NVIDIA TensorRT Optimizations and Quantizations

3^{rd} Paper Study | 2022.08.03 Wed.

홍 세 현 Hong Sehyun

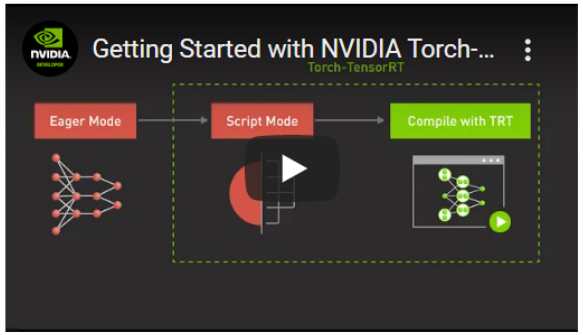
STRADVISION

Outline

1. TensorRT Introduction
2. TensorRT Optimizations
3. TensorRT Quantizations
4. Reference and Q & A

TensorRT Introduction

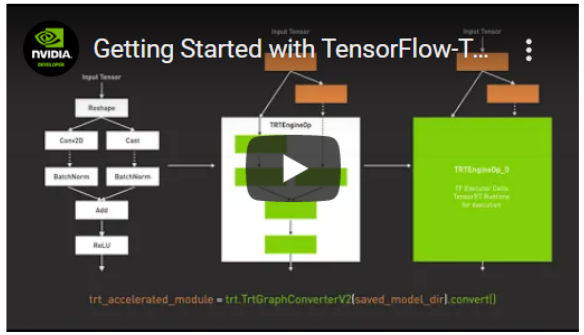
NVIDIA TensorRT



PyTorch

Accelerate PyTorch models using the new Torch-TensorRT Integration with just one line of code. Get 6X faster inference using the TensorRT optimizations in a familiar PyTorch environment.

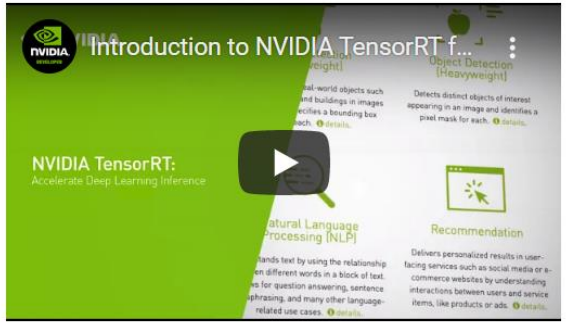
LEARN MORE >



TensorFlow

TensorRT and TensorFlow are tightly integrated so you get the flexibility of TensorFlow with the powerful optimizations of TensorRT like 6X the performance with one line of code.

LEARN MORE >



ONNX

TensorRT provides an ONNX parser so you can easily import ONNX models from popular frameworks into TensorRT. It's also integrated with ONNX Runtime, providing an easy way to achieve high-performance inference in the ONNX format.

