Introduction to TensorRT

Lijian Chen
12 Apr 2019

Overview

- How to measure deep learning inference ?
- What is TensorRT ?
- How does it work ?
- How to use TensorRT ?
- How to use TensorRT Inference Server ?

Measure Inference

Accuracy

Latency

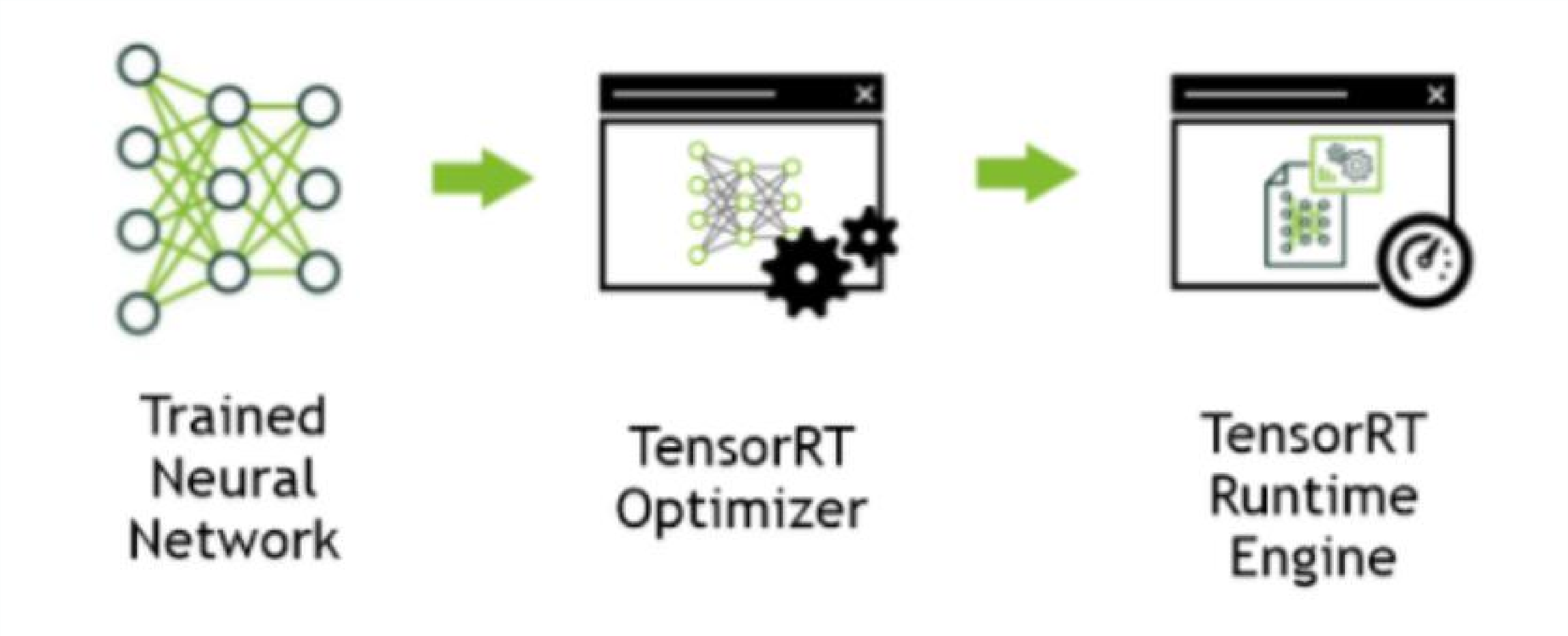
Throughput

Memory Usage

Efficiency

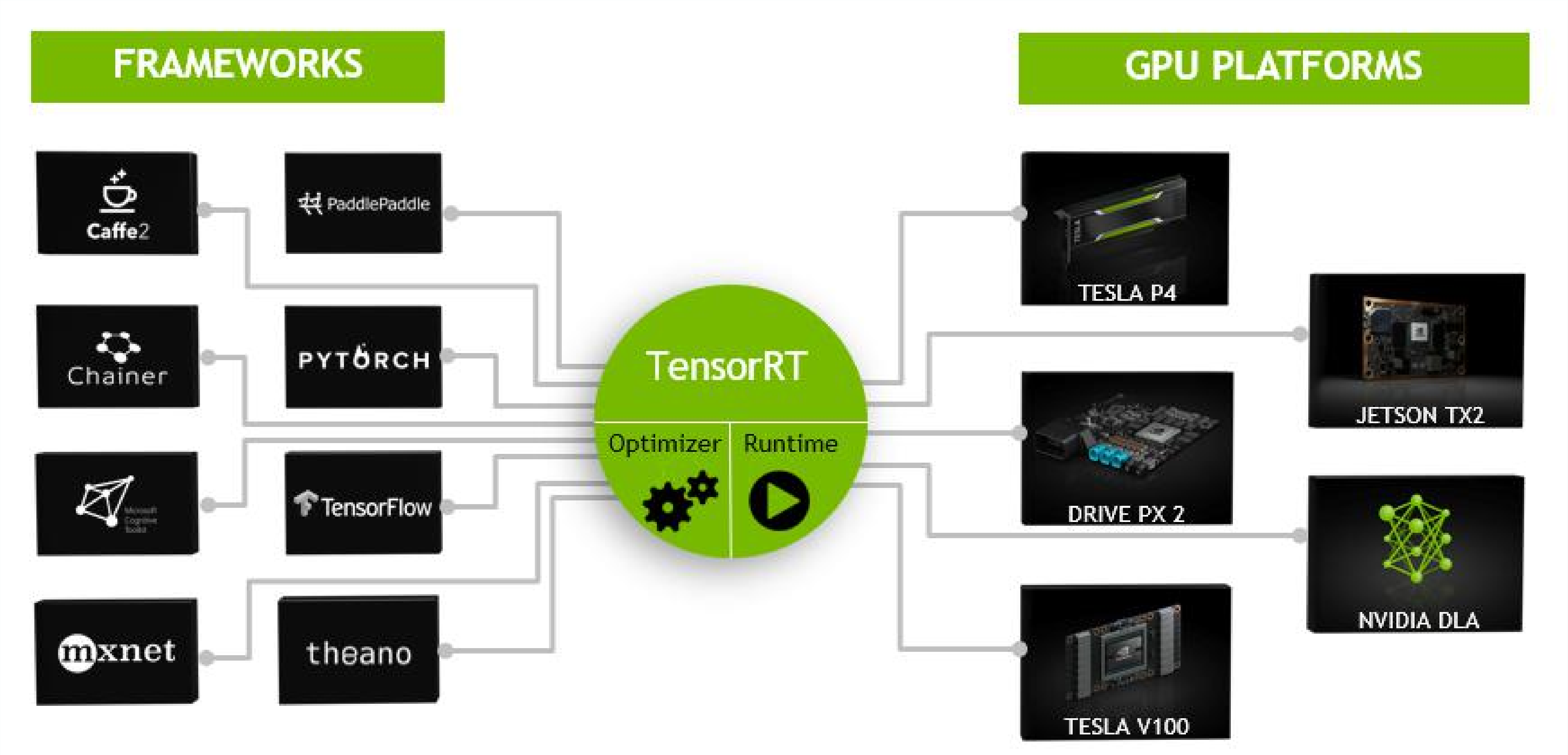
- Accuracy: ROC, Precision-Recall, confusion matrix, …
- Latency: milliseconds / inference
- Throughput: inferences/second
- Memory usage: memory that need to be reserved to do inference
- Efficiency: throughput/watt

