# TABLE OF CONTENTS

Chapter 1. Introduction................................................................. 1
Chapter 2. Accelerating TensorFlow 1.13.1 With TensorRT 5.1.2 RC Using The 19.03 Or 19.04 Container................................. 3
Chapter 3. Accelerating TensorFlow 1.13.0-rc0 With TensorRT 5.0.2 Using The 19.02 Container................................................. 4
Chapter 4. Accelerating TensorFlow 1.12.0 With TensorRT 5.0.2 Using The 18.12 Or 19.01 Container........................................... 6
Chapter 5. Accelerating TensorFlow 1.12.0-rc2 With TensorRT 5.0.2 Using The 18.11 Container.................................................. 8
Chapter 6. Accelerating TensorFlow 1.10 With TensorRT 5.0.0 RC Using The 18.09 Or 18.10 Container........................................... 10
Chapter 7. Accelerating TensorFlow 1.9 With TensorRT 4.0.1 Using The 18.08 Container...................................................... 13
Chapter 8. Accelerating TensorFlow 1.8 With TensorRT 4.0.1 Using The 18.06 Or 18.07 Container.................................................. 15
Chapter 9. Accelerating TensorFlow 1.7 With TensorRT 3.0.4 Using The 18.05 Container...................................................... 17
Chapter 1. INTRODUCTION

TensorFlow

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

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google’s Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks (DNNs) research. The system is general enough to be applicable in a wide variety of other domains, as well.

For visualizing TensorFlow results, the Docker® image also contains TensorBoard. TensorBoard is a suite of visualization tools. For example, you can view the training histories as well as what the model looks like.

For information about the optimizations and changes that have been made to TensorFlow, see the TensorFlow Deep Learning Frameworks Release Notes.

TensorRT

The core of NVIDIA TensorRT is a C++ library that facilitates high performance inference on NVIDIA graphics processing units (GPUs). TensorRT takes a trained network, which consists of a network definition and a set of trained parameters, and produces a highly optimized runtime engine which performs inference for that network.

You can describe a TensorRT network using a C++ or Python API, or you can import an existing Caffe, ONNX, or TensorFlow model using one of the provided parsers.

The TensorRT API includes import methods to help you express your trained deep learning models for TensorRT to optimize and run. TensorRT applies graph
optimizations, layer fusion, and finds the fastest implementation of that model leveraging a diverse collection of highly optimized kernels, and a runtime that you can use to execute this network in an inference context.

TensorRT includes an infrastructure that allows you to leverage the high speed mixed precision capabilities of Pascal, Volta, and Turing GPUs as an optional optimization.

For information about the optimizations and changes that have been made to TensorRT, see the TensorRT Release Notes. For specific TensorRT product documentation, see TensorRT documentation.
Chapter 2.
ACCELERATING TENSORFLOW 1.13.1 WITH TENSORRT 5.1.2 RC USING THE 19.03 OR 19.04 CONTAINER

These release notes are for accelerating TensorFlow 1.13.1 with TensorRT version 5.1.2 RC using the TensorFlow 19.03 or 19.04 container. For specific details about TensorRT, see the TensorRT 5.1.2 RC Release Notes.

Key Features And Enhancements
This release includes the following key features and enhancements.

TensorRT version
Integrated TensorRT 5.1.2 RC into TensorFlow. See the TensorRT 5.1.2 RC Release Notes for a full list of new features.

GitHub updates
Improved examples at GitHub: TF-TRT, including README files, build scripts, benchmark mode, ResNet models from TensorFlow official model zoo, etc...

Compatibility
- TensorFlow 1.13.1
- TensorFlow 19.04 container release
- TensorFlow 19.03 container release
- TensorRT 5.1.2 RC - ensure you are familiar with the TensorRT release notes for any known issues.

Limitations Of Accelerating TensorFlow With TensorRT
- TF-TRT is not supported in the TensorRT containers.
These release notes are for accelerating TensorFlow 1.13.0-rc0 with TensorRT version 5.0.2 using the TensorFlow 19.02 container. For specific details about TensorRT, see the TensorRT 5.0.2 Release Notes.

Key Features And Enhancements

This release includes the following key features and enhancements.

TensorFlow operators

The following operators can now be converted from TensorFlow to TensorRT:
- ExpandDims, Reshape, Sigmoid, Sqrt, Square, Squeeze, StridedSlice and Tanh.
  For more information, see Supported Ops.

