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.6.0.100
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.4.0+nv21.08
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.4.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.4.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's	nvidia-nsys-cli	2021.2.4.12

NVIDIA Product		Version
algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC		
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.1+nv21.08
NVIDIA CUDA NCCL	nvidia-nccl	2.10.3
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.4.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.08
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	8.0.1.6

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 Tesla (for example, T4 or any other Tesla board), you may use NVIDIA driver release 418.40 (or later R418), 440.33 (or later R440), 450.51 (or later R450), 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.

Software Requirements

The 21.08 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.08 is based on 1.15.5.

- Experimental integration of the cutensor library for einsum operations is included in the `21.08-tf2-py3` container. This should improve performance for many einsum operations. To enable export `TF_ENABLE_CUTENSOR_EINSUM=1`.
- Added XLA feature to de-select compilation candidates based on shape inference. To enable this feature, use the environment variable `TF_XLA_DO_NOT_COMPILE_POSSIBLE_DYNAMIC_OPS`.
- Bug fixes for `cudaMallocAsync` GPU memory allocator.
- MKL is enabled for better performance in CPU-only workloads. To enable, set `OMP_NUM_THREADS` to a value `>= 1`.

NVIDIA TensorFlow Wheel Versions

- 21.08
- 21.07
- 21.06
- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Announcements

- The TensorCore example models are no longer provided in the core container (previously shipped in `/workspace/nvidia-examples`). These models are still available on Github or as model-specific containers from the NVIDIA GPU Cloud (NGC). Some python packages, included in previous containers to support these example models, have also been removed. Depending on specific use cases, users may need to install some packages that were previously pre-installed.

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is

concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- The OpenSeq2Seq toolkit is deprecated and will be removed startin in the `21.09-tf1-py3` release. This only affects the TensorFlow 1.x release.
- There is a known issue in TensorRT 8.0 regarding accuracy for a certain case of int8 inferencing on A40 and similar GPUs. The version of TF-TRT in TF2 21.08 includes a feature that works around this issue, but TF1 21.08 does not include that feature and may experience the accuracy drop for a small subset of model/data type/batch size combinations on A40. This will be fixed in the next version of TensorRT.
- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes. There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.21. TensorFlow Wheel Release 21.07

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 21. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.5.2.43
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.4.65
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.4.48
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.4.43
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.2.26
NVIDIA CUDA cuFFT	nvidia-cufft	10.5.0.43
NVIDIA CUDA cuRAND	nvidia-curand	10.2.5.43
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.2.0.43
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.6.0.43
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.3.0+nv21.07
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.3.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.3.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2021.2.1.58
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.1+nv21.07
NVIDIA CUDA NCCL	nvidia-nccl	2.10.3
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.2.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.07
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5

NVIDIA Product		Version
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	8.0.1.6

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.

Software Requirements

The 21.07 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.07 is based on 1.15.5.
- Integrate TRT 8 Support
- Increase GPU memory reservation to avoid OOM errors in some cases.
- Improve NVTX markers to include XLA cluster names.
- Fix deadlock in XLA by backporting upstream PR 50280 to TF1 and TF2.
- Fix bug so that CUDNN now respects TF32 disable switch.
- TF2 implements support for embedding ops on GPU
  - SparseFillEmptyRows[Grad]
  - fp16 embedding_lookup_sparse
  - fp16 SparseSegmentSumGrad
  - SparseSegmentSum/Mean
  - SparseSegmentSum/MeanGrad
  - hash value to string
- TF2 - Use CUDA occupancy calculator to improve performance of BiasAdd

NVIDIA TensorFlow Wheel Versions

- 21.07
- 21.06
- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the

specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- TF1 and TF2 containers include a version of Django with a known vulnerability that was discovered late in our QA process. See CVE-2021-35042 for details. This will be fixed in the next release.

- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.22. TensorFlow Wheel Release 21.06

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 22. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.5.1.109
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.3.111
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.3.109
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.3.109
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.1.32
NVIDIA CUDA cuFFT	nvidia-cufft	10.4.2.109
NVIDIA CUDA cuRAND	nvidia-curand	10.2.4.109
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.1.2.109
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.6.0.109
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.2.0+nv21.06
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.2.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.2.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's	nvidia-nsys-cli	2021.2.1.58

NVIDIA Product		Version
algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC		
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.22.0+nv21.06
NVIDIA CUDA NCCL	nvidia-nccl	2.9.9
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.2.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.06
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.3.4

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.

Software Requirements

The 21.06 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.06 is based on 1.15.5.

- Fixed bug that caused XLA to initialize TensorFlow on all visible GPUs leading to OOM errors in Horovod and other multi-process configurations.
- Fixed bug in `FakeQuantizeAndDequantize` op that would result in non-symmetric quantization when `max=-min`.
- Implemented GPU kernels for ops common in recommender model input pipelines: `SparseApplyFtrl`, `[Sparse]ApplyProximalAdagrad`, `SparseReshape`, `SparseToDense`.
- Vectorized GPU Gather op to improve performance.
- Introduced env var `TF_CPP_VLOG_FILENAME` to direct VLOG output to a file.
- Improved CUDNN kernel selection by switching to `CUDNN_HEUR_B` kernel selector.
- Updated tensorflow-addons to r0.13.
- Added support for FussedBatchNormGrad op to optimize side-inputs and activations.
- Patched recently announced vulnerabilities in TF 1.15.5: CVE-2021-29591, CVE-2021-29605, CVE-2021-29606, CVE-2021-29614.

NVIDIA TensorFlow Wheel Versions

- 21.06
- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1

convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- In certain cases, TensorFlow may claim too much memory on Pascal-based GPUs leading to failures due to OOM and potentially an application hang. This can be worked around by setting the environment variable `TF_DEVICE_MIN_SYS_MEMORY_IN_MB` to `675`. This will be fixed in the 21.07 release.
- A known regression can reduce the training performance of VGG-16 by up to 12% at certain batch sizes.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.23. TensorFlow Wheel Release 21.05

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 23. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.5.1.101
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.3.58
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.3.58
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.3.58
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.0.51

NVIDIA Product		Version
NVIDIA CUDA cuFFT	nvidia-cufft	10.4.2.58
NVIDIA CUDA cuRAND	nvidia-curand	10.2.4.58
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.1.1.58
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.5.0.58
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.0.0+nv21.05
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.0.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.1.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2021.2.1.58
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.21.3+nv21.05
NVIDIA CUDA NCCL	nvidia-nccl	2.9.8
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.2.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.4
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.3.4

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.

Software Requirements

The 21.05 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.05 is based on 1.15.5.

NVIDIA TensorFlow Wheel Versions

- 21.05
- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1

convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
TF_NUM_INTEROP_THREADS=1 horovodrun -np 8 bash -c
'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK python ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- There is a known cuDNN performance regression affecting certain batch sizes of VGG based models by up to 45%. This will be fixed in a later release.

2.24. TensorFlow Wheel Release 21.04

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 24. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.5.1.101
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.3.58
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.3.58
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.3.58

NVIDIA Product		Version
NVIDIA CUDA cuDNN	nvidia-cudnn	8.2.0.41
NVIDIA CUDA cuFFT	nvidia-cufft	10.4.2.58
NVIDIA CUDA cuRAND	nvidia-curand	10.2.4.58
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.1.1.58
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.5.0.58
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	1.0.0+nv21.04
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	1.0.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.1.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2021.2.1.58
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.21.3+nv21.04
NVIDIA CUDA NCCL	nvidia-nccl	2.9.6
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	1.2.0
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.4
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.3.4

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.