What is TensorRT

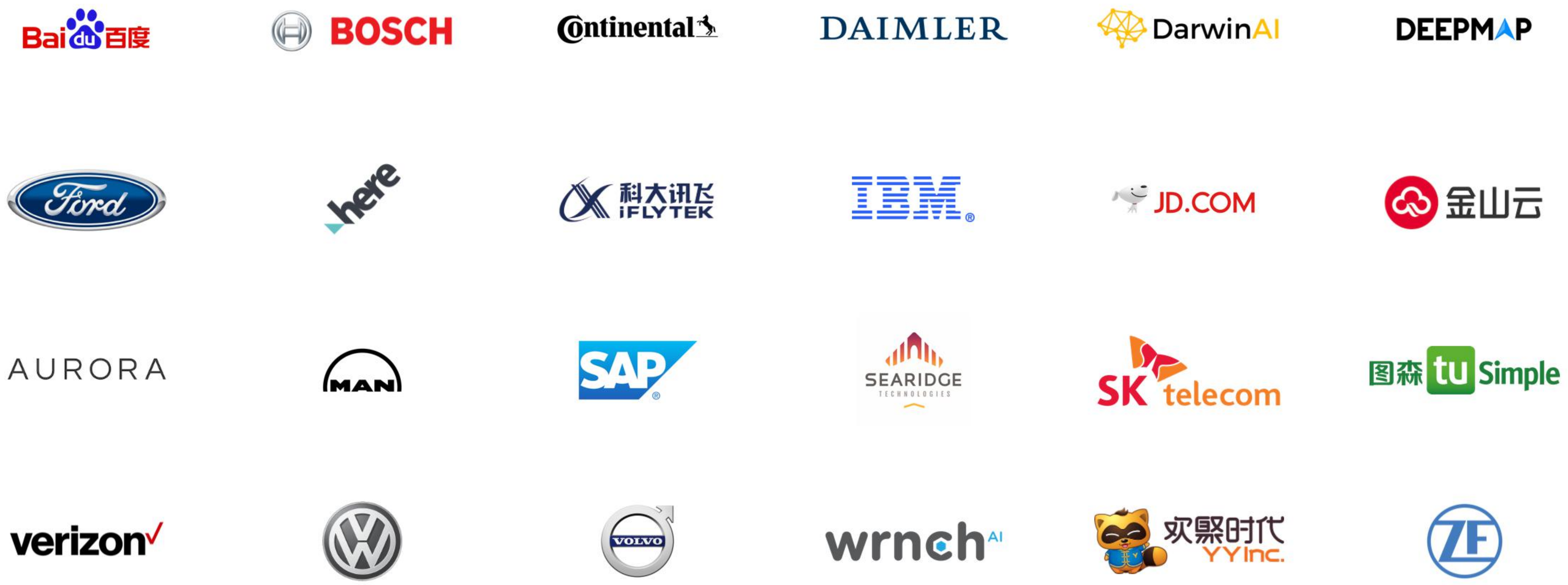


a C++ library
an optimiser + a runtime
provides C++ and Python API
integrated with frameworks

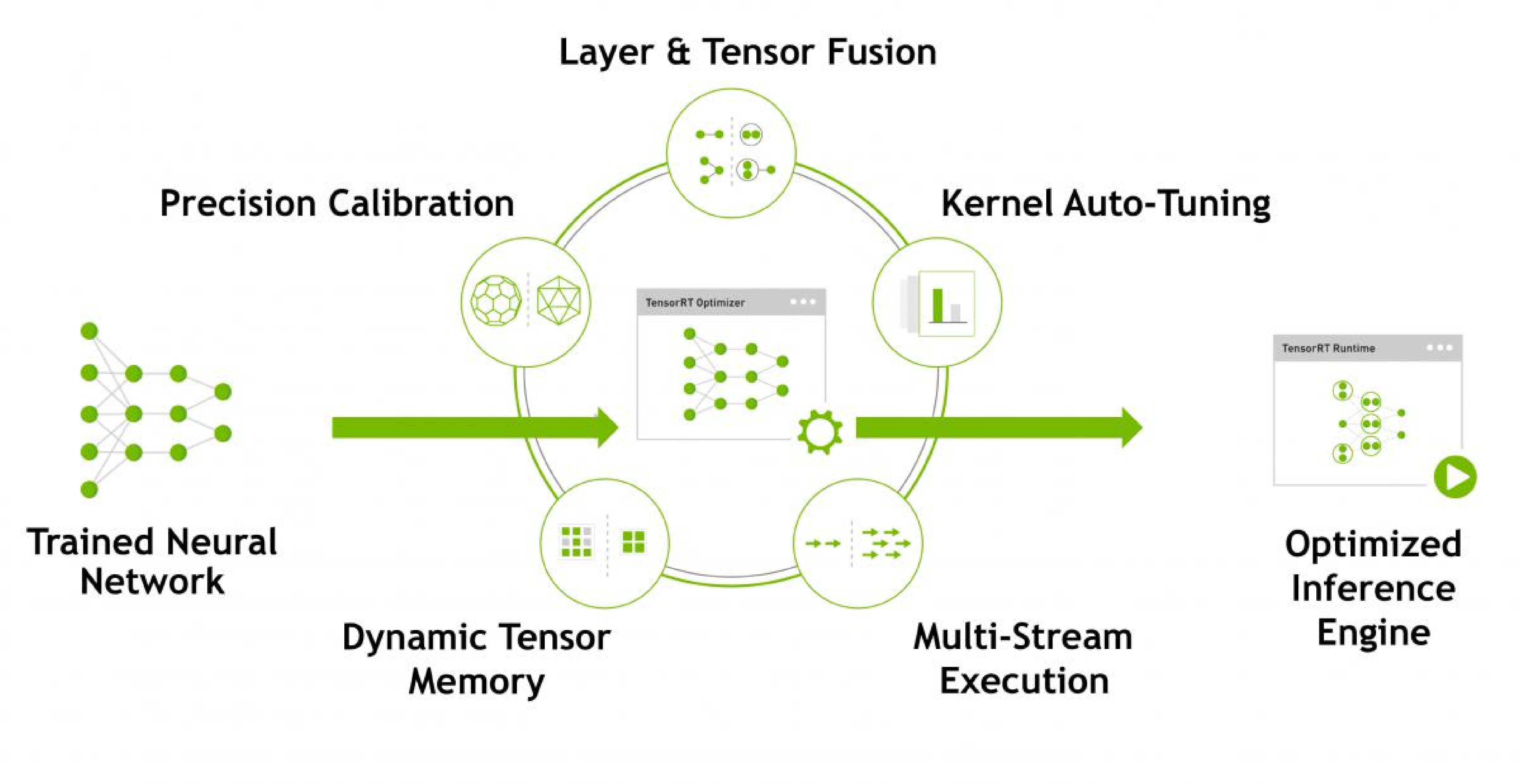
deep learning inference platform
support INT8, FP16, FP32 precisions.
GPU-specific


FRAMEWORKS
Caffe2
PaddlePaddle
Chainer
PYTORCH
Microsoft Cognitive Toolkit
TensorFlow
mxnet
theano
TensorRT
Optimizer
Runtime
GPU PLATFORMS
TESLA P4
JETSON TX2
DRIVE PX 2
NVIDIA DLA
TESLA V100