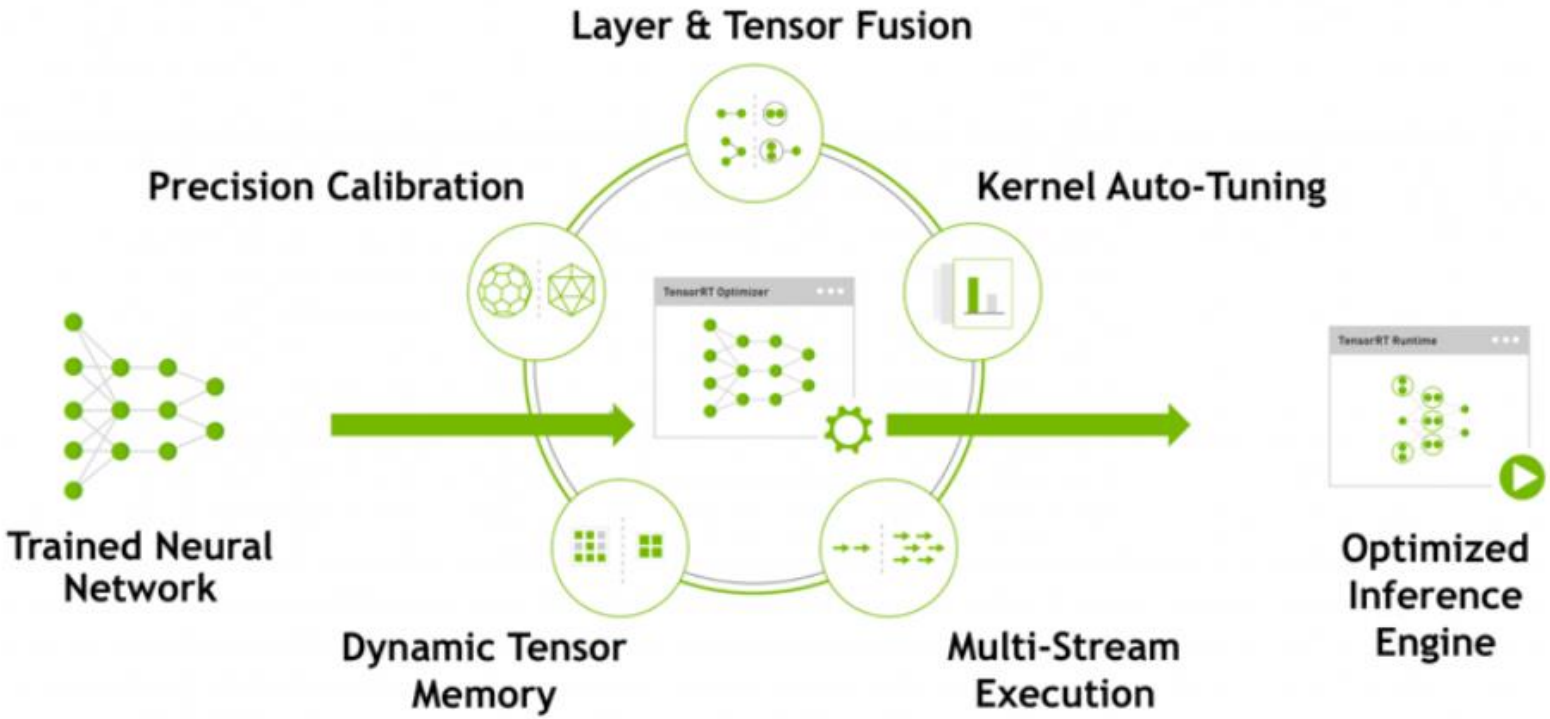
LEARN MORE >

Matlab

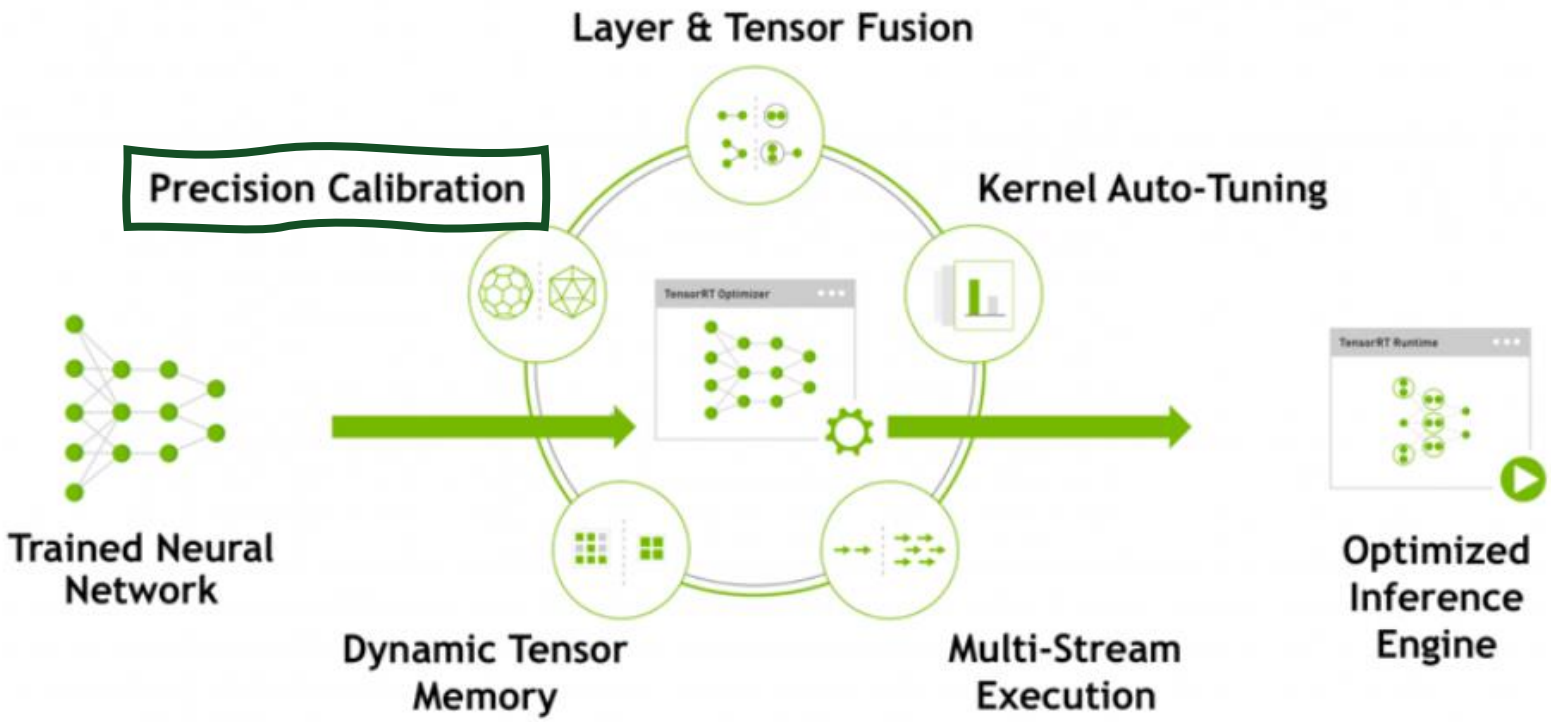
MATLAB is integrated with TensorRT through GPU Coder so you can automatically generate high performance inference engines for NVIDIA Jetson™, NVIDIA DRIVE®, and data center platforms.

