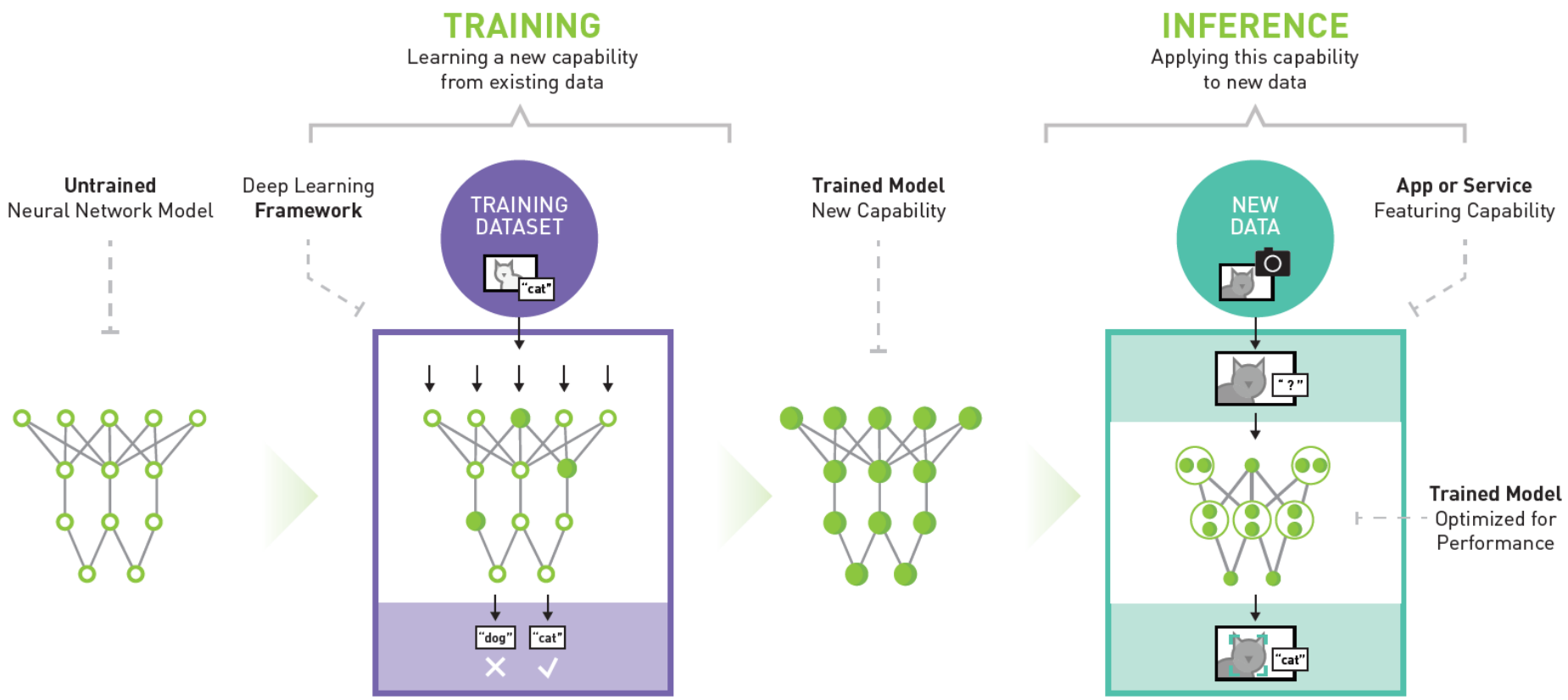
AI CONFERENCE

Inference Optimization Using TensorRT with Use Cases

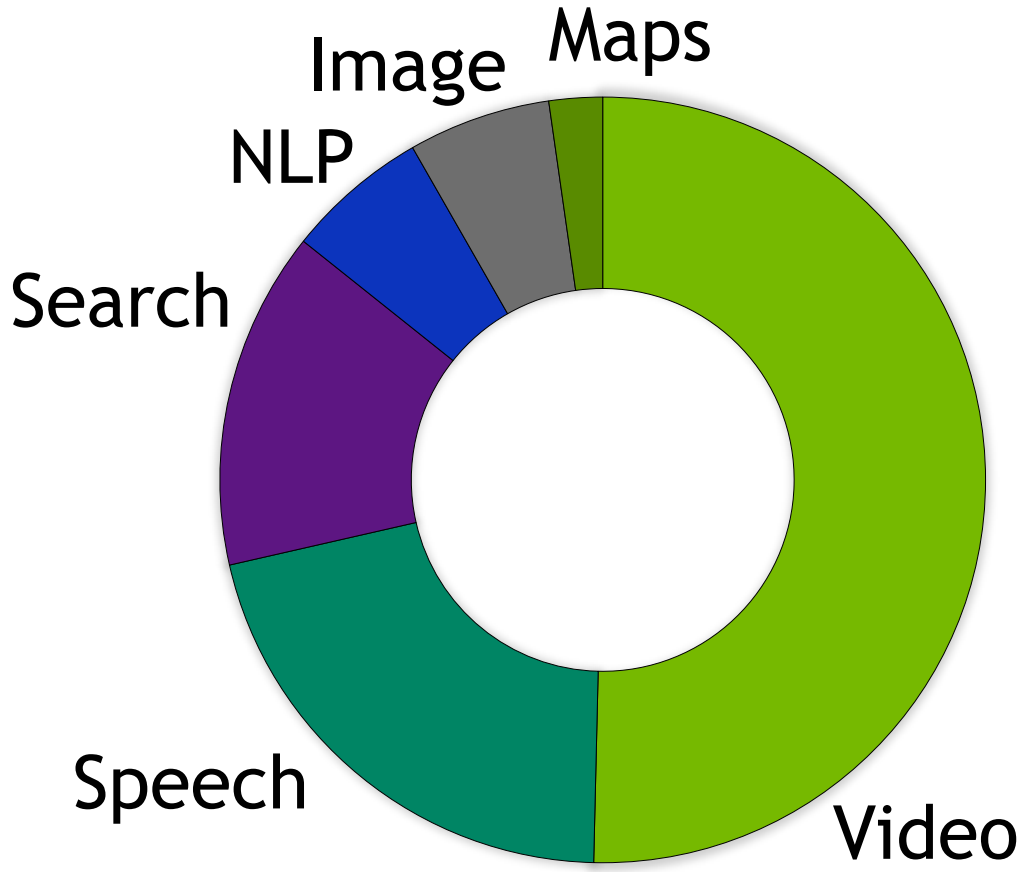
Jack Han / 한재근 | Solutions Architect | NVIDIA