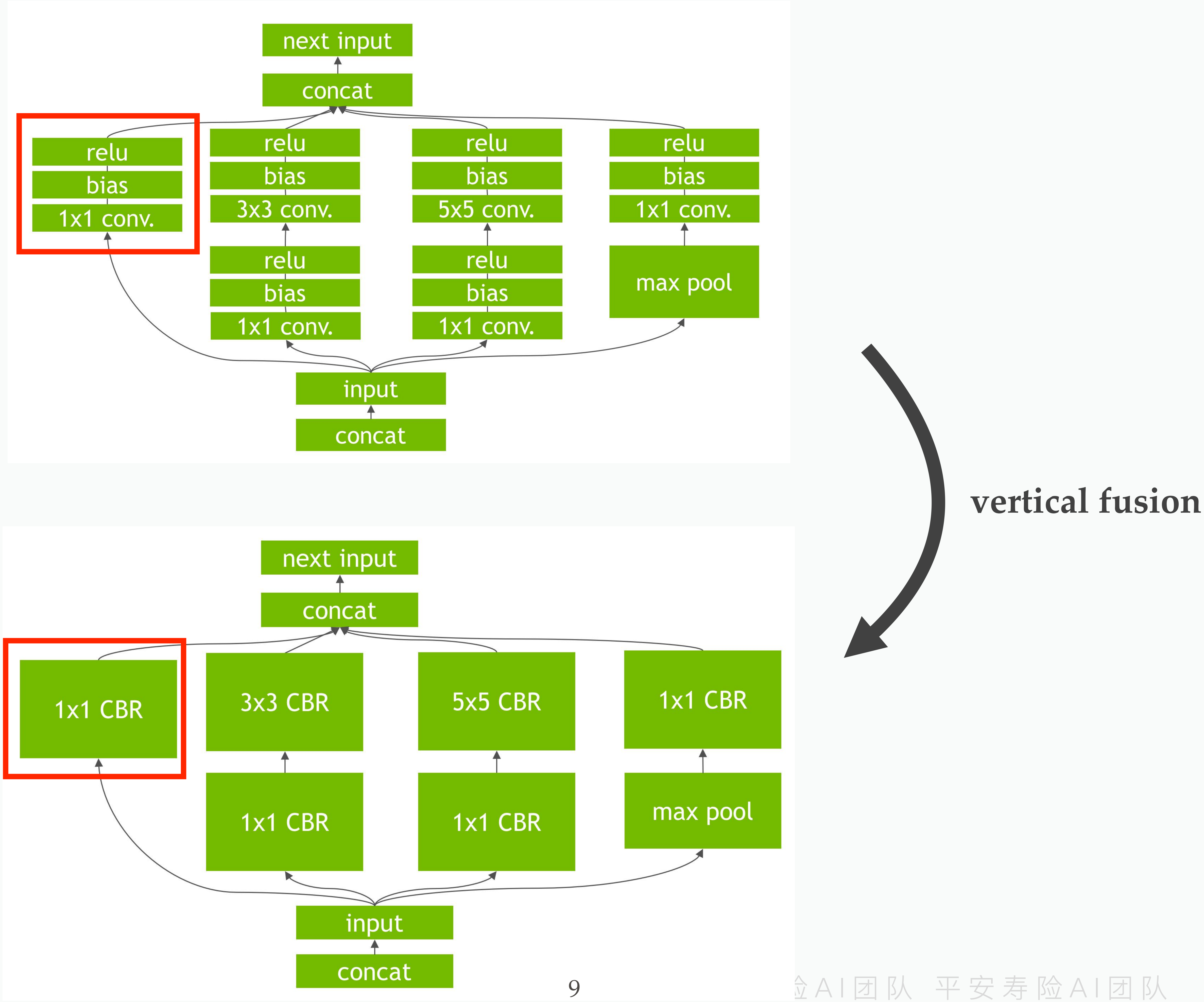
Widely Adopted

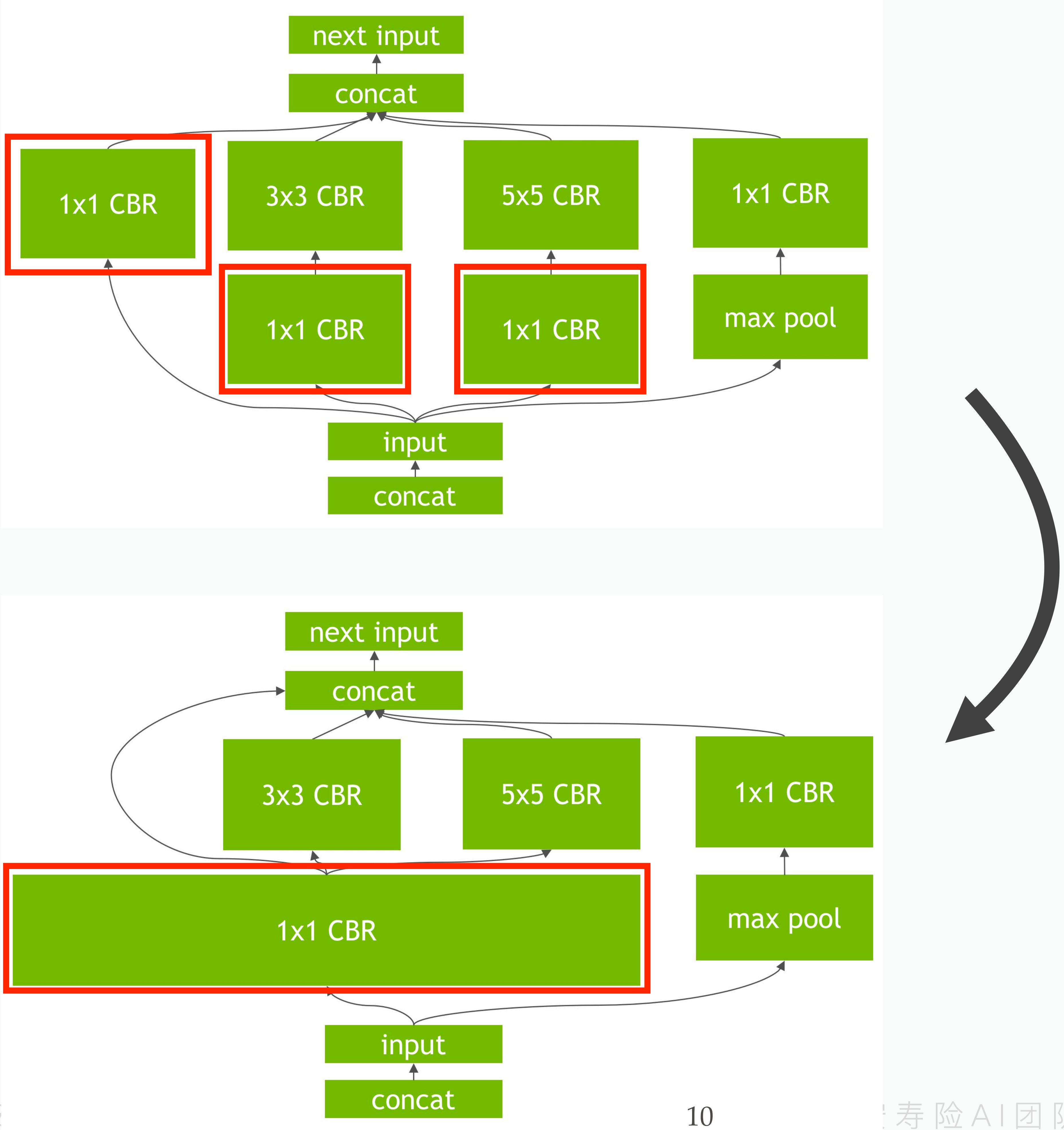


How it works


Layer & Tensor Fusion
Precision Calibration
Kernel Auto-Tuning
TensorRT Optimizer
TensorRT Runtime
Trained Neural Network
Optimized Inference Engine
Dynamic Tensor Memory
Multi-Stream Execution