LEARN MORE >

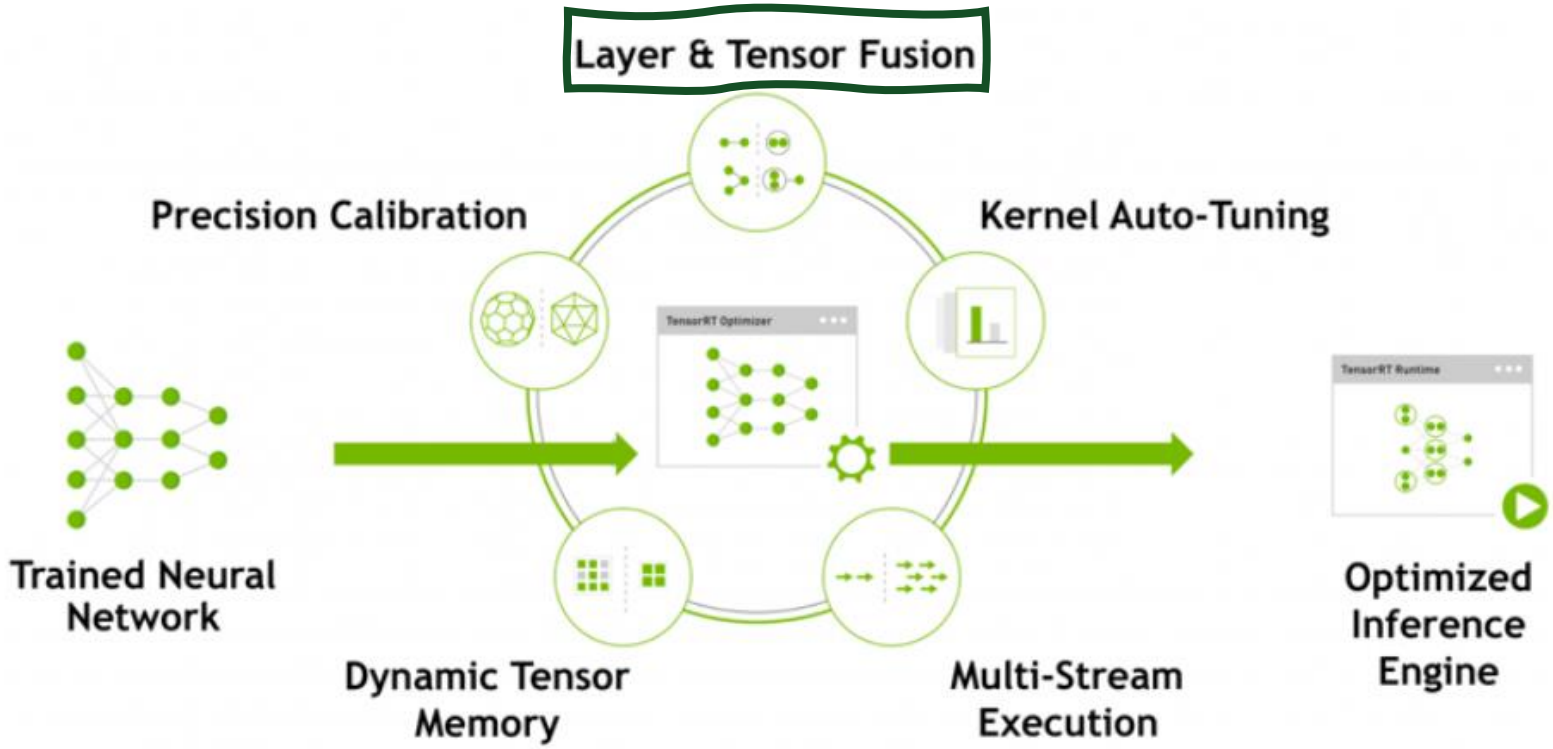
TensorRT Introduction



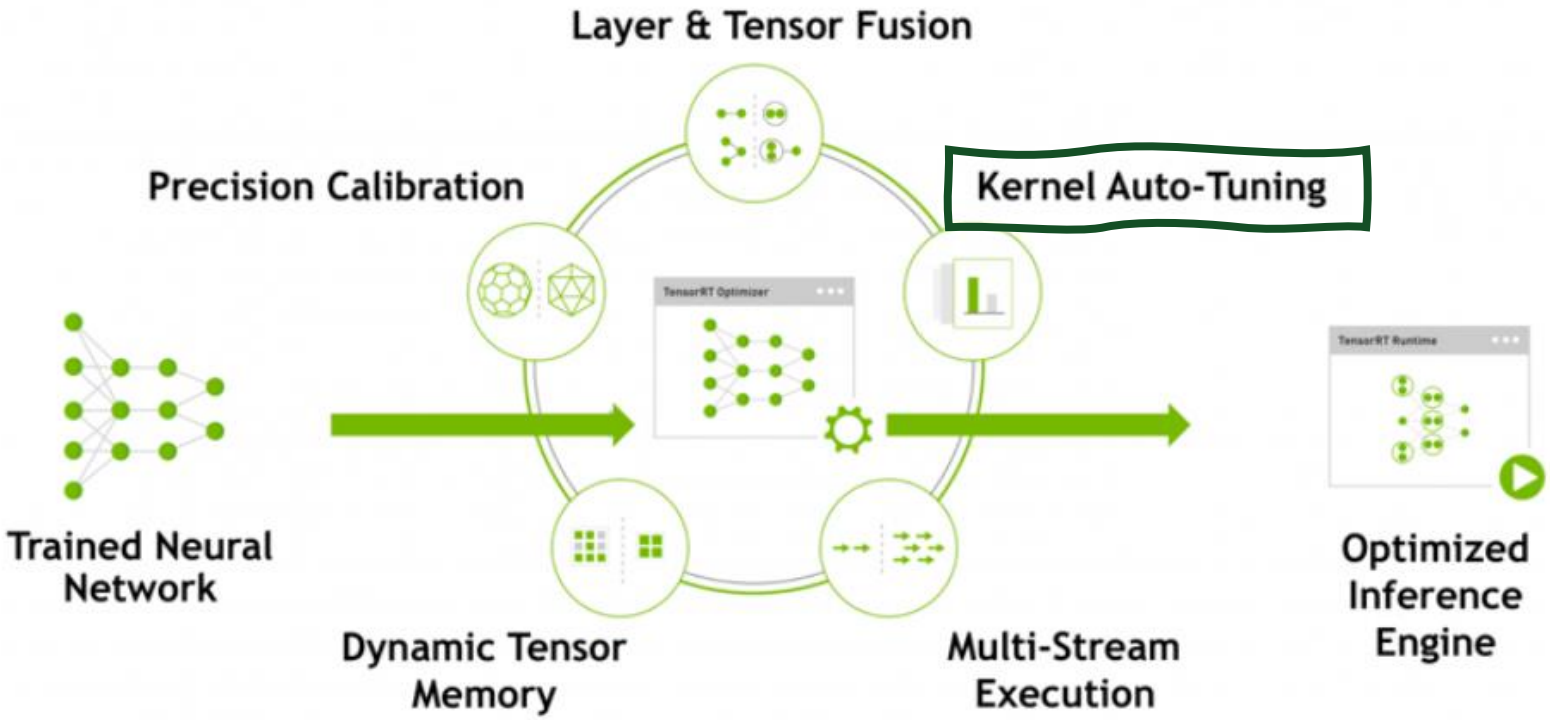
TensorRT Introduction



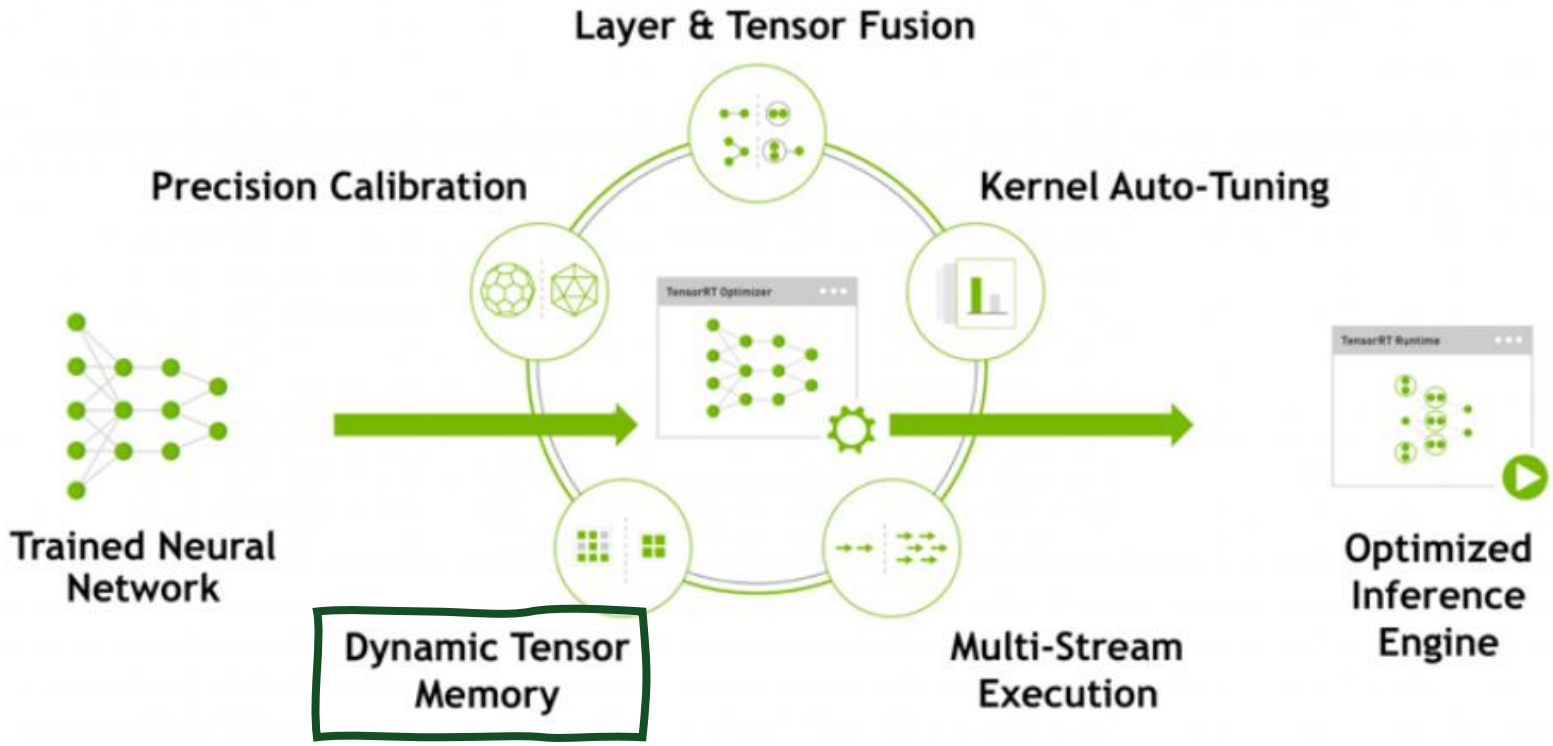
TensorRT Introduction



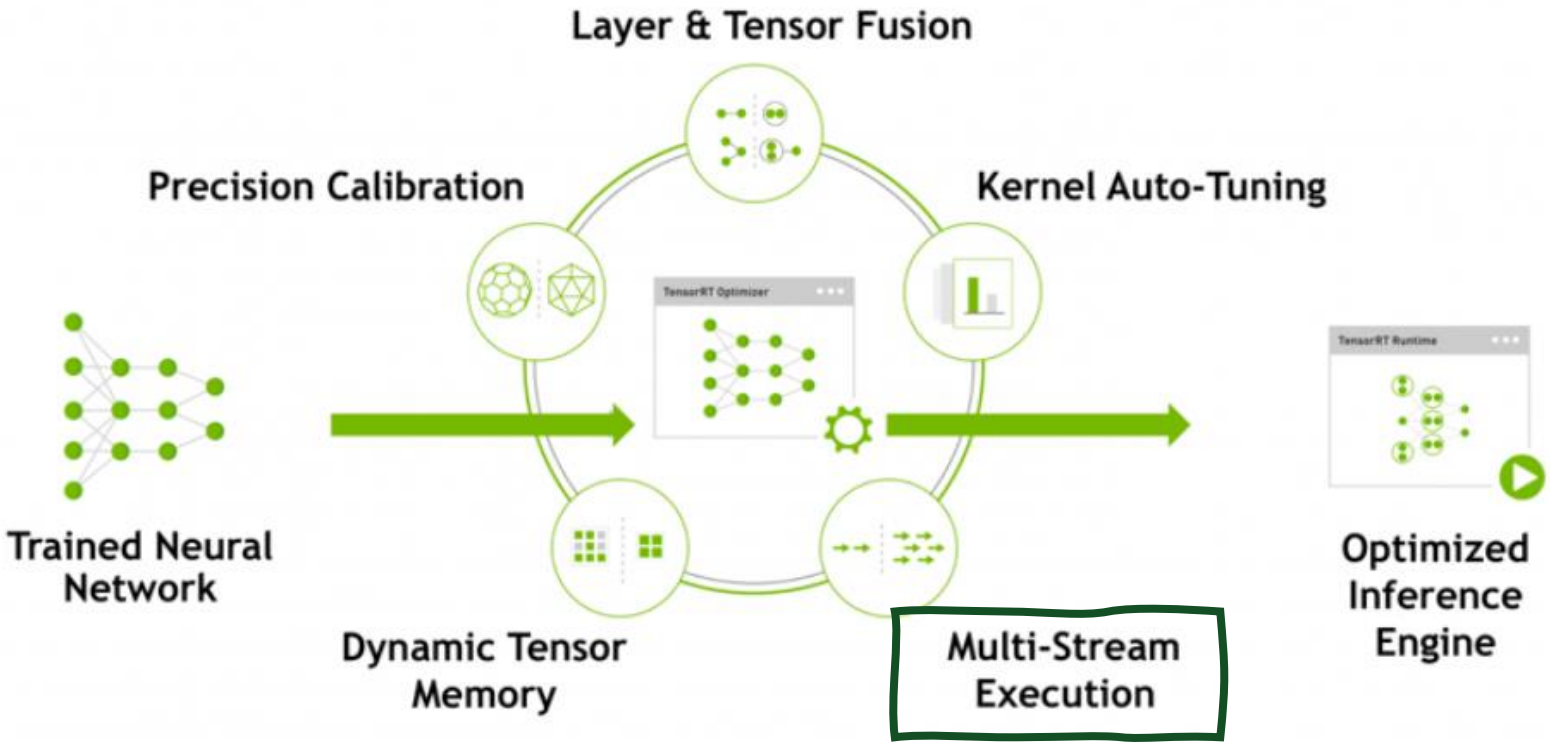
TensorRT Introduction



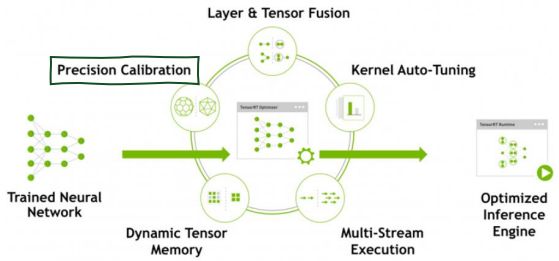
TensorRT Introduction

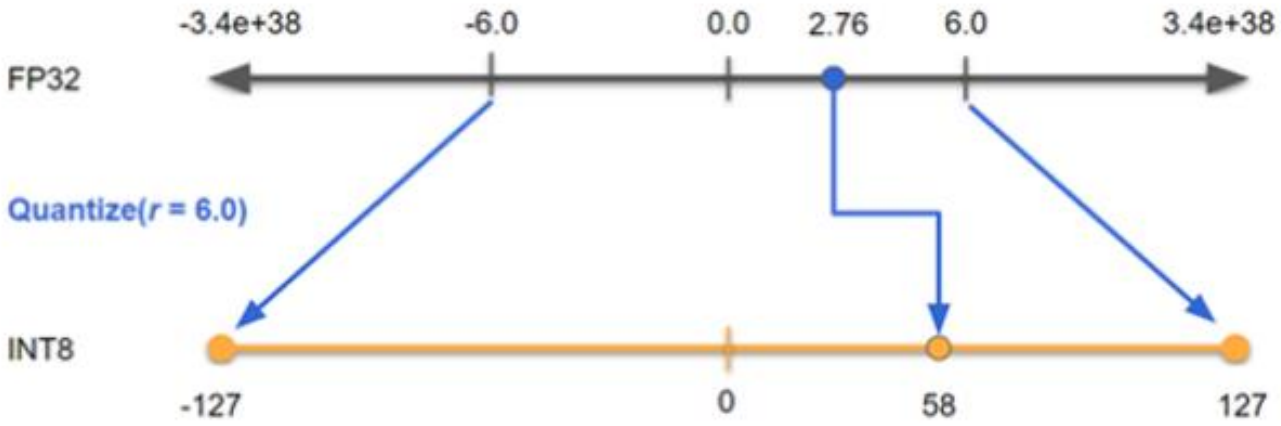


TensorRT Introduction



TensorRT Optimizations