NVIDIA AI CONFERENCE



TensorRT 4 Adoption

Use Cases



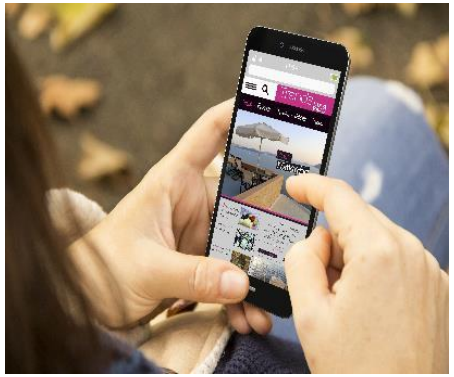
AI Inference is exploding

1 Billion
Videos Watched Per Day
Facebook

LIVE VIDEO

1 Billion
Voice Searches Per Day
Google, Bing, etc.

SPEECH



1 Trillion
Ads/Rankings Per Day
Impressions

RECOMMENDATIONS

Bigger and More Compute Intensive in quest for accuracy

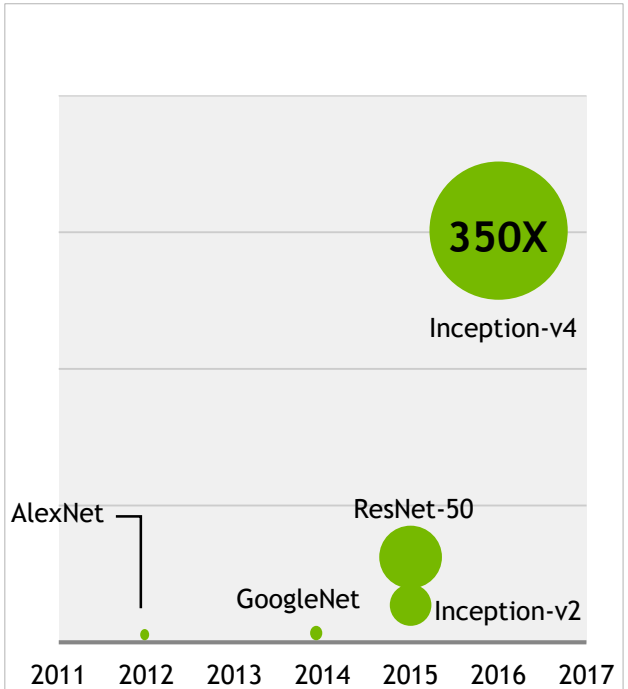
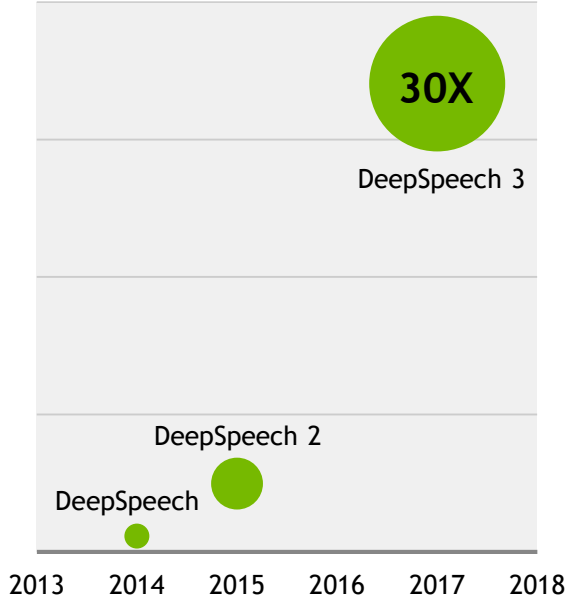
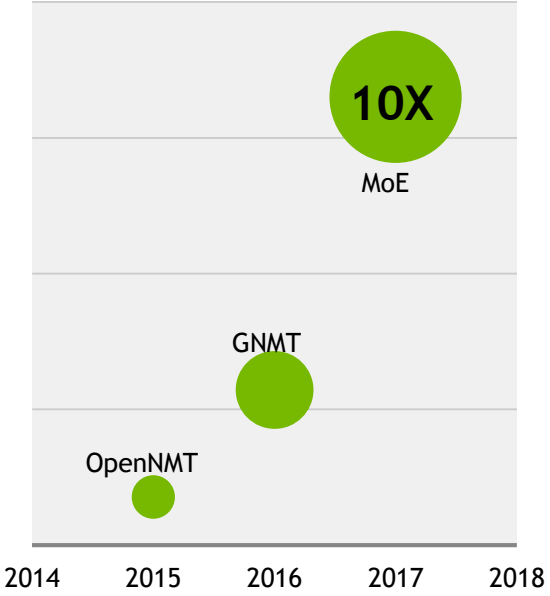


Image
(GOP * Bandwidth)



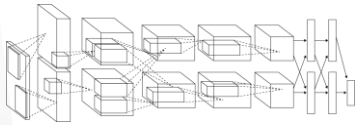
Speech
(GOP * Bandwidth)



Translation
(GOP * Bandwidth)