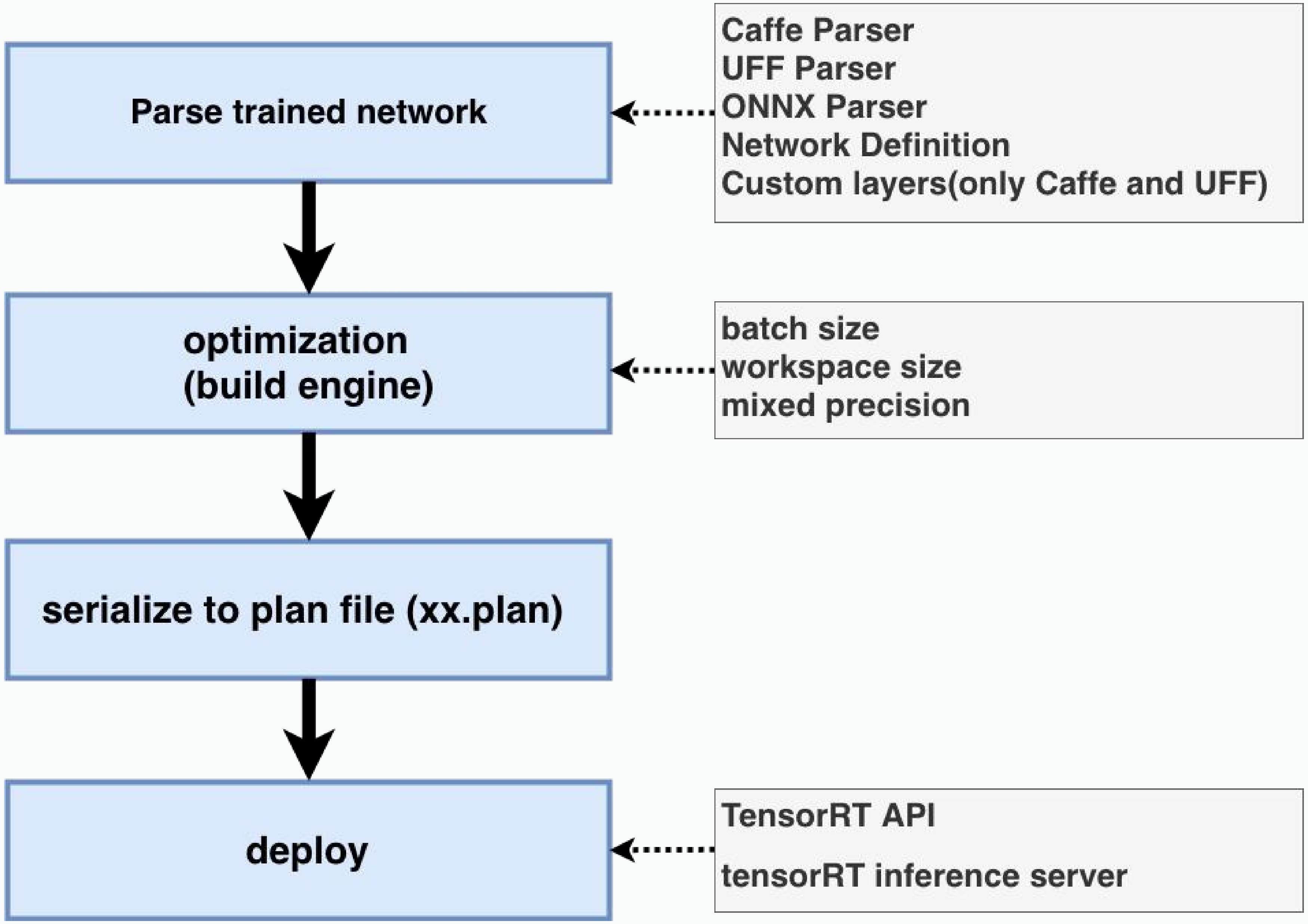
layer fusion example


next input
concat
relu
bias
1x1 conv.
relu
bias
3x3 conv.
relu
bias
1x1 conv.
relu
bias
5x5 conv.
relu
bias
1x1 conv.
relu
bias
1x1 conv.
max pool
input
concat
vertical fusion
next input
concat
1x1 CBR
3x3 CBR
1x1 CBR
5x5 CBR
1x1 CBR
1x1 CBR
max pool
input
concat



horizontal fusion

How to use it



Parse trained network

optimization
(build engine)

serialize to plan file (xx.plan)

deploy

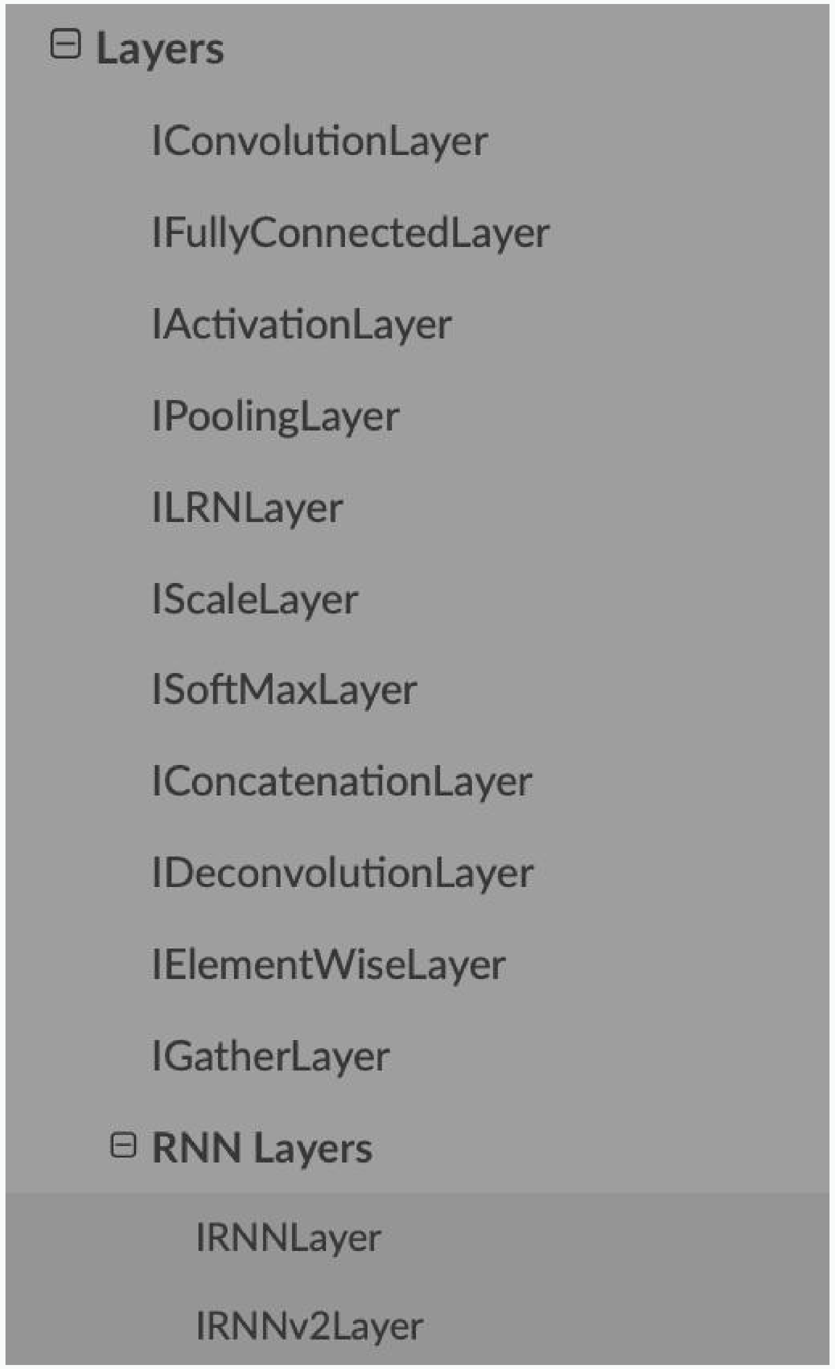
Caffe Parser
UFF Parser
ONNX Parser
Network Definition
Custom layers(only Caffe and UFF)