INT8 Pre-quantization

You can manually insert quantization ranges (generated during quantization-aware training) to the graph, and then TF-TRT can use them during INT8 inference. That means calibration is not required with this feature. For more information, see INT8 Quantization.

Compatibility

- TensorFlow 1.13.0-rc0
- TensorFlow 19.02 container release
- TensorRT 5.0.2. Ensure you are familiar with the TensorRT 5.0.2 Release Notes for any known issues.
Limitations Of Accelerating TensorFlow With TensorRT

- TF-TRT is not supported in the TensorRT containers.

Deprecated Features

- Support for TensorRT 3 has been removed.
Chapter 4.
ACCELERATING TENSORFLOW 1.12.0 WITH TENSORRT 5.0.2 USING THE 18.12 OR 19.01 CONTAINER

These release notes are for accelerating TensorFlow 1.12.0 with TensorRT version 5.0.2 using either the TensorFlow 18.12 or TensorFlow 19.01 container. For specific details about TensorRT, see the TensorRT 5.0.2 Release Notes.

Compatibility

- TensorFlow 1.12.0
- TensorFlow 19.01 container release
- TensorFlow 18.12 container release
- TensorRT 5.0.2. Ensure you are familiar with the TensorRT 5.0.2 Release Notes for any known issues.

Using TensorFlow 1.12.0 With TensorRT 5.0.2

- The image-classification examples were moved from /opt/tensorflow/nvidia-examples/inference/image-classification/scripts to https://github.com/tensorflow/tensorrt/tree/master/tftrt/examples/image-classification.

Deprecated Features

- Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a
production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

- The `check_accuracy.py` script, used to check whether the accuracy generated by the example matches with the expectation, was removed from the example. Refer to the published accuracy numbers to verify whether your generated accuracy numbers match with the expectation.
These release notes are for accelerating TensorFlow 1.12.0-rc2 with TensorRT version 5.0.2 using the TensorFlow 18.11 container. For specific details about TensorRT, see the TensorRT 5.0.2 Release Notes.

**Key Features and Enhancements**
This release includes the following key features and enhancements.

- Added support for dilated convolution.
- Fixed a bug in the **Identity** op.
- Fixed a bug in the **Relu6** op.
- Support added to allow empty const tensor.
- Added object detection example to [nvidia-examples/inference](https://github.com/NVIDIA/nvidia-examples/tree/main/inference).

**Compatibility**

- [TensorFlow 1.12.0-rc2](https://www.tensorflow.org/)  
- [TensorFlow 18.11 container release](https://github.com/NVIDIA/nvidia-examples/tree/main/inference)  
- TensorRT 5.0.2. Ensure you are familiar with the [TensorRT 5.0.2 Release Notes](https://docs.nvidia.com/deeplearning/tensorrt/5.0.2/index.html) for any known issues.

**Deprecated Features**

- Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across
platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

**Known Issues**

- In the TF-TRT API, the `minimum_segment_size` argument default value is 3. In the image classification examples under `nvidia-examples/inference`, we define a command line argument for `minimum_segment_size` which has its own default value. In 18.10, the default value was 7 and in 18.11 we changed it to 2. Smaller values for this argument would cause to convert more TensorFlow nodes to TensorRT which typically should improve the performance, however, we have observed cases where the performance gets worse. In particular, Resnet-50 with smaller batch sizes gets slower with `minimum_segment_size=2` comparing to `minimum_segment_size=7`. 
These release notes are for accelerating TensorFlow 1.10 with TensorRT version 5.0.0 Release Candidate (RC) using the TensorFlow 18.09 or TensorFlow 18.10 container. For specific details about TensorRT, see the TensorRT 5.0.0 RC Release Notes.

**Key Features and Enhancements**
This release includes the following key features and enhancements.

- New examples at `nvidia-examples/tftrt` with good accuracy and performance.
- Built TF-TRT with TensorRT 5.0.0 which introduces the new TensorRT APIs into TF-TRT.
- In 18.10, we added support for the TensorFlow operator RELU6 (using `Relu6(x) = min(Relu(x), 6)`).
- In 18.10, we made improvements in the image classification example, such as bug fixes and using the `dynamic_op` feature.