Software Requirements

The 21.04 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.04 is based on 1.15.5.
- Add GPU-deterministic tf.sparse.sparse_dense_matmul support (for the tf.float32 data type). When TF_DETERMINISTIC_OPS is set to "true" or "1" then tf.sparse.sparse_dense_matmul will operate deterministically in both the forward and backward direction.
- Integrated CUDNN v8 API for RNN and fused conv+bias+activation ops.
- Fixed an issue that caused OOM errors in some cases when using a batch size of 1.
- Improved XLA handling of dynamic ops to avoid frequent recompilation.
- Implemented XLA persistent cache.
- Implemented custom learning rate support in Horovod.

NVIDIA TensorFlow Wheel Versions

- 21.04
- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
TF_NUM_INTEROP_THREADS=1 horovodrun -np 8 bash -c
'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK python ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- There is a known CUDNN performance regression affecting certain batch sizes of VGG based models by up to 45%. This will be fixed in a later release.

2.25. TensorFlow Wheel Release 21.03

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 25. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.4.1.1026
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.2.135
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.2.142
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.2.146
NVIDIA CUDA cuDNN	nvidia-cudnn	8.1.1.33
NVIDIA CUDA cuFFT	nvidia-cufft	10.4.0.135
NVIDIA CUDA cuRAND	nvidia-curand	10.2.3.135
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.1.0.135
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.4.0.135
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.31.0+nv21.03
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	0.31.0
NVIDIA DLprof binary installation	nvidia-dlprof	1.0.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2020.4.1.117
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.21.3+nv21.03
NVIDIA CUDA NCCL	nvidia-nccl	2.8.4
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	0.12
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.3
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5

NVIDIA Product		Version
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.2.3

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.xx, 440.30, 450.51, or 455.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 and NVIDIA CUDA and Drivers Support.

Software Requirements

The 21.03 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.03 is based on 1.15.5.
- CUDA 11.2.1 including cuBLAS 11.4.1.1026
- NVIDIA cuDNN 8.1.1
- DALI 0.31.0
- DLProf 1.0.0
- Ubuntu 20.04
- NVTX profiling annotation ranges more accurately report the execution of asynchronous operations. Note that when profiling NVTX ranges must now be explicitly enabled by setting the environment variable `TF_ENABLE_NVTX_RANGES=1`.
- The CUDNN backend API is now used for convolutional ops. This provides a significant performance benefit by reducing CPU overheads of convolutions.
- The fused Conv+Bias+Relu op regression in CUDNN has been fixed and this op has been re-enabled in both XLA and the TF grappler optimizers. This improves performance particularly for inference in convolutional models.
- Bugs relating to auto-graph in TensorFlow 1.15 with Python 3.8 were fixed.

NVIDIA TensorFlow Wheel Versions

- 21.03
- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-

art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- Using XLA together with Horovod to parallelize training on a single node can result in out-of-memory errors. A workaround is to execute the job as follows. This will be fixed in a future release.

```
XLA_FLAGS=--xla_multiheap_size_constraint_per_heap=2000000000
TF_NUM_INTEROP_THREADS=1 horovodrun -np 8 bash -c
'CUDA_VISIBLE_DEVICES=$OMPI_COMM_WORLD_LOCAL_RANK python ...'
```

- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.

- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- Training the UNET3D models with a batch size of 1 can result in OOM (Out-Of-Memory) in the TensorFlow 1 container. This is caused by the map_and_batch_fusion optimizer from using the tf.datasets. One workaround solution is to add:

```
if self._batch_size == 1:
    options = dataset.options()
    options.experimental_optimization.map_and_batch_fusion = False
    dataset = dataset.with_options(options)
```

2.26. TensorFlow Wheel Release 21.02

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 26. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.3.1.68
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.2.67
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.2.67
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.2.72
NVIDIA CUDA cuDNN	nvidia-cudnn	8.1.0.77
NVIDIA CUDA cuFFT	nvidia-cufft	10.4.0.72
NVIDIA CUDA cuRAND	nvidia-curand	10.2.3.68
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.0.2.68
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.3.1.68
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.29.0+nv21.02
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	0.29.0
NVIDIA DLprof binary installation	nvidia-dlprof	0.19.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to	nvidia-nsys-cli	2020.4.1.117