batch size
workspace size
mixed precision

TensorRT API

tensorRT inference server

TensorRT Supported Layers

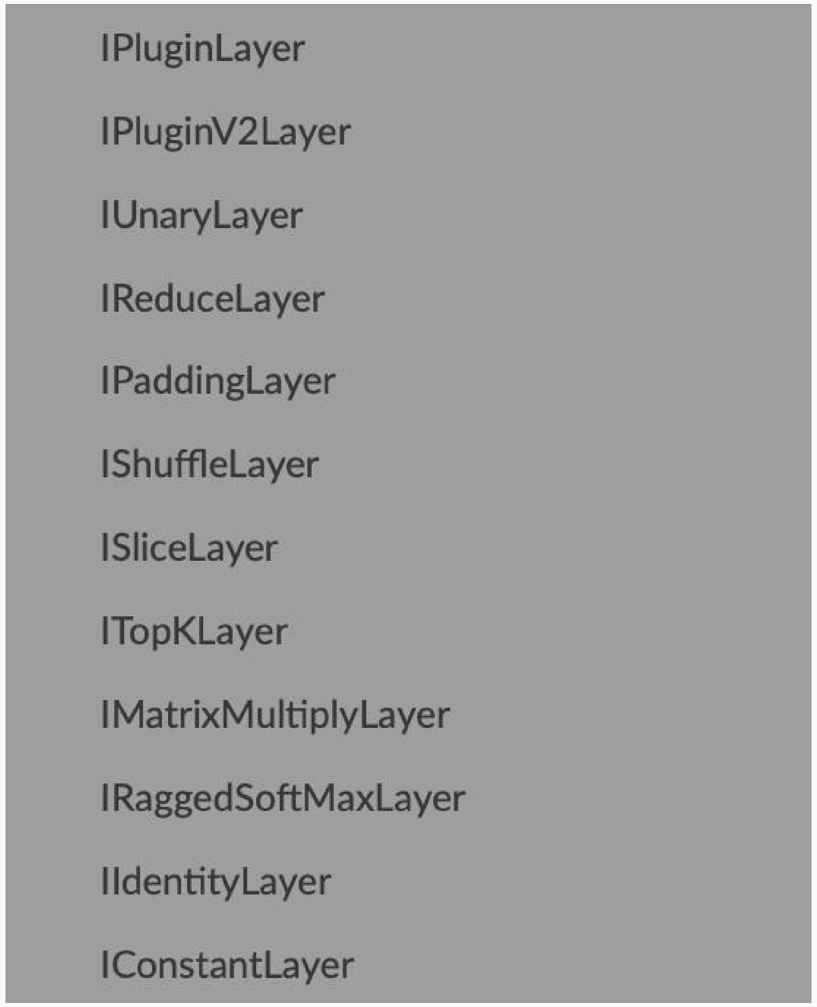


⊟ Layers

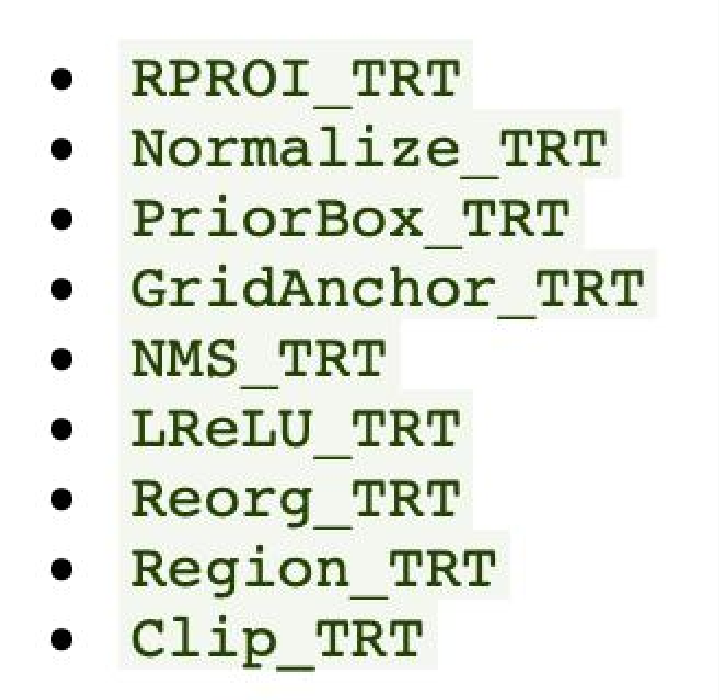
- IConvolutionLayer
- IFullyConnectedLayer
- IActivationLayer
- IPoolingLayer
- ILRNLayer
- IScaleLayer
- ISoftMaxLayer
- IConcatenationLayer
- IDeconvolutionLayer
- IElementWiseLayer
- IGatherLayer

⊟ RNN Layers

- IRNNLayer
- IRNNv2Layer

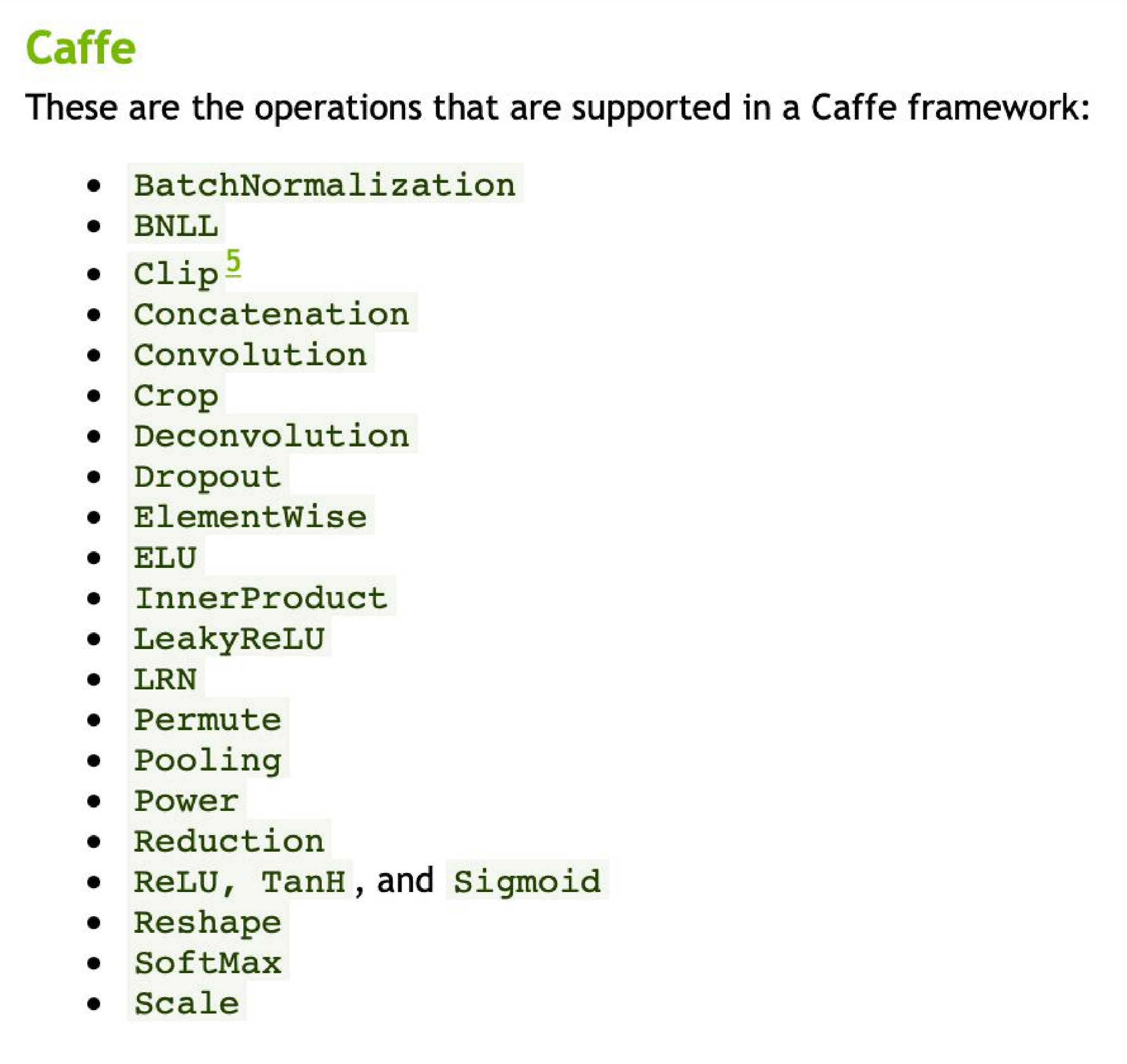


- IPluginLayer
- IPluginV2Layer
- IUnaryLayer
- IReduceLayer
- IPaddingLayer
- IShuffleLayer
- ISliceLayer
- ITopKLayer
- IMatrixMultiplyLayer
- IRaggedSoftMaxLayer
- IIdentityLayer
- IConstantLayer



- `RPROI_TRT`
- `Normalize_TRT`
- `PriorBox_TRT`
- `GridAnchor_TRT`
- `NMS_TRT`
- `LReLU_TRT`
- `Reorg_TRT`
- `Region_TRT`
- `Clip_TRT`