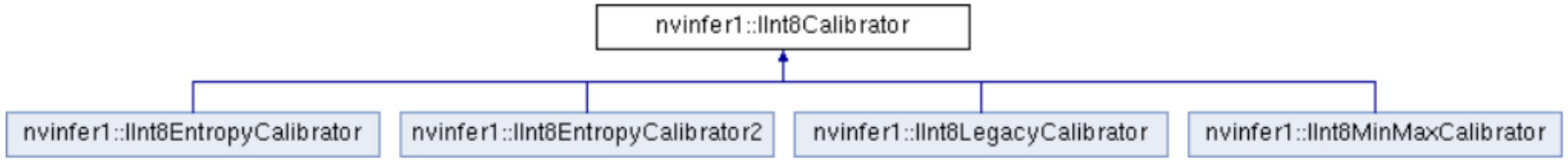
$$Quantize(x, r) = round(s * clip(x, -r, r))$$
where $s = 127 / r$

Symmetric linear quantization

x : Input
r : Floating point range
s : Scaling factor

TensorRT Optimizations

Inheritance diagram for nvinfer1::IInt8Calibrator:

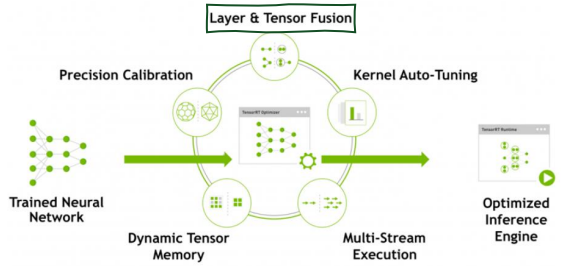


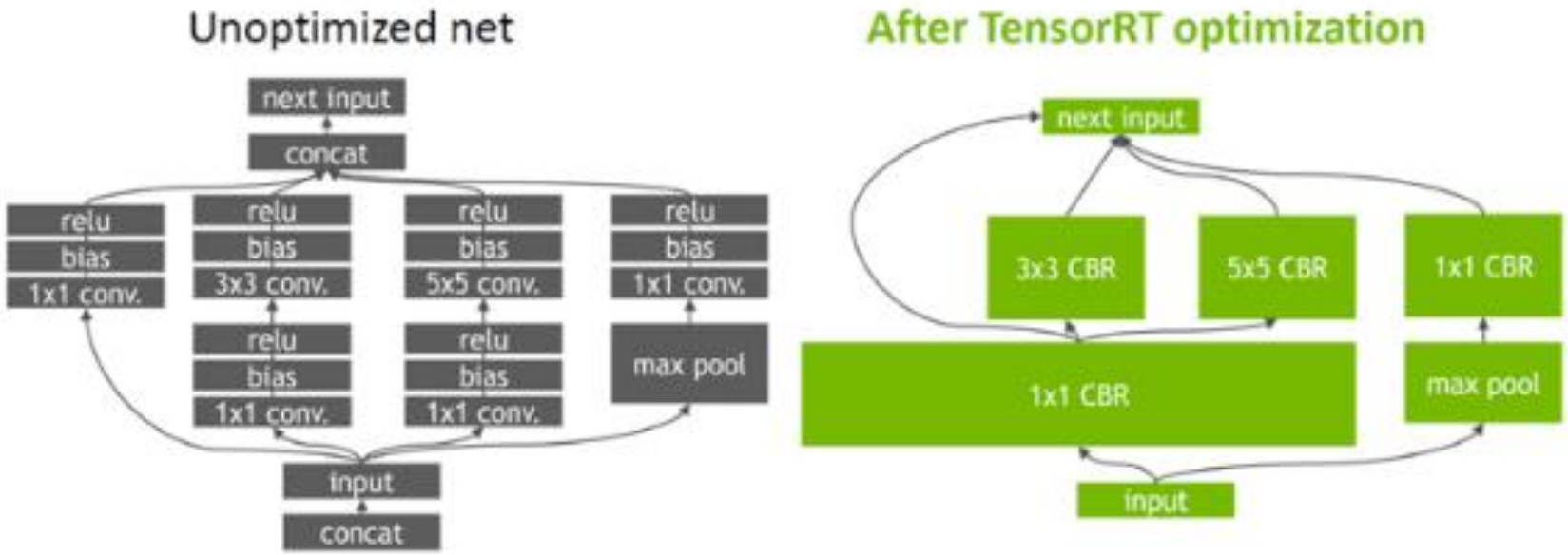
Saturate above |T| to 127

In general,
Low-bit quantization occurs
Significant accuracy loss.