Cambrian Explosion

Convolutional Networks




Encoder/Decoder


ReLu

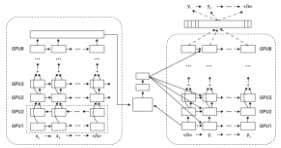

BatchNorm


Concat


Dropout


Pooling

Recurrent Networks




LSTM


GRU

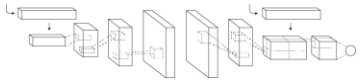

Beam Search

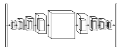

WaveNet


CTC


Attention

Generative Adversarial Networks




3D-GAN

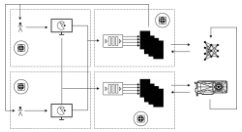

MedGAN


Conditional GAN

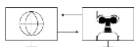

Coupled GAN


Speech Enhancement GAN

Reinforcement Learning

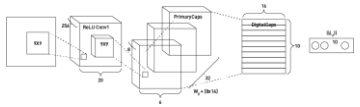


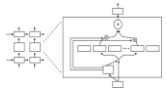

DQN

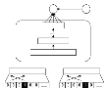

Simulation


DDPG

New Species



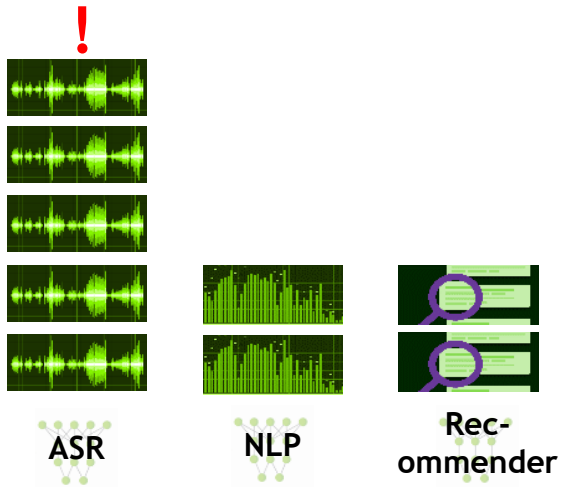

Mixture of Experts


Neural Collaborative Filtering


Block Sparse LSTM

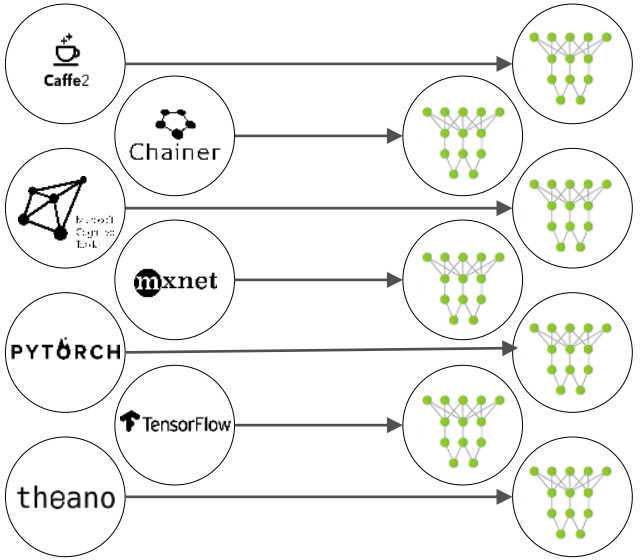
Inefficiency Limits Innovation

Single Model Only



Some systems are overused while others are underutilized

Single Framework Only

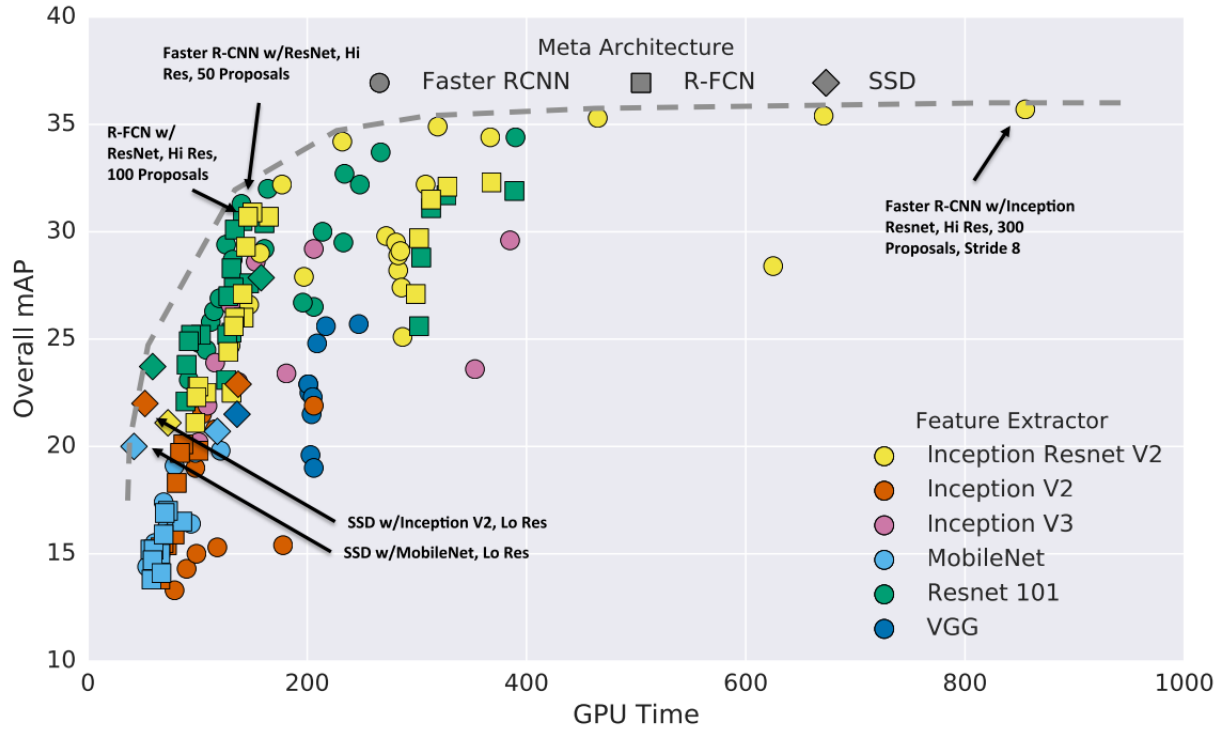


Solutions can only support models from one framework

Custom Development

Developers need to reinvent the plumbing for every application

More Accuracy = Time



Speed/accuracy trade-offs for modern convolutional object detectors, Google Research