NVIDIA Product		Version
visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC		
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.21.0+nv21.02
NVIDIA CUDA NCCL	nvidia-nccl	2.8.4
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	0.11
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv21.2
NVIDIA TensorFlow	nvidia-tensorflow	1.15.5
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.2.3

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.xx, 440.30, 450.51, or 455.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 and NVIDIA CUDA and Drivers Support.

Software Requirements

The 21.02 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 21.02 is based on 1.15.5.

- CUDA 11.2.0 including cuBLAS 11.3.1
- NVIDIA cuDNN 8.1.0
- NCCL 2.8.4 (optimized for NVLink)
- TensorRT 7.2.2
- Nsight Compute 2020.3.0.18
- Nsight Systems 2020.4.3.7
- DALI 0.29.0
- DLProf 0.19.0
- Ubuntu 20.04

NVIDIA TensorFlow Wheel Versions

- 21.02
- 21.01
- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

Note: If you encounter functional or performance issues when XLA is enabled, please refer to the XLA Best Practices document. It offers pointers on how to diagnose symptoms and possibly address them.

- A regression (only observed with NVIDIA Ampere GPU architecture) in cuDNN's fused Convolution+Bias+Activation implementation can cause performance regressions of up to 24% in some models such as UNet Medical. This will be fixed in a future cuDNN release.
- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4, which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- Training the UNET3D models with a batch size of 1 can result in OOM (Out-Of-Memory) in the TensorFlow 1 container. This is caused by the map_and_batch_fusion optimizer from using the tf.datasets. One workaround solution is to add:

```
if self._batch_size == 1:
    options = dataset.options()
    options.experimental_optimization.map_and_batch_fusion = False
    dataset = dataset.with_options(options)
```

2.27. TensorFlow Wheel Release 21.01

The NVIDIA container image release for TensorFlow Wheel 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.

2.28. TensorFlow Wheel Release 20.12

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 27. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.3.0.106
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.1.105
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.1.105
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.1.74
NVIDIA CUDA cuDNN	nvidia-cudnn	8.0.5.43
NVIDIA CUDA cuFFT	nvidia-cufft	10.3.0.105
NVIDIA CUDA cuRAND	nvidia-curand	10.2.2.105
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.0.1.105
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.3.0.10
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.28.0+nv20.12
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	0.28.0
NVIDIA DLprof binary installation	nvidia-dlprof	0.18.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2020.4.1.117

NVIDIA Product		Version
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.20.2+nv20.12
NVIDIA CUDA NCCL	nvidia-nccl	2.8.3
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	0.10
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv20.12
NVIDIA TensorFlow	nvidia-tensorflow	1.15.3
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.2.1

Driver Requirements

Release 20.12 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.xx, 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.

Software Requirements

The 20.12 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 20.04 (64-bit)
- Python 3.8
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.12 is based on 1.15.4.
- CUDA 11.1.1 including cuBLAS 11.3.0
- NVIDIA cuDNN 8.0.5
- NCCL 2.8.3 (optimized for NVLink)
- TensorRT 7.2.2
- Nsight Systems 2020.4.1.117
- OpenMPI 4.0.5

- DLProf 0.18.0
- Ubuntu 20.04

NVIDIA TensorFlow Wheel Versions

- 20.12
- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language

Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- In certain cases running on Pascal GPUs may result in out-of-memory errors which may present as apparent job hangs. This can be worked around by exporting the following environment variable:

```
TF_DEVICE_MIN_SYS_MEMORY_IN_MB=550
```

- A regression in cuDNN's fused Convolution+Bias+Activation implementation can cause performance regressions of up to 24% in some models such as UNet Medical. This will be fixed in a future cuDNN release.
- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4, which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and

MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.

- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy:

```
TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
```

- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- There is a known issue of OOM (Out-Of-Memory) when training the UNET3D models when batch size = 1 in TensorFlow (TF1) container.

2.29. TensorFlow Wheel Release 20.11

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 28. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.2.1.74
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.1.69
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.1.74

NVIDIA Product		Version
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.1.74
NVIDIA CUDA cuDNN	nvidia-cudnn	8.0.4.30
NVIDIA CUDA cuFFT	nvidia-cufft	10.3.0.74
NVIDIA CUDA cuRAND	nvidia-curand	10.2.2.74
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.0.0.74
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.2.0.275
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.27.0+nv20.11
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	0.27.0
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.27.0+nv20.11
NVIDIA DLprof binary installation	nvidia-dlprof	0.17.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2020.4.1.117
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.20.2+nv20.11
NVIDIA CUDA NCCL	nvidia-nccl	2.8.2
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	0.9
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv20.11
NVIDIA TensorFlow	nvidia-tensorflow	1.15.3
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.1.6

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.xx, 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.

Software Requirements

The 20.11 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.11 is based on 1.15.4.
- CUDA 11.1.0
- cuDNN 8.0.4
- TensorRT 7.2.1
- DALI 0.27
- DLProf 0.17.0

NVIDIA TensorFlow Wheel Versions

- 20.11
- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- In certain cases running on Pascal GPUs may result in out-of-memory errors which may present as apparent job hangs. This can be worked around by exporting the following environment variable:
  ```
  TF_DEVICE_MIN_SYS_MEMORY_IN_MB=550
  ```
- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4, which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per

the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy:

```
TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
```

- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.30. TensorFlow Wheel Release 20.10

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 29. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.2.1.74
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.1.69
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.1.74
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.1.74
NVIDIA CUDA cuDNN	nvidia-cudnn	8.0.4.30
NVIDIA CUDA cuFFT	nvidia-cufft	10.3.0.74
NVIDIA CUDA cuRAND	nvidia-curand	10.2.2.74
NVIDIA CUDA cuSOLVER	nvidia-cusolver	11.0.0.74
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.2.0.275
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.26.0+nv20.10
NVIDIA DALI for CUDA 11.0	nvidia-dali-cuda110	0.26.0
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.25.1+nv20.09

NVIDIA Product		Version
NVIDIA DLprof binary installation	nvidia-dlprof	0.16.0
NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an application's algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC	nvidia-nsys-cli	2020.4.1.117
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.20.0+nv20.10
NVIDIA CUDA NCCL	nvidia-nccl	2.7.8
DLprof TensorBoard plugin	nvidia-tensorboard-plugin-dlprof	0.8
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv20.10
NVIDIA TensorFlow	nvidia-tensorflow	1.15.3
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.2.1.4

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 or 440.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.

Software Requirements

The 20.10 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.10 is based on 1.15.4.
- CUDA 11.1.0
- cuDNN 8.0.4
- TensorRT 7.2.1
- DALI 0.26
- DLProf 0.16.0

NVIDIA TensorFlow Wheel Versions

- 20.10
- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the

existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- Some image-based inference workloads see a regression of up to 50% for the smallest batch sizes. This is due to regressions in cuDNN 8.0.4, which will be addressed in a future release.
- A few models see performance regressions compared to the 20.08 release. Training WideAndDeep sees regressions of up to 30% on A100. In FP32 the TF1 Unet Industrial and Bert fine tuning training regress from 10-20%. Also the TF2 Unet Medical and MaskRCNN models regress by about 20% in some cases. These regressions will be addressed in a future release.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy:
  ```
  TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
  ```
- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100 and Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.31. TensorFlow Wheel Release 20.09