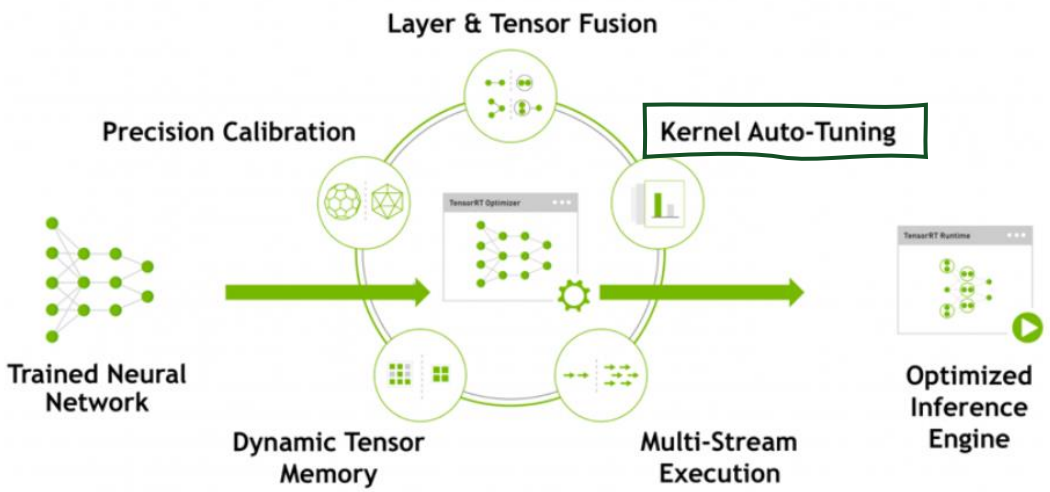
Use calibration to get proper |T|
(To minimize information loss,
find value which shows min-entropy
on quantization)

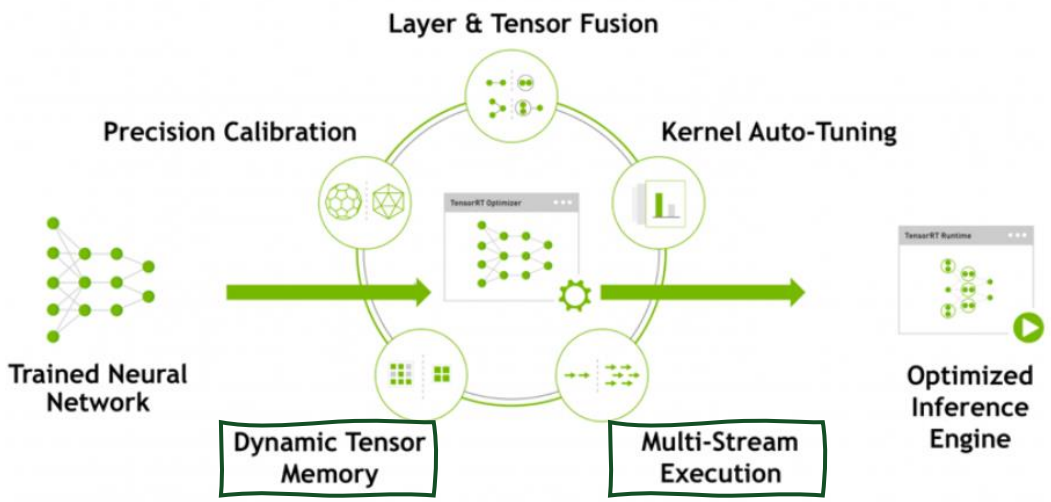
TensorRT Optimizations



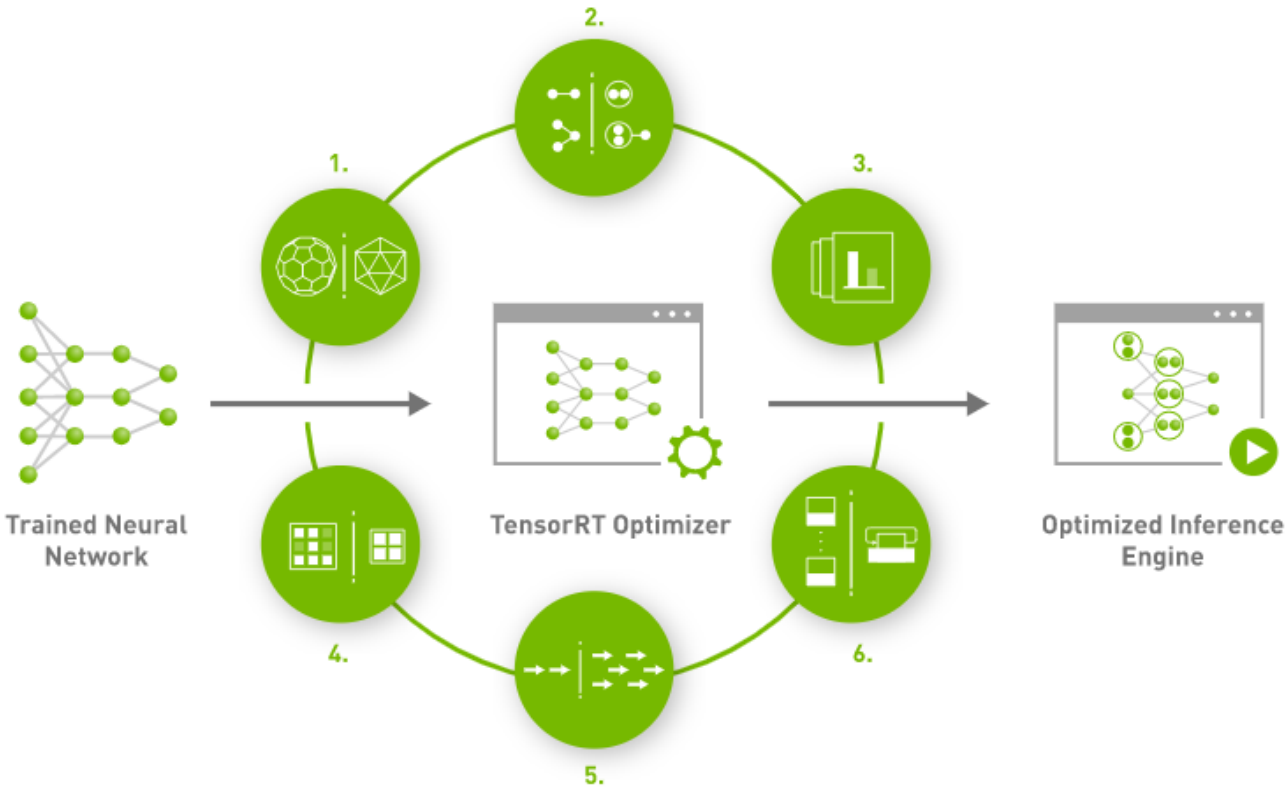


TensorRT Optimizations





TensorRT Optimizations



1. **Weight & Activation Precision Calibration**
   Maximizes throughput by quantizing models to INT8 while preserving accuracy