Inference Performance

Latency

Throughput

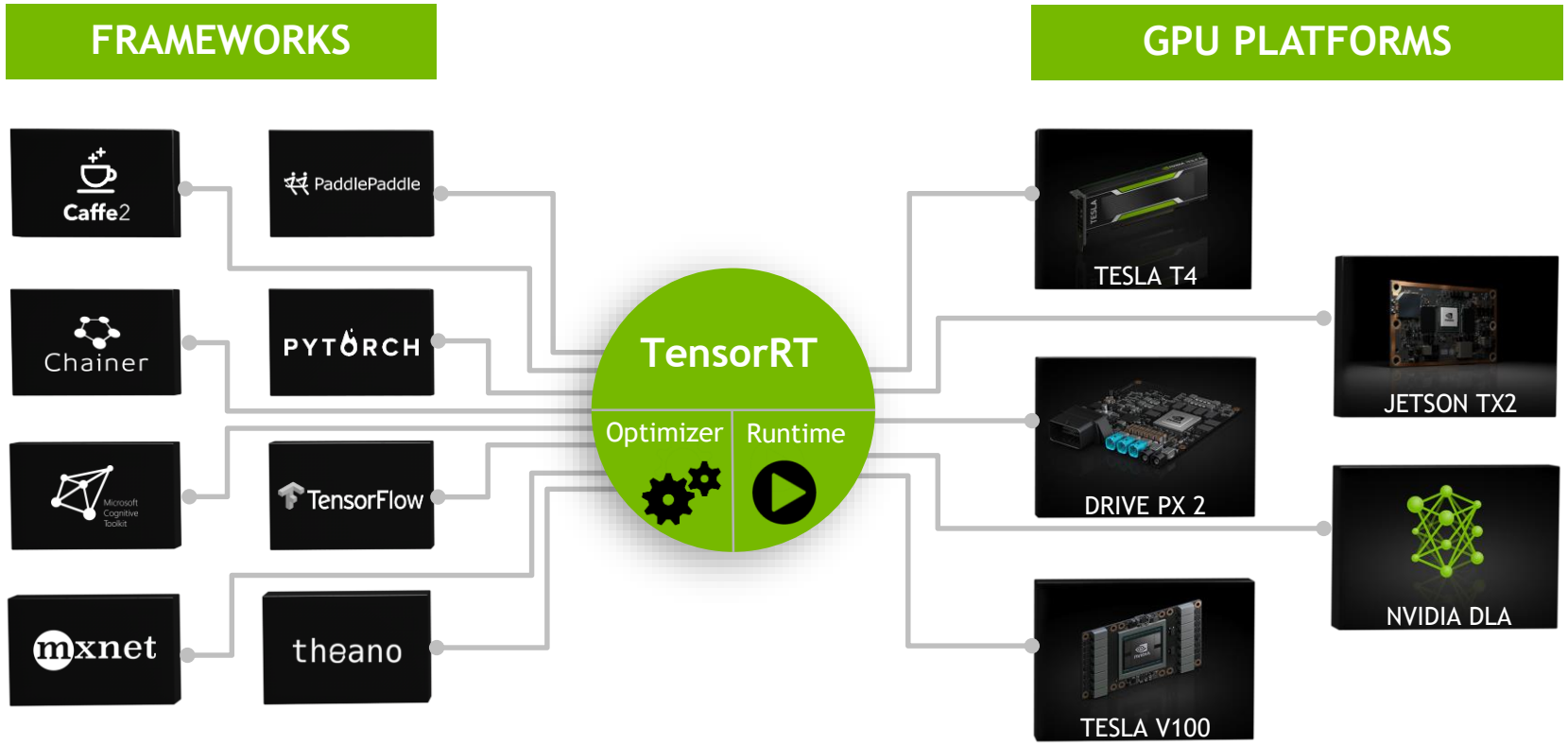
Efficiency

Easy Deploy

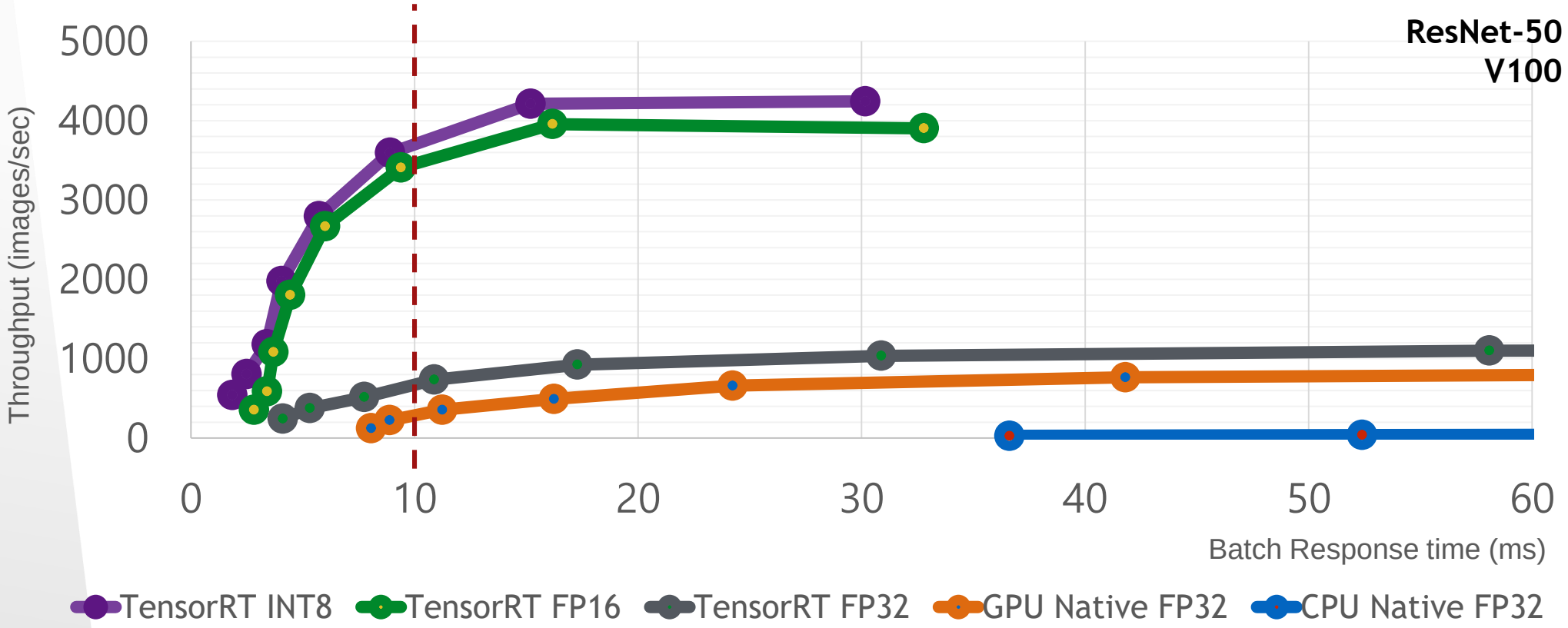
Inference Optimization

Lightweight Model	Deep Compression
Small Computation	No Network Modification
Suitable Lightweight Hardware	Continuous the Enovation
E.g;	E.g;
MobileNet	Quantization and Binarization