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 30. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product		Version
NVIDIA CUDA cuBLAS	nvidia-cublas	11.2.0.252
NVIDIA CUDA CUPTI	nvidia-cublas-cupti	11.0.221
NVIDIA CUDA NVCC	nvidia-cuda-nvcc	11.0.221
NVIDIA CUDA NVRTC	nvidia-cuda-nvrtc	11.0.*
NVIDIA CUDA Runtime	nvidia-cuda-runtime	11.0.221
NVIDIA CUDA cuDNN	nvidia-cudnn	8.0.4.12
NVIDIA CUDA cuFFT	nvidia-cufft	10.2.1.245
NVIDIA CUDA cuRAND	nvidia-curand	10.2.1.245
NVIDIA CUDA cuSOLVER	nvidia-cusolver	10.6.0.245
NVIDIA CUDA cuSPARSE	nvidia-cusparse	11.1.1.245
NVIDIA DALI for CUDA 11.0	nvidia-dali	0.25.1
NVIDIA DALI TensorFlow Plugin for CUDA 11.0	nvidia-dali-nvtf-plugin	0.25.1+nv20.09
Distributed training framework for TensorFlow, Keras, and PyTorch	nvidia-horovod	0.19.2+nv20.09
NVIDIA CUDA NCCL	nvidia-nccl	2.7.8
TensorBoard lets you watch Tensors Flow	nvidia-tensorboard	1.15.0+nv20.09
NVIDIA TensorFlow	nvidia-tensorflow	1.15.3
NVIDIA TensorRT, a high-performance deep learning inference library	nvidia-tensorrt	7.1.3.4

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.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.

Software Requirements

The 20.09 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.09 is based on 1.15.3.
- NVIDIA cuDNN 8.0.4
- DALI 0.25

NVIDIA TensorFlow Wheel Versions

- 20.09
- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1

convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy:
  ```
  TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
  ```
- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100 and Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.32. TensorFlow Wheel Release 20.08

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 31. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product	Version
nvidia-cublas	11.2.0.252
nvidia-cublas-cupti	11.0.221
nvidia-cuda-nvcc	11.0.221
nvidia-cuda-nvrtc	11.0.*
nvidia-cuda-runtime	11.0.221
nvidia-cudnn	8.0.2.39
nvidia-cufft	10.2.1.245
nvidia-curand	10.2.1.245
nvidia-cusolver	10.6.0.245
nvidia-cusparse	11.1.1.245
nvidia-dali	0.24
nvidia-dali-tf-plugin	0.24
nvidia-horovod	0.19.5
nvidia-nccl	2.7.8
nvidia-tensorflow	1.15.3
nvidia-tensorrt	7.1.3.4
nvidia-cuda-nvrtc	11.0

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.

Software Requirements

The 20.08 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6

- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.08 is based on 1.15.3.
- Ubuntu 18.04 with July 2020 updates

NVIDIA TensorFlow Wheel Versions

- 20.08
- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model

is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- The memory required to train MaskRCNN with a given batch size has increased from 20.07 to 20.08. As a result, the batch size may need to be decreased.
- There are several known performance regressions compared to 20.07. UNet Medical and Industrial on V100 and A100 GPUs can be up to 20% slower. VGG can be up to 95% slower on A100 and 15% slower on Turing GPUs. Googlenet can be up to 20% slower on V100. And ResNet50 inferencing can be up to 30% slower on A100 and Turing GPUs.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.08 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is

to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy:

```
TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
```

- There is a known performance regression of 15% compared to the 20.03 release when training the JoC V-Net Medical models with small batch size and fp32 data type on Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.

2.33. TensorFlow Wheel Release 20.07

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 32. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product	Version
nvidia-cublas	11.1.0
nvidia-cublas-cupti	11.0.194
nvidia-cuda-nvcc	11.0.194
nvidia-cuda-nvrtc	11.0.194
nvidia-cuda-runtime	11.0.194
nvidia-cudnn	8.0.1
nvidia-cufft	10.2.0.218
nvidia-curand	10.2.1.218
nvidia-cusolver	10.5.0.218
nvidia-cusparse	11.1.0.218
nvidia-dali	0.23
nvidia-dali-tf-plugin	0.23

NVIDIA Product	Version
nvidia-horovod	0.19.5
nvidia-nccl	2.7.6
nvidia-tensorflow	1.15.3
nvidia-tensorrt	7.1.3

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.