2. **Layer & Tensor Fusion**
   Optimizes use of GPU memory and bandwidth by fusing nodes in a kernel

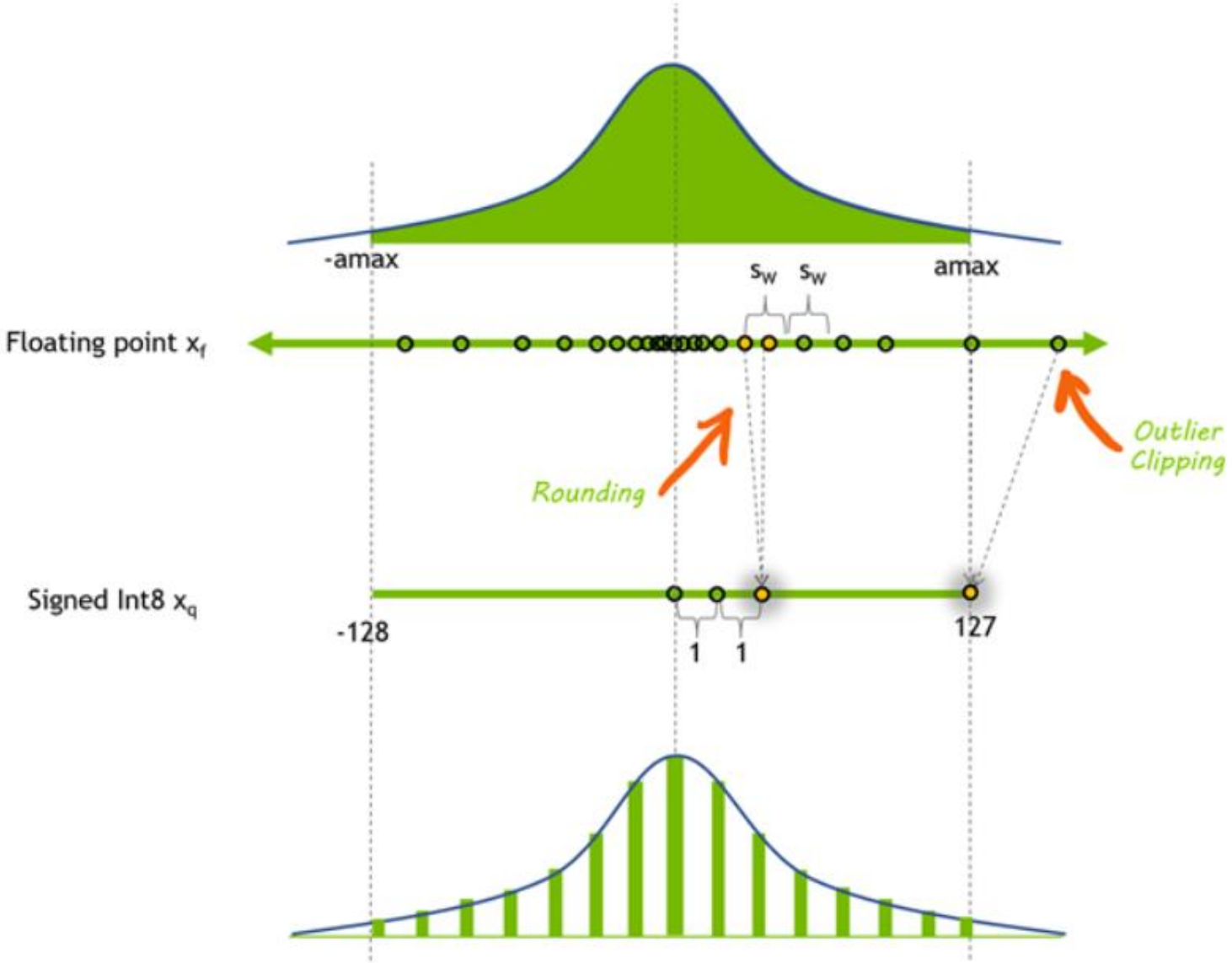
3. **Kernel Auto-Tuning**
   Selects best data layers and algorithms based on target GPU platform

4. **Dynamic Tensor Memory**
   Minimizes memory footprint and re-uses memory for tensors efficiently

5. **Multi-Stream Execution**
   Scalable design to process multiple input streams in parallel

6. **Time Fusion**
   Optimizes recurrent neural networks over time steps with dynamically generated kernels

TensorRT Quantizations

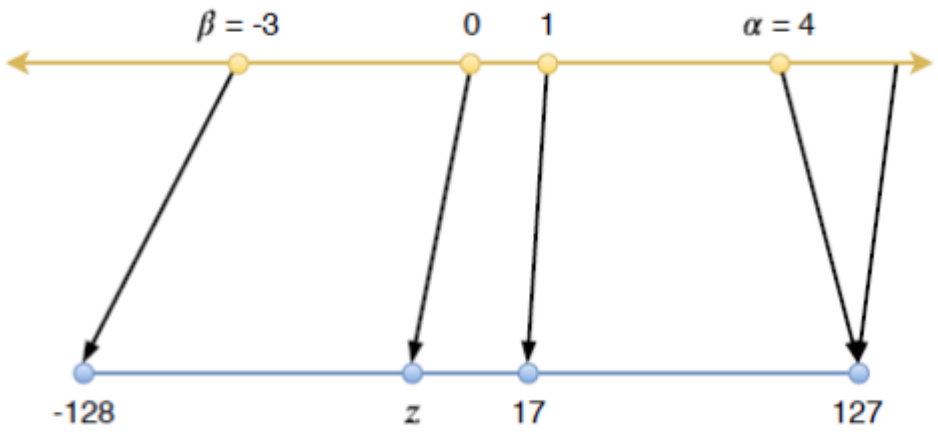


$$amax = max(abs(x_f))$$

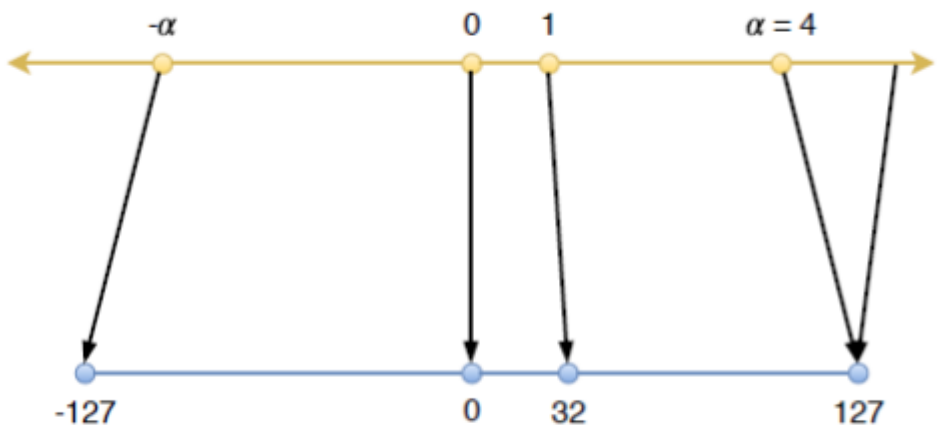
$$scale = (2 * amax)/256$$

$$x_q = Clip(Round(x_f/scale))$$

TensorRT Quantizations



(a) Affine quantization



(b) Scale quantization

TensorRT Quantizations

7.2. Setting Dynamic Range

TensorRT provides APIs to set *dynamic range* (the range that must be represented by the quantized tensor) directly, to support implicit quantization where these values have been calculated outside TensorRT.

The API allows setting the dynamic range for a tensor using minimum and maximum values. Since TensorRT currently supports only symmetric range, the scale is calculated using `max(abs(min_float), abs(max_float))`. Note that when `abs(min_float) != abs(max_float)`, TensorRT uses a larger dynamic-range than configured, which may increase the rounding error.