SqueezeNet	Network Pruning and Sharing
SEP-Nets	Distillation / Factorization

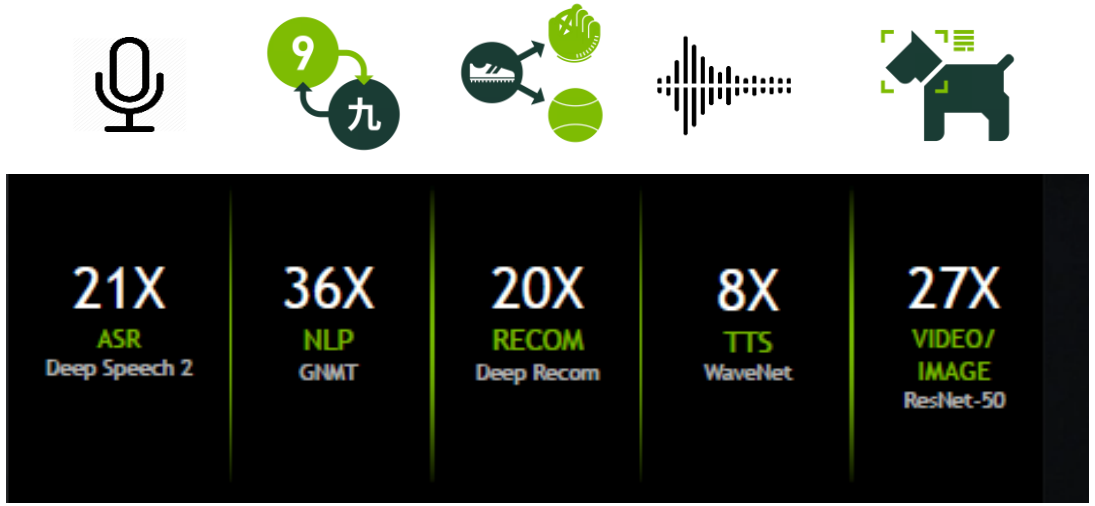
TensorRT
From framework to target

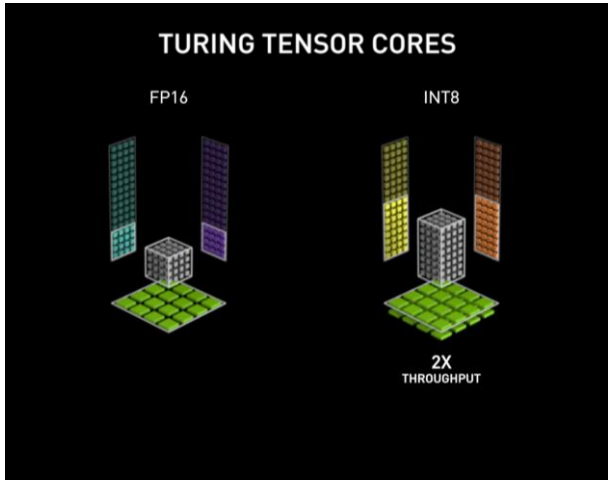


Speed-up


ResNet-50
V100
Throughput (images/sec)
5000
4000
3000
2000
1000
0
0
10
20
30
40
50
60
Batch Response time (ms)
TensorRT INT8
TensorRT FP16
TensorRT FP32
GPU Native FP32
CPU Native FP32