Software Requirements

The 20.07 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.07 is based on 1.15.3.
- Improved XLA to avoid excessive recompilations
- Enhancements for Automatic Mixed Precision with einsum, 3D Convolutions, and list operations
- Improved 3D Convolutions to support NDHWC format
- Default TF32 support

NVIDIA TensorFlow Wheel Versions

- 20.07
- 20.06

Tensor Core Examples

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

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

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as “a method for detecting objects in images using a single deep neural network”. Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod, Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and U-Net Industrial models with small batch size on V100. This will be addressed in a future release.
- An out-of-memory condition can occur in TensorFlow (TF1) 20.07 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the "XLA Best Practices" section of the *TensorFlow User Guide*, running XLA with the following environment variable opts in to that strategy: `TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false`
- There is a known performance regression of 15% compared to the 20.03 release when training the JoC V-Net Medical models with small batch size and fp32 data type on Turing GPUs. This will be addressed in a future release.
- There is a known performance regression of up to 60% when running inference using TF-TRT for SSD models with small batch size. This will be addressed in a future release.
- There is a known performance regression of up to 30% when training SSD models with fp32 data type on T4 GPUs. This will be addressed in a future release.
- There is a known issue where attempting to convert some models using TF-TRT produces an error "Failed to import metagraph". This issue is still under investigation and will be resolved in a future release.
- There is a known performance regression of 5-15% on the VAE-CF model when using the pip wheel compared to the corresponding NGC Docker container. This will be addressed in a future release.

2.34. TensorFlow Wheel Release 20.06

Dependencies of NVIDIA TensorFlow Wheel

This installation of the NVIDIA TensorFlow Wheel will include several other components from NVIDIA.

Table 33. TensorFlow Wheel compatibility with NVIDIA components

NVIDIA Product	Version
nvidia-cublas	11.1.0.213
nvidia-cublas-cupti	11.0.167
nvidia-cuda-nvcc	11.0.167
nvidia-cuda-nvrtc	11.0.167
nvidia-cuda-runtime	11.0.167
nvidia-cudnn	8.0.1.13
nvidia-cufft	10.1.3.191
nvidia-curand	10.2.0.191
nvidia-cusolver	10.4.0.191
nvidia-cusparse	11.0.0.191
nvidia-dali	0.22.0
nvidia-dali-tf-plugin	0.22.0
nvidia-horovod	0.19.1
nvidia-nccl	2.7.5
nvidia-tensorflow	1.15.2 + nv20.06
nvidia-tensorrt	7.1.2.8

Driver Requirements

Release 20.06 is based on NVIDIA CUDA 11.0.167, which requires NVIDIA Driver release 450.36. 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.

Software Requirements

The 20.06 release of TensorFlow Wheel requires the following software to be installed:

- Ubuntu 18.04 (64-bit) or Ubuntu 20.04 (64-bit)

Note: Ubuntu 18.04 defaults to Python 3.6; however, Ubuntu 20.04 requires the user to install Python 3.6.

- Python 3.6
- Pip 19.09 or later

Key Features and Enhancements

This TensorFlow Wheel release includes the following key features and enhancements.

- TensorFlow Wheel version 20.06 is based on TensorFlow 1.15.2.
- Integrated latest NVIDIA Deep Learning SDK to support NVIDIA A100 using CUDA 11 and cuDNN 8
- Improved NVTX annotations for XLA clusters for use with DLProf
- Improved XLA to avoid excessive recompilations
- Enhancements for Automatic Mixed Precision with einsum, 3D Convolutions, and list operations
- Improved 3D Convolutions to support NDHWC format
- Default TF32 support
- Ubuntu 18.04 with May 2020 updates

NVIDIA TensorFlow Wheel Versions

20.06 is the first release of the TensorFlow Wheel.