Dynamic range is needed for all floating-point inputs and outputs of an operation that will execute in INT8.

You can set the dynamic range for a tensor as follows:

C++

```
tensor->setDynamicRange(min_float, max_float);
```

Python

```
tensor.dynamic_range = (min_float, max_float)
```

TensorRT Quantizations

PTQ (Post Training Quantization)

`IInt8EntropyCalibrator2`

Entropy calibration chooses the tensor's scale factor to optimize the quantized tensor's information-theoretic content, and usually suppresses outliers in the distribution. This is the current and recommended entropy calibrator and is required for DLA. Calibration happens before Layer fusion by default. Calibration batch size may impact the final result. It is recommended for CNN-based networks.

7.3.2. Calibration Using Python

The following steps illustrate how to create an INT8 calibrator object using the Python API.

Procedure

1. Import TensorRT:
```
import tensorrt as trt
```
2. Similar to test/validation datasets, use a set of input files as a calibration dataset. Make sure that the calibration files are representative of the overall inference data files. For TensorRT to use the calibration files, you must create a `batchstream` object. A `batchstream` object is used to configure the calibrator.
```
NUM_IMAGES_PER_BATCH = 5
batchstream = ImageBatchStream(NUM_IMAGES_PER_BATCH, calibration_files)
```
3. Create an `Int8_calibrator` object with input nodes names and batch stream:
```
Int8_calibrator = EntropyCalibrator(["input_node_name"], batchstream)
```
4. Set INT8 mode and INT8 calibrator:
```
config.set_flag(trt.BuilderFlag.INT8)
config.int8_calibrator = Int8_calibrator
```

PTQ (Post Training Quantization)

7.3.3. Quantization Noise Reduction

For networks with implicit quantization, TensorRT attempts to reduce quantization noise in the output by forcing some layers near the network outputs to run in FP32, even if INT8 implementations are available.

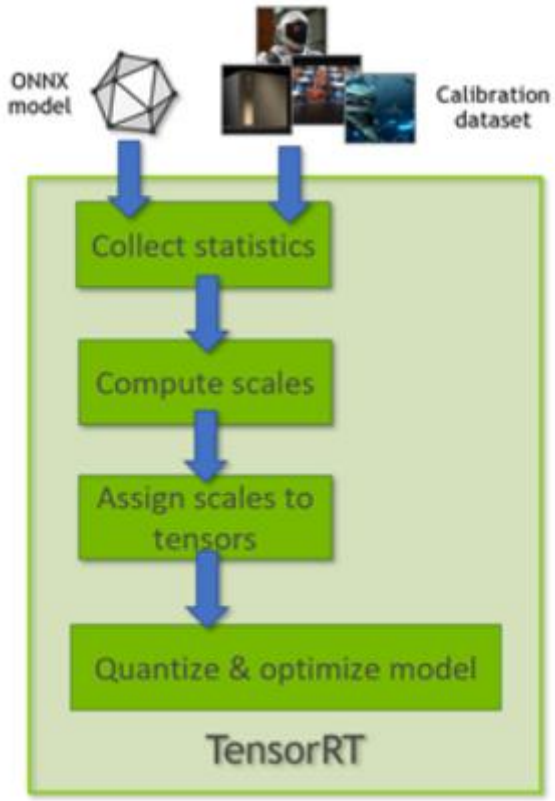
The heuristic attempts to ensure that INT8 quantization is smoothed out by summation of multiple quantized values. Layers considered to be "smoothing layers" are convolution, deconvolution, a fully connected layer, or matrix multiplication before reaching the network output. For example, if a network consists of a series of (convolution + activation + shuffle) subgraphs and the network output has type FP32, the last convolution will output FP32 precision, even if INT8 is allowed and faster.

The heuristic does not apply in the following scenarios:

- The network output has type INT8.
- An operation on the path (inclusively) from the last smoothing layer to the output is constrained by `ILayer::setOutputType` or `ILayer::setPrecision` to output INT8.
- There is no smoothing layer with a path to the output, or said that path has an intervening plug-in layer.
- The network uses explicit quantization.

TensorRT Quantizations

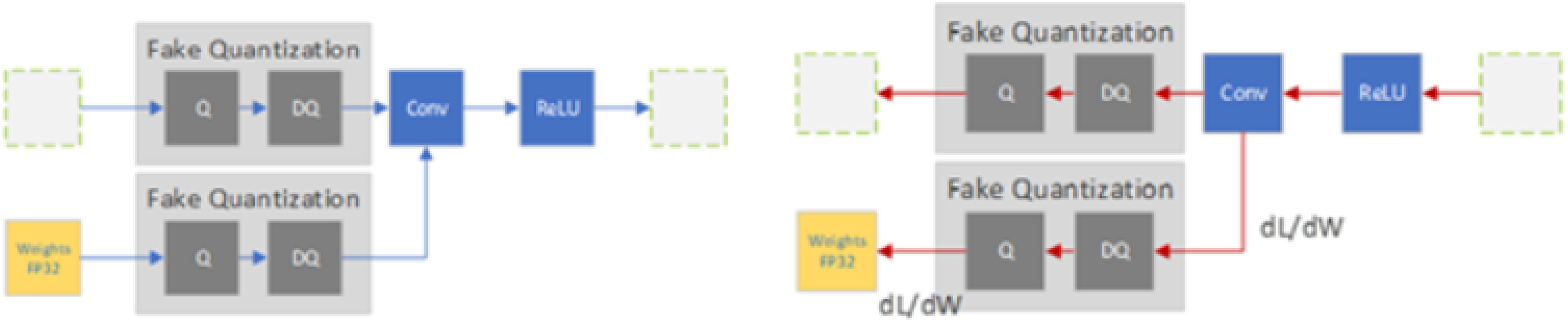
PTQ (Post Training Quantization)



Full TensorRT PTQ workflow

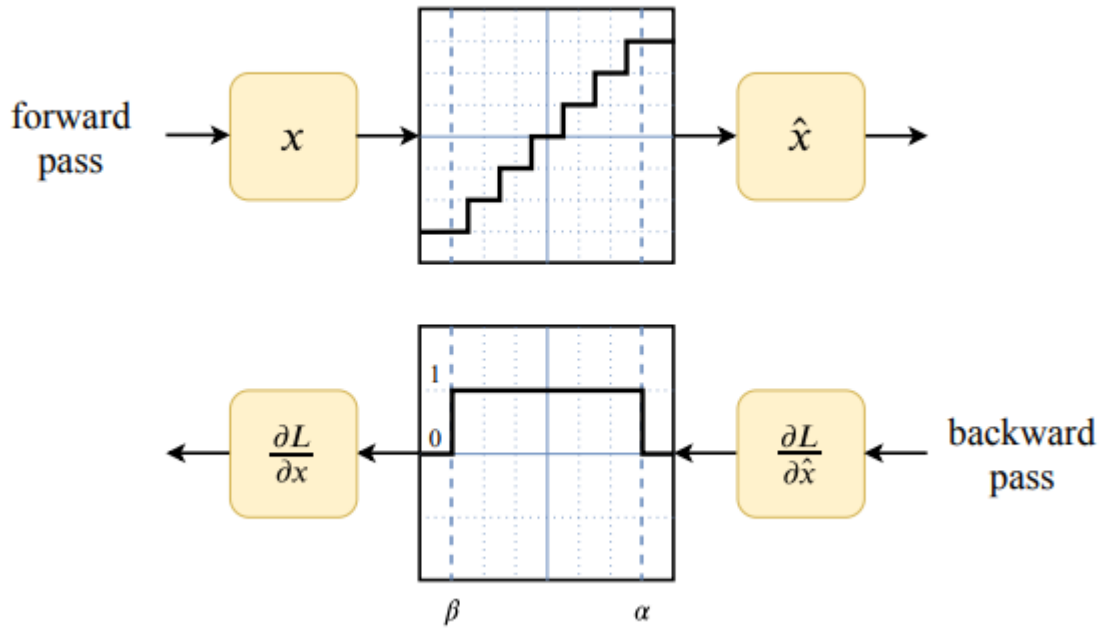
TensorRT Quantizations

QAT (Quantization Aware Training)