TensorRT 5





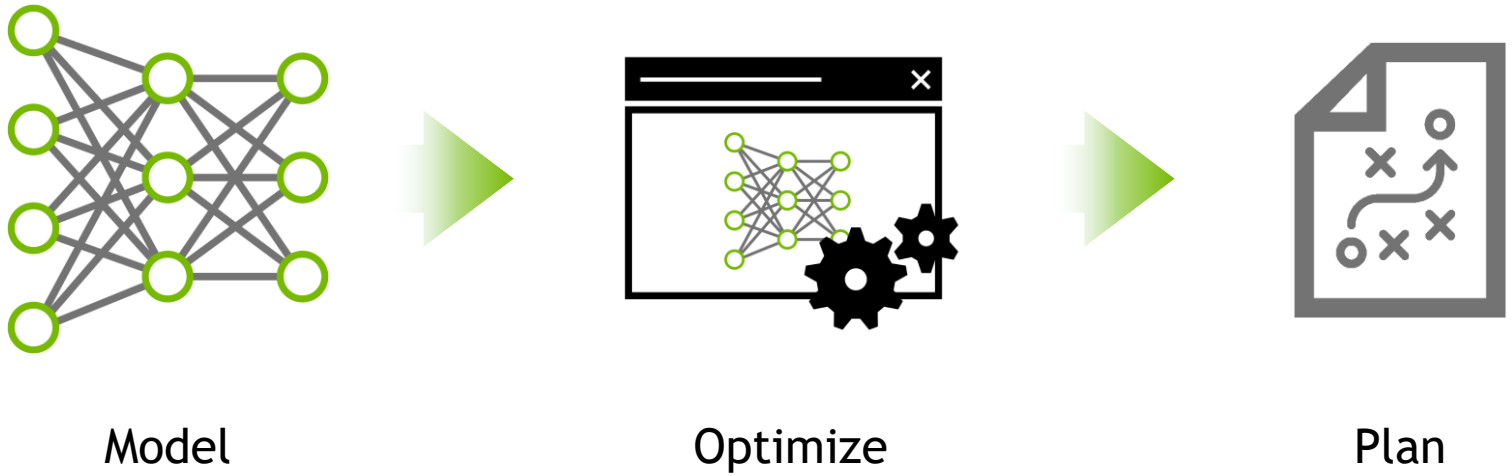
More Layers / Plugin / APIs

Inference Server Integration

ubuntu CentOS Windows

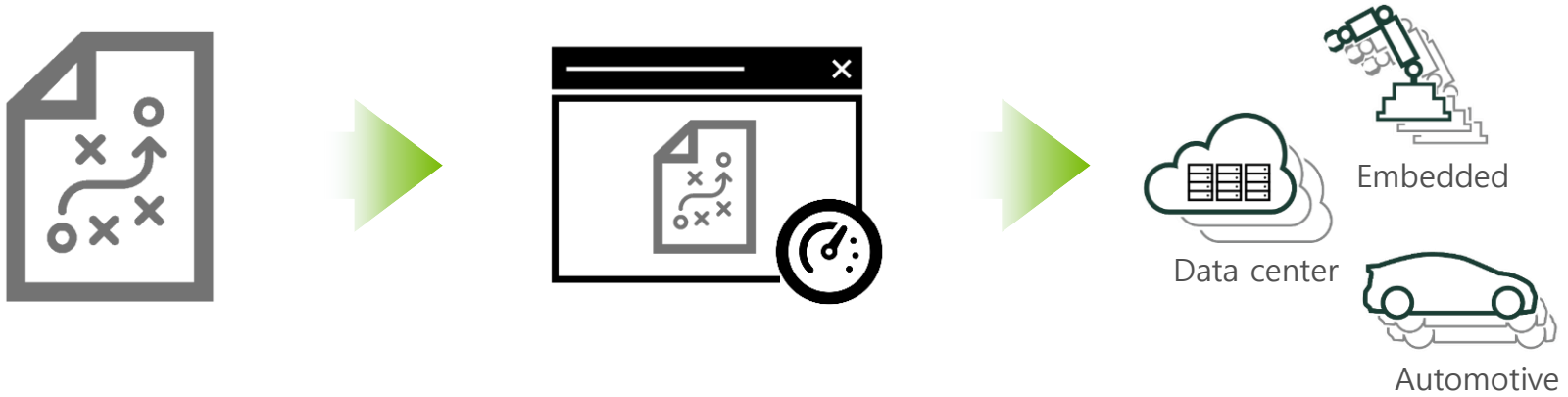
TensorRT Workflow

Step 1. Optimize trained model

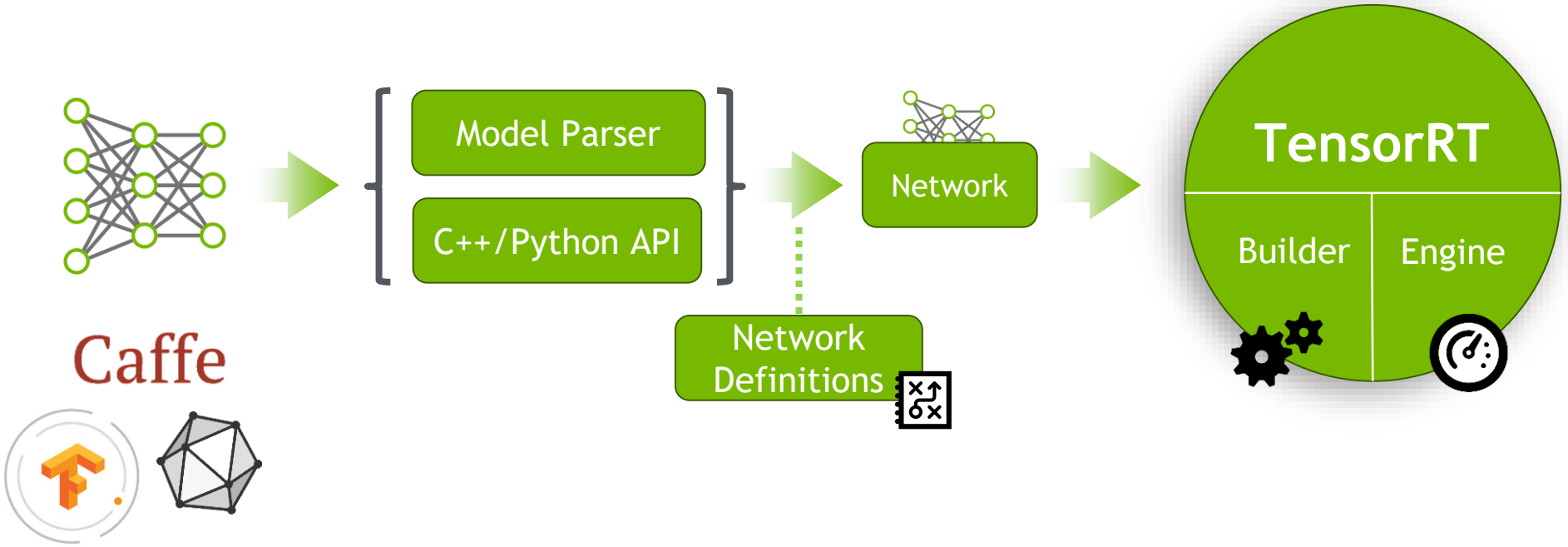


TensorRT Workflow

Step 2. Deploy optimized plan



Step-by-Step TensorRT Plan Build



https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html#support_op

7 Steps to Deployment with TensorRT

Step 1: Convert trained model into TensorRT format

Step 2: Create a model parser

Step 3: Register inputs and outputs

Step 4: Optimize model and create a runtime engine

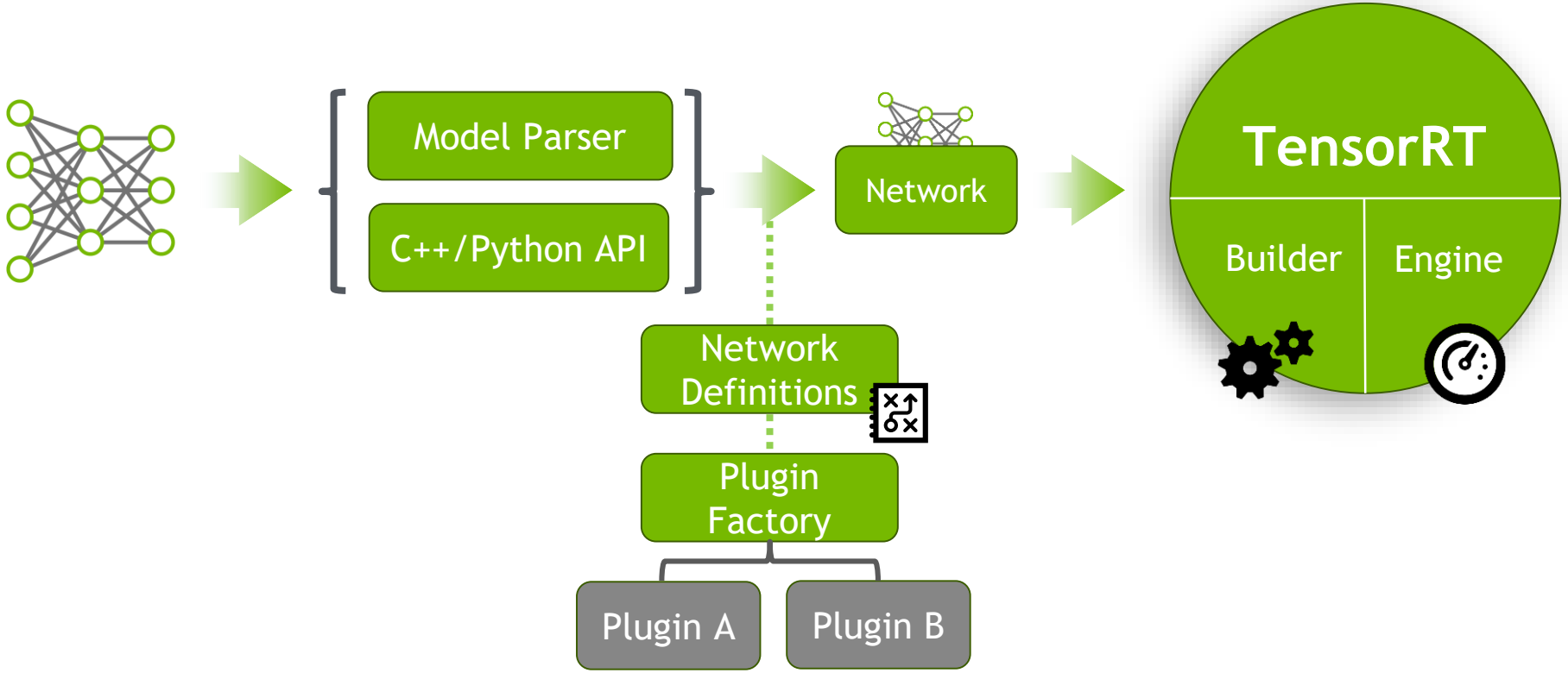
Step 5: Serialize optimized engine

Step 6: De-serialize engine

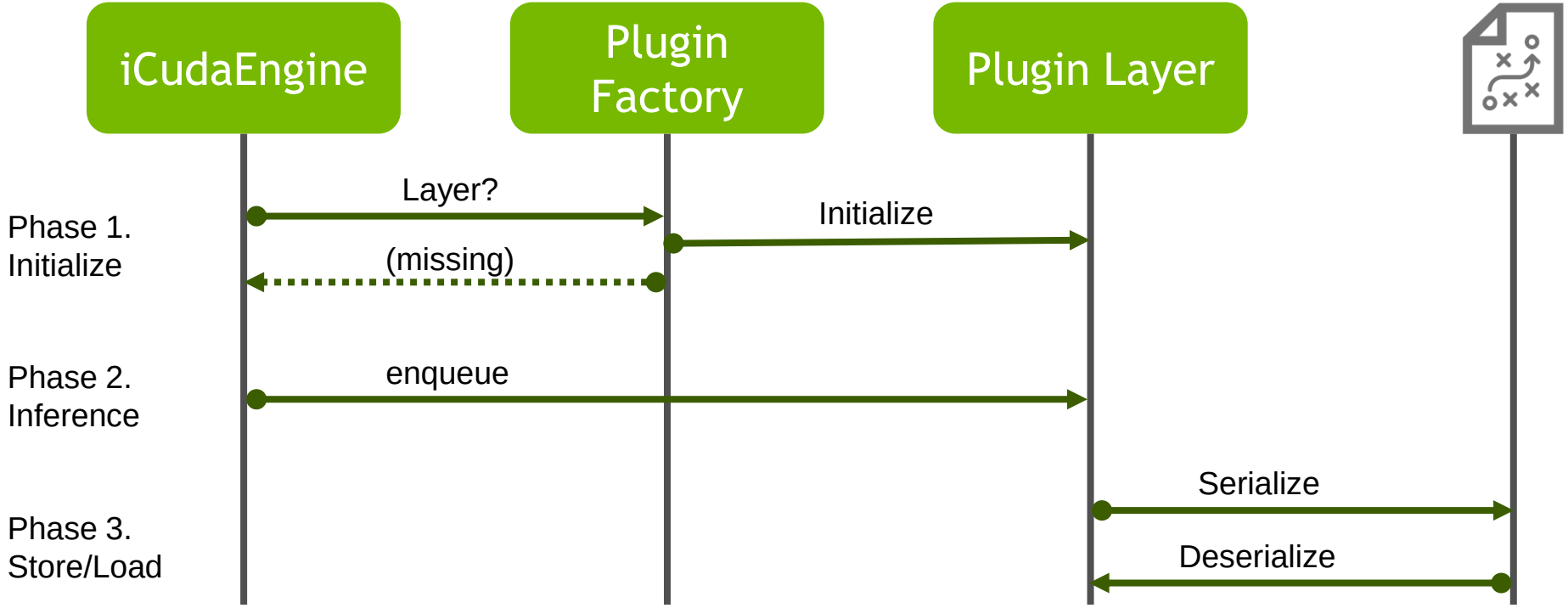
Step 7: Perform inference

```
uff_model = uff.from_tensorflow_frozen_model("frozen_model_file.pb",
                                             OUTPUT_LAYERS)

parser = uffparser.create_uff_parser()


parser.register_input(INPUT_LAYERS[0], (INPUT_C,INPUT_H,INPUT_W),0)
parser.register_output(OUTPUT_LAYERS[0])


engine = trt.utils.uff_to_trt_engine(G_LOGGER,
                                     uff_model,
                                     parser,
                                     INFERENCE_BATCH_SIZE,
                                     1<<20,
                                     trt.infer.DataType.FLOAT)


trt.utils.write_engine_to_file(save_path, engine.serialize())



engine = Engine(PLAN=plan,
                postprocessors={"output_layer_name":post_processing_function})

result = engine_single.infer(image)
```