Tensor Core Examples

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

Each example model trains with mixed precision Tensor Cores on A100 and Volta architectures, therefore you can get results much faster than training without Tensor Cores. These models are tested against each NVIDIA Optimized Frameworks monthly container release to ensure consistent accuracy and performance over time.

- U-Net Medical model. The U-Net model is a convolutional neural network for 2D image segmentation. This repository contains a U-Net implementation as described in the

paper U-Net: Convolutional Networks for Biomedical Image Segmentation, without any alteration. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

- SSD320 v1.2 model. The SSD320 v1.2 model is based on the SSD: Single Shot MultiBox Detector paper, which describes an SSD as "a method for detecting objects in images using a single deep neural network". Our implementation is based on the existing model from the TensorFlow models repository. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- Neural Collaborative Filtering (NCF) model. The NCF model is a neural network that provides collaborative filtering based on implicit feedback, specifically, it provides product recommendations based on user and item interactions. The training data for this model should contain a sequence of user ID, item ID pairs indicating that the specified user has interacted with, for example, was given a rating to or clicked on, the specified item. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- BERT model. BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. This model is based on BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding paper. NVIDIA's BERT is an optimized version of Google's official implementation, leveraging mixed precision arithmetic and Tensor Cores on V100 GPUS for faster training times while maintaining target accuracy. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- U-Net Industrial Defect Segmentation model. This U-Net model is adapted from the original version of the U-Net model which is a convolutional auto-encoder for 2D image segmentation. U-Net was first introduced by Olaf Ronneberger, Philip Fischer, and Thomas Brox in the paper: U-Net: Convolutional Networks for Biomedical Image Segmentation. This work proposes a modified version of U-Net, called TinyUNet which performs efficiently and with very high accuracy on the industrial anomaly dataset DAGM2007. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- GNMT v2 model. The GNMT v2 model is similar to the one discussed in the Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation paper. The most important difference between the two models is in the attention mechanism. In our model, the output from the first LSTM layer of the decoder goes into the attention module, then the re-weighted context is concatenated with inputs to all subsequent LSTM layers in the decoder at the current timestep. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).
- ResNet-50 v1.5 model. The ResNet-50 v1.5 model is a modified version of the original ResNet-50 v1 model. The difference between v1 and v1.5 is in the bottleneck blocks which requires downsampling, for example, v1 has stride = 2 in the first 1x1 convolution, whereas v1.5 has stride = 2 in the 3x3 convolution. The following features were implemented in this model; data-parallel multi-GPU training with Horovod,

Tensor Cores (mixed precision) training, and static loss scaling for Tensor Cores (mixed precision) training. This model script is available on GitHub as well as NVIDIA GPU Cloud (NGC).

Known Issues

- An out-of-memory condition can occur in TensorFlow (TF1) 20.06 for some models (such as ResNet-50, and ResNext) when Horovod and XLA are both in use. In XLA in TensorFlow 20.05, we added an optimization that skips compiling a cluster the very first time it is executed, which can help avoid unnecessary recompilations for models with dynamic shapes. On the other hand, for models like ResNet-50, the preferred compilation strategy is to aggressively compile clusters, as compiled clusters are executed many times. Per the XLA Best Practices Guide, running XLA with the following environment variable opts in to that strategy:
  ```
  TF_XLA_FLAGS=--tf_xla_enable_lazy_compilation=false
  ```
- TensorFlow Wheel 20.06 NCF will not work until CuPy is updated to support CUDA 11.
- There is a known performance regression of 10 to 30% compared to the 20.03 release when training the JoC V-Net Medical and and U-Net Industrial models with small batch size on V100. This will be addressed in a future release.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

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