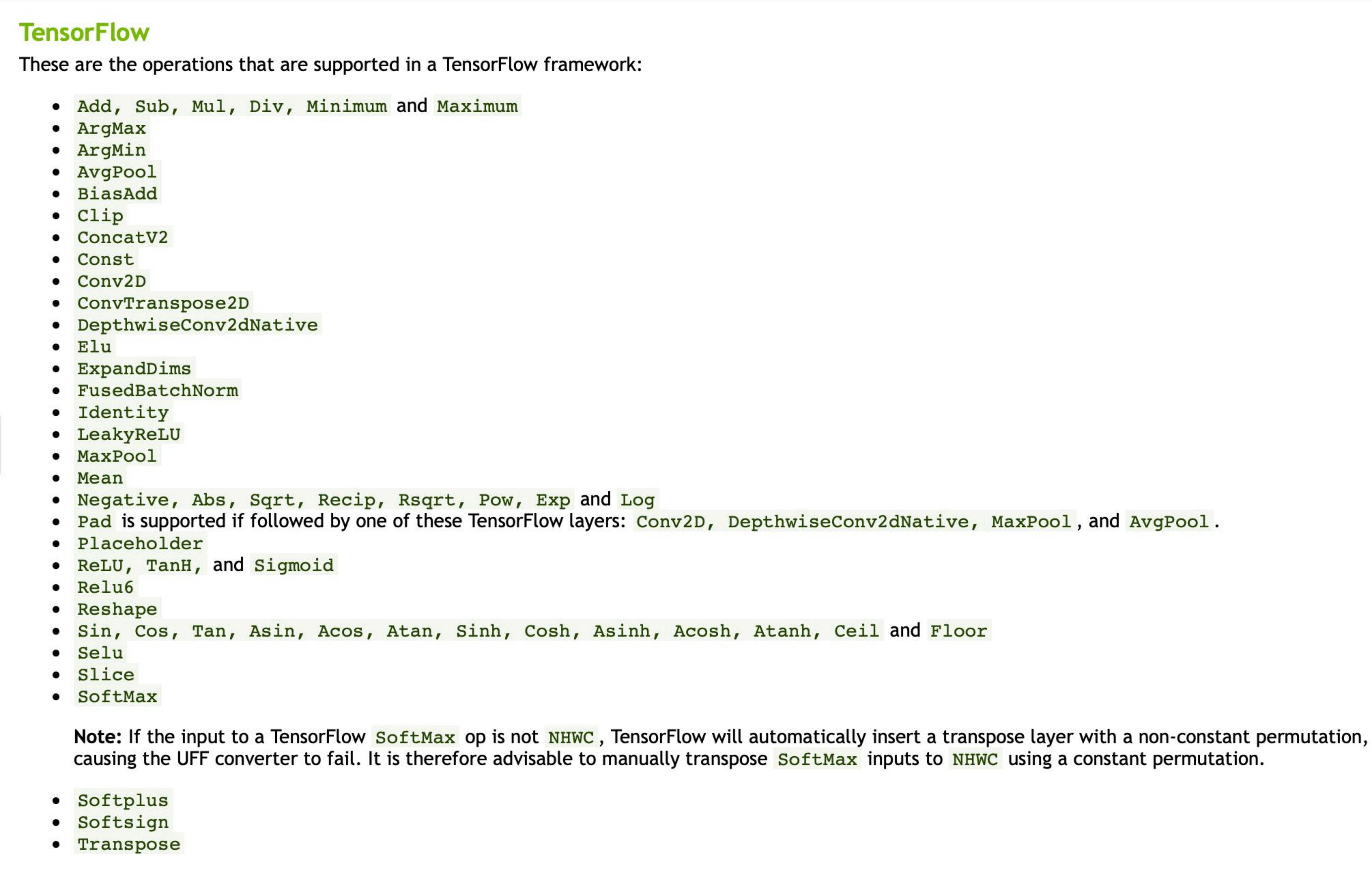
Supported Layers in Frameworks



Caffe

These are the operations that are supported in a Caffe framework:

- `BatchNormalization`
- `BNLL`
- `Clip`[5]
- `Concatenation`
- `Convolution`
- `Crop`
- `Deconvolution`
- `Dropout`
- `ElementWise`
- `ELU`
- `InnerProduct`
- `LeakyReLU`
- `LRN`
- `Permute`
- `Pooling`
- `Power`
- `Reduction`
- `ReLU, TanH`, and `Sigmoid`
- `Reshape`
- `SoftMax`
- `Scale`



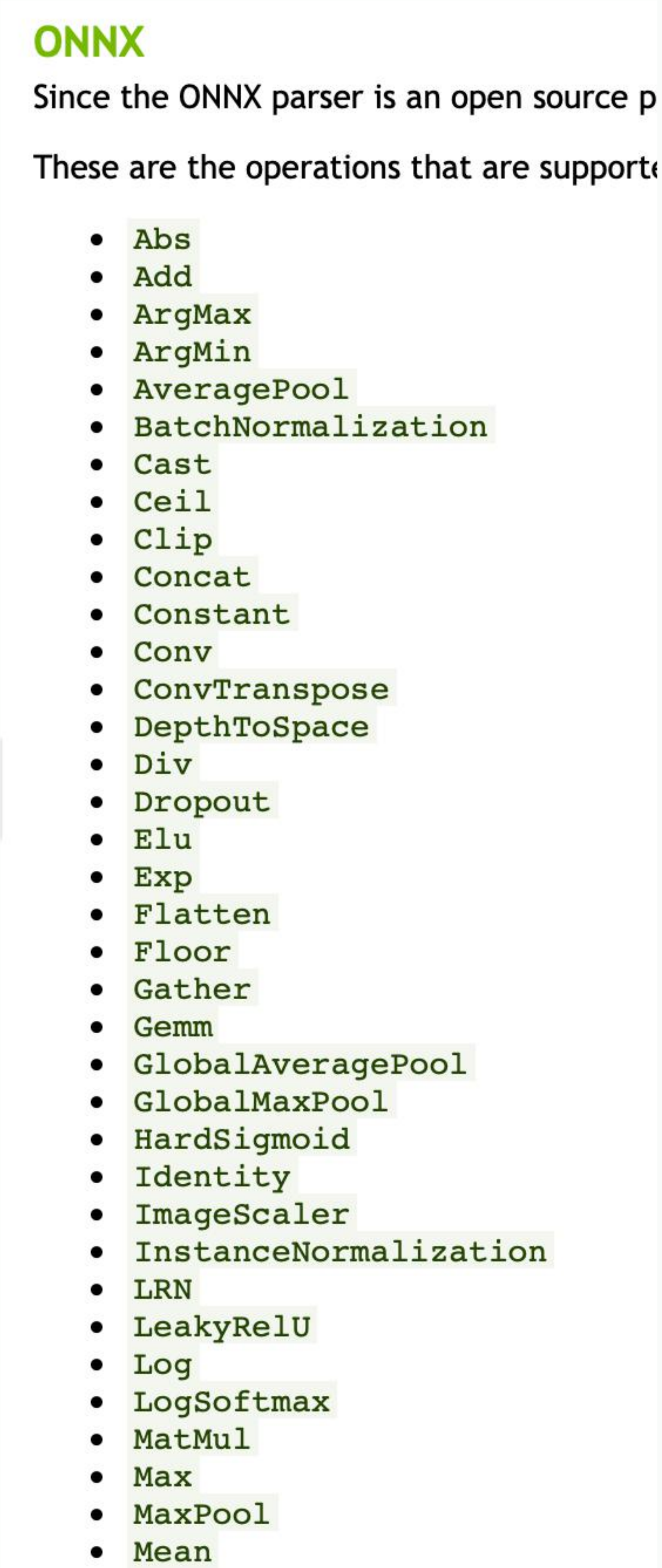
TensorFlow

These are the operations that are supported in a TensorFlow framework:

- `Add, Sub, Mul, Div, Minimum` and `Maximum`
- `ArgMax`
- `ArgMin`
- `AvgPool`
- `BiasAdd`
- `Clip`
- `ConcatV2`
- `Const`
- `Conv2D`
- `ConvTranspose2D`
- `DepthwiseConv2dNative`
- `Elu`
- `ExpandDims`
- `FusedBatchNorm`
- `Identity`
- `LeakyReLU`
- `MaxPool`
- `Mean`
- `Negative, Abs, Sqrt, Recip, Rsqrt, Pow, Exp` and `Log`
- `Pad` is supported if followed by one of these TensorFlow layers: `Conv2D, DepthwiseConv2dNative, MaxPool`, and `AvgPool`.
- `Placeholder`
- `ReLU, TanH,` and `Sigmoid`
- `Relu6`
- `Reshape`
- `Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Asinh, Acosh, Atanh, Ceil` and `Floor`
- `Selu`
- `Slice`
- `SoftMax`

  Note: If the input to a TensorFlow `SoftMax` op is not `NHWC`, TensorFlow will automatically insert a transpose layer with a non-constant permutation, causing the UFF converter to fail. It is therefore advisable to manually transpose `SoftMax` inputs to `NHWC` using a constant permutation.