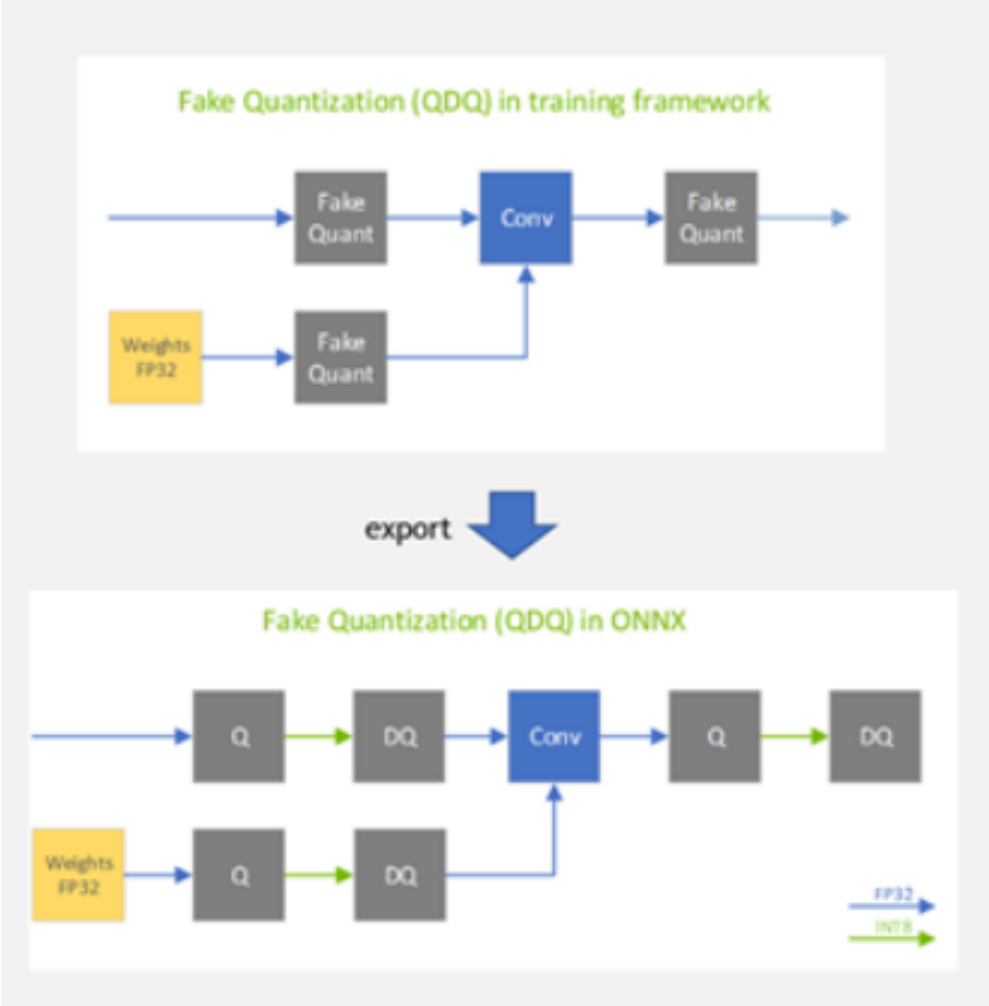
QAT (Quantization Aware Training)

Straight-through Estimator(STE)[

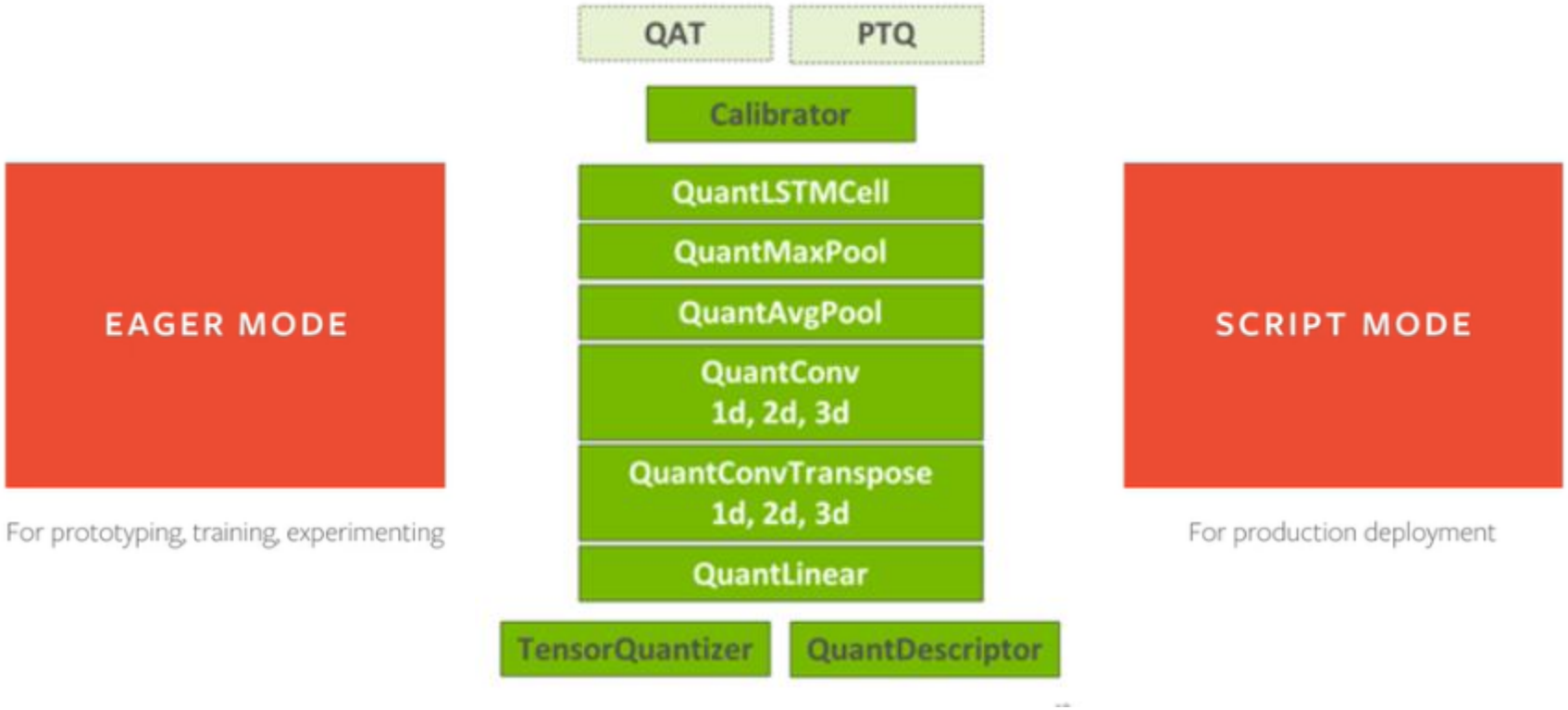


$$\frac{d\hat{x}}{dx} = \begin{cases} 0, & y < \beta \\ 1, & \beta \leq y \leq \alpha \\ 0, & y > \alpha \end{cases}$$

TensorRT Quantizations



TensorRT Quantizations



References

NVIDIA TensorRT – Inference 최적화 및 가속화를 위한 NVIDIA의 Toolkit

How to Get Started with TensorRT

NVIDIA TensorRT Introduction

TensorRT 8.4.2 Documentation

NVIDIA TensorRT Documentation

NVIDIA TensorRT SSD Object Detection Demo

NVIDIA TensorRT Quantization Aware Training Article

Q & A

Thank you ☺