**Compatibility**

- TensorFlow 1.10
- TensorFlow 18.09 container release or TensorFlow 18.10 container release
- TensorRT 5.0.0 RC. Ensure you are familiar with the TensorRT 5.0.0 RC release notes for any known issues.
Limitations Of Accelerating TensorFlow With TensorRT

There are some limitations you may experience after accelerating TensorFlow 1.10 with TensorRT 5.0.0 RC, such as:

- Not all the new TensorRT 5.0.0 features are supported yet in TF-TRT including INT8 quantization ranges and the plugins registry.

- We have only tested image classification models with TF-TRT including the ones we have provided in our examples inside the container (nvidia-examples/tftrt). This means object detection, translation (convolutional and recurrent based) are not yet supported due to either functionality or performance limitations.

- TF-TRT has an implementation of optimizing the TensorFlow graph by specifying appropriate TensorFlow session arguments without using the Python TF-TRT API (create_inference_graph), however, we have not thoroughly tested this functionality yet, therefore, we don’t support it.

- In 18.09, TF-TRT has an implementation of the dynamic conversion of a TensorFlow graph, but we have not thoroughly tested this functionality yet, therefore, we don’t support it.

Deprecated Features

- Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

Known Issues

- Running inference with batch sizes larger than the maximum batch size is not supported by TensorRT.

- Due to certain logs (errors or warnings) of TF-TRT, they could be misleading and point to the TensorRT graph as broken while it’s not. It is recommended to check whether there is any TensorRT op in the graph (the type of op is TRTEngineOp). If there is not TensorRT ops in the graph, that means no conversion has happened and the inference should fall back to the native TensorFlow. Currently, the best way to verify whether a frozen graph resulting from the conversion is not broken is to run inference on it and check the accuracy of the results.
There are operators that are not supported by either TensorRT or the conversion algorithm. The convertor is supposed to skip these ops but this skip may not happen properly due to bugs. One way to get around this problem is to increase the value of the `minimum_segment_size` parameter and hope that the subgraphs that contain those ops are too small and remain out of the conversion.

We have observed functionality problems in optimizing:

- NASNet models with TF-TRT in FP16 precision mode.
- ResNet, MobileNet, and NASNet models with TF-TRT in INT8 precision mode.

TF-TRT cannot optimize certain models such as ResNet in INT8 precision mode because of a lacking feature in TensorRT regarding the dimensionality of tensors. Usually, increasing the value of `minimum_segment_size` is a workaround by removing those unsupported dimensions out of the TensorRT sub-graph.

TF-TRT doesn't work with TensorFlow Lite due to a TensorRT bug that causes Flatbuffer symbols to be exposed. This means you cannot import both `tf.contrib.tensorrt` and `tf.lite` in the same process.

We have observed a bit low accuracy on image classification models with TF-TRT on Jetson AGX Xavier.

INT8 calibration on `mobilenet_v1` and `mobilenet_v2` using TF-TRT fails if the calibration dataset has only one element.
Chapter 7.
ACCELERATING TENSORFLOW 1.9 WITH TENSORRT 4.0.1 USING THE 18.08 CONTAINER

These release notes are for accelerating TensorFlow 1.9 with TensorRT version 4.0.1 using the TensorFlow 18.08 container. For specific details about TensorRT, see the TensorRT 4.0.1 Release Notes.

Key Features and Enhancements
This release includes the following key features and enhancements.

‣ TensorRT conversion has been integrated into optimization pass. The `tensorflow/contrib/tensorrt/test/test_tftrt.py` script has an example showing the use of optimization pass.

Compatibility

‣ TensorFlow 1.9
‣ TensorFlow 18.08 container release
‣ TensorRT 4.0.1. Ensure you are familiar with the TensorRT 4.0.1 release notes for any known issues.

Limitations Of Accelerating TensorFlow With TensorRT
There are some limitations you may experience after accelerating TensorFlow 1.9 with TensorRT 4.0.1, such as:

‣ TensorRT conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.
• Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.

• Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.

• Current optimization pass does not support INT8 yet.

Known Issues

• Input tensors are required to have rank 4 for quantization mode (INT8 precision).
Chapter 8.
ACCELERATING TENSORFLOW 1.8 WITH TENSORRT 4.0.1 USING THE 18.06 OR 18.07 CONTAINER

These release notes are for accelerating TensorFlow 1.8 with TensorRT version 4.0.1 using either the TensorFlow TensorFlow 18.06 or TensorFlow 18.07 container. For specific details about TensorRT, see the TensorRT 4.0.1 Release Notes.