- `Softplus`
- `Softsign`
- `Transpose`

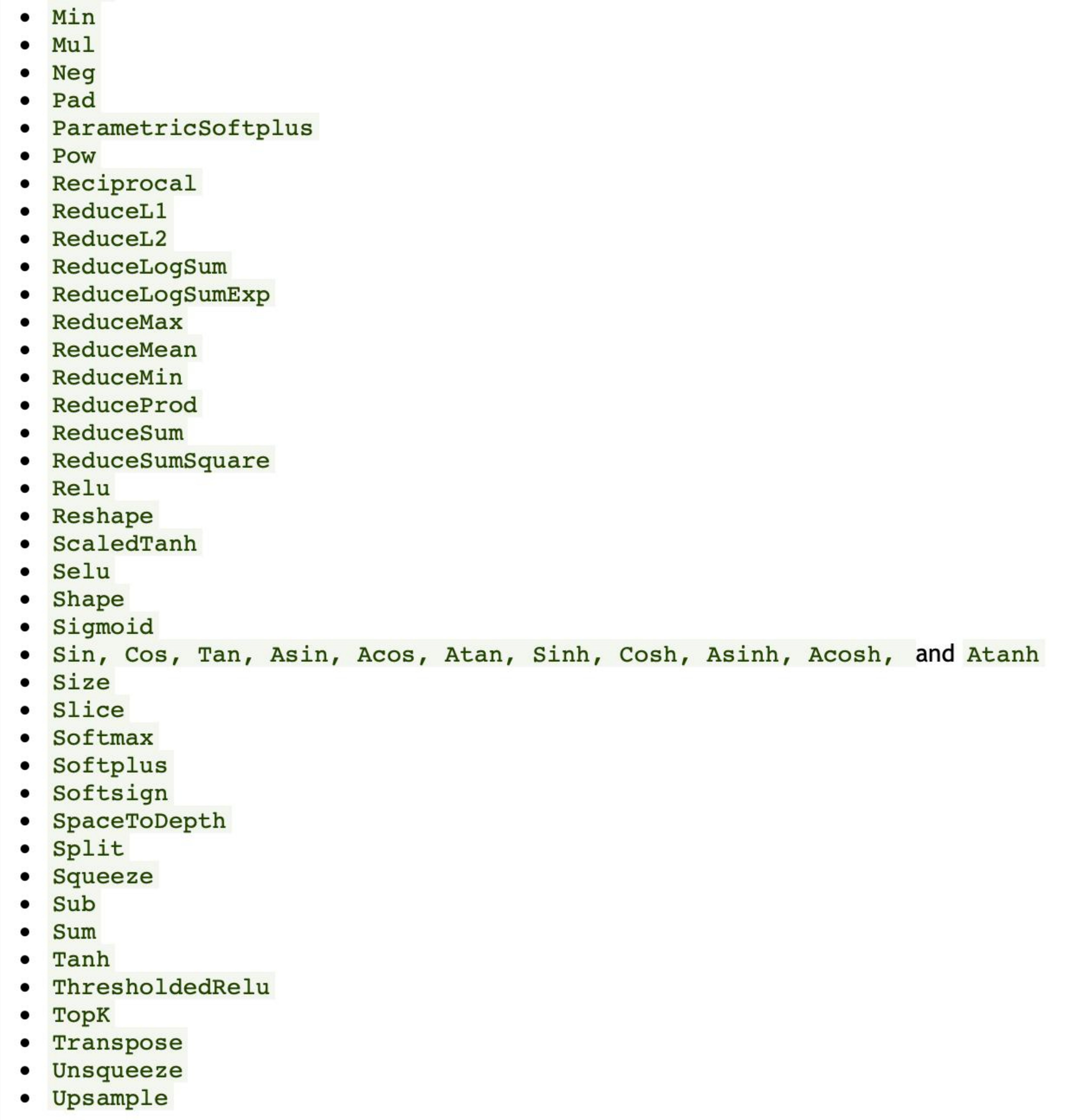


ONNX

Since the ONNX parser is an open source p

These are the operations that are support

- `Abs`
- `Add`
- `ArgMax`
- `ArgMin`
- `AveragePool`
- `BatchNormalization`
- `Cast`
- `Ceil`
- `Clip`
- `Concat`
- `Constant`
- `Conv`
- `ConvTranspose`
- `DepthToSpace`
- `Div`
- `Dropout`
- `Elu`
- `Exp`
- `Flatten`
- `Floor`
- `Gather`
- `Gemm`
- `GlobalAveragePool`
- `GlobalMaxPool`
- `HardSigmoid`
- `Identity`
- `ImageScaler`
- `InstanceNormalization`
- `LRN`
- `LeakyRelU`
- `Log`
- `LogSoftmax`
- `MatMul`
- `Max`
- `MaxPool`
- `Mean`



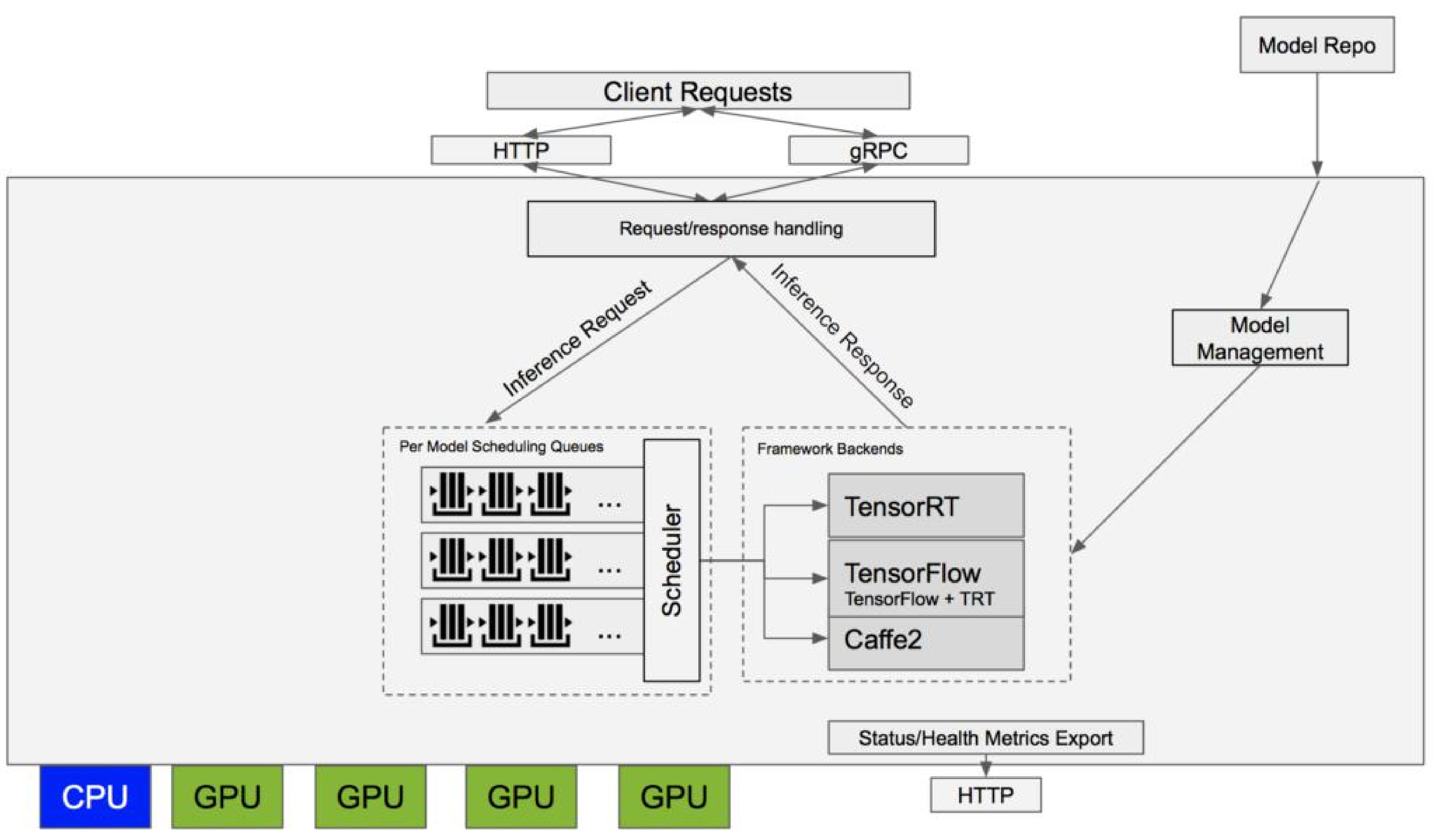
- `Min`
- `Mul`
- `Neg`
- `Pad`
- `ParametricSoftplus`
- `Pow`
- `Reciprocal`
- `ReduceL1`
- `ReduceL2`
- `ReduceLogSum`
- `ReduceLogSumExp`
- `ReduceMax`
- `ReduceMean`
- `ReduceMin`
- `ReduceProd`
- `ReduceSum`
- `ReduceSumSquare`
- `Relu`
- `Reshape`
- `ScaledTanh`
- `Selu`
- `Shape`
- `Sigmoid`
- `Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Asinh, Acosh,` and `Atanh`
- `Size`
- `Slice`
- `Softmax`
- `Softplus`
- `Softsign`
- `SpaceToDepth`
- `Split`
- `Squeeze`
- `Sub`
- `Sum`
- `Tanh`
- `ThresholdedRelu`
- `TopK`
- `Transpose`
- `Unsqueeze`
- `Upsample`

Design your network with tensorRT-supported layers !

Demo

(ResNet50)

TensorRT inference server



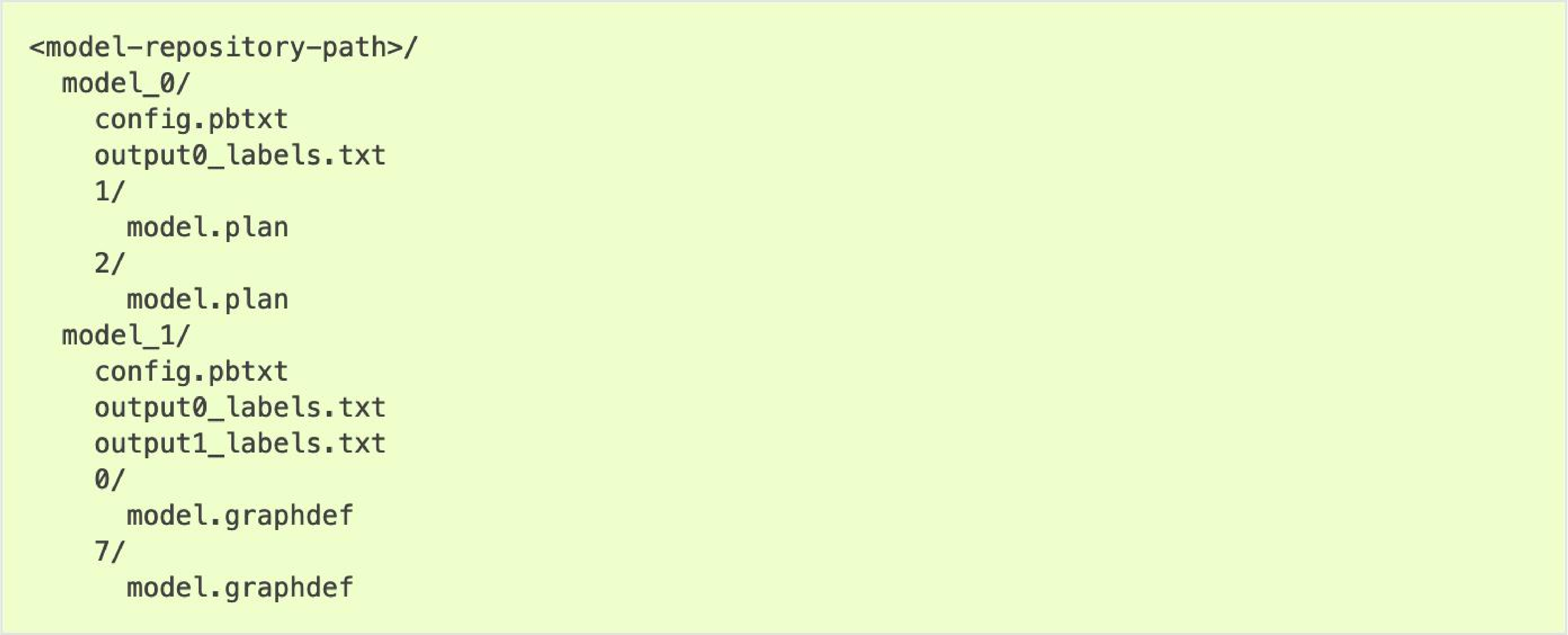
multiple models **multi-GPUs** **multi-framework** **model repository**

TensorRT inference server

Usage:

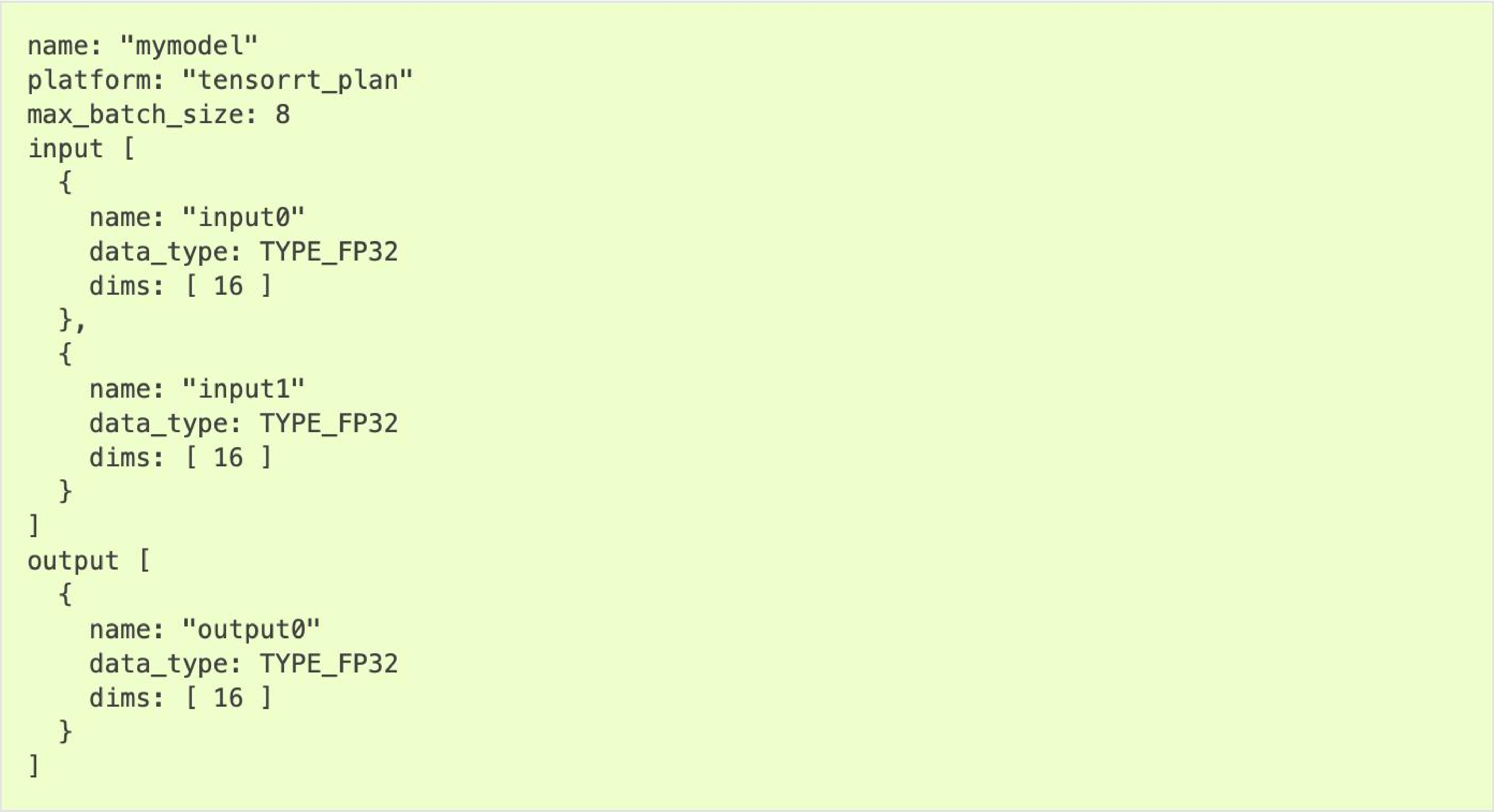
```
nvidia-docker run   --rm --shm-size=1g --ulimit memlock=-1  \
                    --ulimit stack=67108864  \
                    -p 8000:8000   -p 8001:8001   -p 8002:8002  \
                    -v /path/to/model/repository:/models  \
                    <tensorrtserver image name>  trtserver  \
                    --model-store=/models
```

Model repository



```
<model-repository-path>/
  model_0/
    config.pbtxt
    output0_labels.txt
    1/
      model.plan
    2/
      model.plan
  model_1/
    config.pbtxt
    output0_labels.txt
    output1_labels.txt
    0/
      model.graphdef
    7/
      model.graphdef
```

Model configuration file



```
name: "mymodel"
platform: "tensorrt_plan"
max_batch_size: 8
input [
  {
    name: "input0"
    data_type: TYPE_FP32
    dims: [ 16 ]
  },
  {
    name: "input1"
    data_type: TYPE_FP32
    dims: [ 16 ]
  }
]
output [
  {
    name: "output0"
    data_type: TYPE_FP32
    dims: [ 16 ]
  }
]
```

Demo

(ResNet50)

References

[1] tensorRT developer guide
https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html#dla_topic

[2] tensor best practices
https://docs.nvidia.com/deeplearning/sdk/tensorrt-best-practices/index.html#fusion-types

[3] tensorRT inference server guide
https://docs.nvidia.com/deeplearning/sdk/tensorrt-inference-server-guide/docs/index.html

[4] tensorRT inference server GitHub sources
https://github.com/NVIDIA/tensorrt-inference-server

[5] supported layers
https://docs.nvidia.com/deeplearning/sdk/tensorrt-archived/tensorrt-512rc/tensorrt-support-matrix/index.html

[6] layer fusion
https://devblogs.nvidia.com/deploying-deep-learning-nvidia-tensorrt/

Question ?