Key Features and Enhancements
This release includes the following key features and enhancements.

‣ Added TensorRT 4.0 API support with extended layer support. This support includes the FullyConnected layer and **BatchedMatMul** op.

‣ Resource management added, where memory allocation is uniformly managed by TensorFlow.

‣ Bug fixes and better error handling in conversion.

Compatibility

‣ **TensorFlow 1.8**

‣ **TensorFlow 18.07 container release** or **TensorFlow 18.06 container release**

‣ TensorRT 4.0.1. Ensure you are familiar with the **TensorRT 4.0.1 release notes** for any known issues.

Limitations Of Accelerating TensorFlow With TensorRT

There are some limitations you may experience after accelerating TensorFlow 1.8 with TensorRT 4.0.1, such as:
TensorRT conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.

Batchsize for converted TensorRT engines are fixed at conversion time. Inference can only run with batchsize smaller than the specified number.

Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.

**Deprecated Features**

In the 18.05 container, you need to create a TensorFlow session with the `per_process_gpu_memory_fraction` option. With the resource management fully integrated, you no longer need to reserve GPU memory from TensorFlow. Therefore, the option is not necessary for mixed TensorFlow-TensorRT (TF-TRT) model.

**Known Issues**

Input tensors are required to have rank 4 for quantization mode (INT8 precision).
These release notes are for accelerating TensorFlow 1.7 with TensorRT version 3.0.4 using the TensorFlow 18.05 container. For specific details about TensorRT, see the TensorRT 3.0.4 Release Notes.

**Attention** Support for accelerating TensorFlow with TensorRT 3.x will be removed in a future release (likely TensorFlow 1.13). The generated plan files are not portable across platforms or TensorRT versions. Plans are specific to the exact GPU model they were built on (in addition to platforms and the TensorRT version) and must be retargeted to the specific GPU in case you want to run them on a different GPU. Therefore, models that were accelerated using TensorRT 3.x will no longer run. If you have a production model that was accelerated with TensorRT 3.x, you will need to convert your model with TensorRT 4.x or later again.

For more information, see the Note in Serializing A Model In C++ or Serializing A Model In Python.

**Key Features and Enhancements**
This release includes the following key features and enhancements.

- TensorRT backend accelerates inference performance for frozen TensorFlow models.
- Automatic segmenter that recognizes TensorRT compatible subgraphs and converts them into TensorRT engines. TensorRT engines are wrapped with TensorFlow custom ops that moves the execution of the subgraph to TensorRT backend for optimized performance, while fall back to TensorFlow for non-TensorRT compatible ops.
Supported networks are slim classification networks including ResNet, VGG, and Inception.

Mixed precision and quantization are supported.

**Compatibility**

- TensorFlow 1.7
- TensorFlow 18.05 container release
- TensorRT 3.0.4. Ensure you are familiar with the TensorRT 3.0.4 release notes for any known issues.

**Limitations Of Accelerating TensorFlow With TensorRT**

There are some limitations you may experience after accelerating TensorFlow 1.7 with TensorRT 3.0.4, such as:

- Conversion relies on static shape inference, where the frozen graph should provide explicit dimension on all ranks other than the first batch dimension.

- Batch size for converted TensorRT engines are fixed at conversion time. Inference can only run with batch size smaller than the specified number.

- Current supported models are limited to CNNs. Object detection models and RNNs are not yet supported.

- Resource management is not integrated, therefore, ensure you limit the memory claimed by TensorFlow in order for TensorRT to acquire the necessary resource. To limit the memory, use `setting per_process_gpu_memory_fraction to < 1.0` and pass it to session creation, for example:

```python
gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.333) sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options))
```

**Known Issues**

- The TensorRT engine only accepts input tensor with `rank == 4 ..`
Notice

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS GUIDE IS PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the product described in this guide shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do 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 guide. NVIDIA does not accept any 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 guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, DGX, DGX-1, DGX-2, and DGX Station are trademarks and/or registered trademarks of NVIDIA Corporation in the United States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2019 NVIDIA Corporation. All rights reserved.

www.nvidia.com