Custom Layer Support using Plugin Layer



Plugin Layer Procedure



Supporting Features

Feature	C++	Python (x86 only)	NvCaffeParser	NvUffParser
CNNs	Yes	Yes	Yes	Yes
RNNs	Yes	Yes	No	No
INT8 Calibration	Yes	Yes	N/A	N/A
Asymmetric Padding	Yes	Yes	No	No

TensorRT API

Define your network using C/C++ or Python

```
INetworkDefinition* network = builder->createNetwork();

auto data = network->addInput(INPUT_BLOB_NAME, DataType::kFLOAT, Dims2
auto rnn = network->addRNNv2(*data, LAYER_COUNT, HIDDEN_SIZE, SEQ_SIZE

auto fcwts = network->addConstant(Dims2(VOCAB_SIZE, HIDDEN_SIZE), weig
auto matrixMultLayer = network->addMatrixMultiply(*fcwts->getOutput(0)
auto fcbias = network->addConstant(Dims2(VOCAB_SIZE, 1), weightMap[FCB
auto addBiasLayer = network->addElementWise(*matrixMultLayer->getOutpu
auto pred = network->addTopK(*addBiasLayer->getOutput(0), nvinfer1::To
network->markOutput(*pred->getOutput(1));
```

convertRNNWeights, convertRNNBias
setWeightsForGate, setBiasForGate

addReduce (**ITensor** &input, **ReduceOperation** operation, uint32_t reduceAxes, bool keepDimensions)=0
Add a reduce layer to the network. More...

addTopK (**ITensor** &input, **TopKOperation** op, int k, uint32_t reduceAxes)=0
Add a TopK layer to the network. More...

addGather (**ITensor** &data, **ITensor** &indices, int axis)=0
Add a gather layer to the network. More...

addRaggedSoftMax (**ITensor** &input, **ITensor** &bounds)=0
Add a RaggedSoftMax layer to the network. More...

addMatrixMultiply (**ITensor** &input0, bool transpose0, **ITensor** &input1, bool transpose1)=0
Add a MatrixMultiply layer to the network. More...

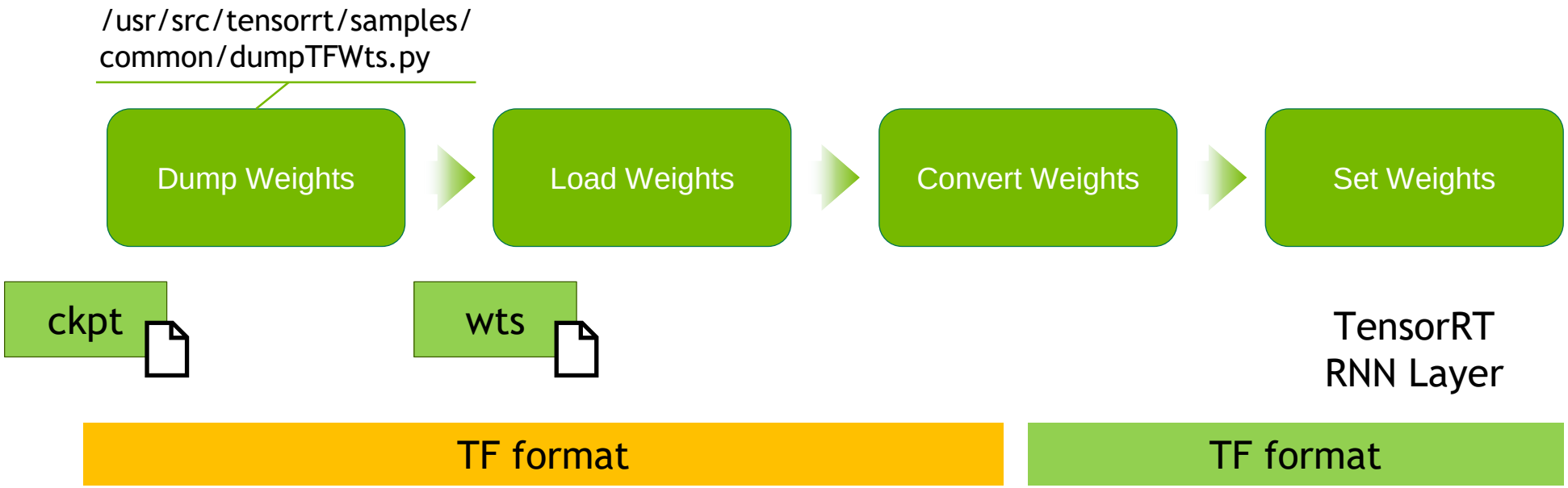
addConstant (**Dims** dimensions, **Weights** weights)=0
Add a constant layer to the network. More...

addRNNv2 (**ITensor** &input, int32_t layerCount, int32_t hiddenSize, int32_t maxSeqLen, **RNNOperation** op)=0
Add an layerCount deep RNN layer to the network with hiddenSize internal states that can take a batch with fixed or variable sequence lengths. More...

addPluginExt (**ITensor** *const *inputs, int nbInputs, **IPluginExt** &plugin)=0
Add a plugin layer to the network using an **IPluginExt** interface. More...

addIdentity (**ITensor** &input)=0
Add an identity layer. More...

TensorRT RNN Programming



Get started with sampleCharRNN in documentation

NMT with Transformer Inference

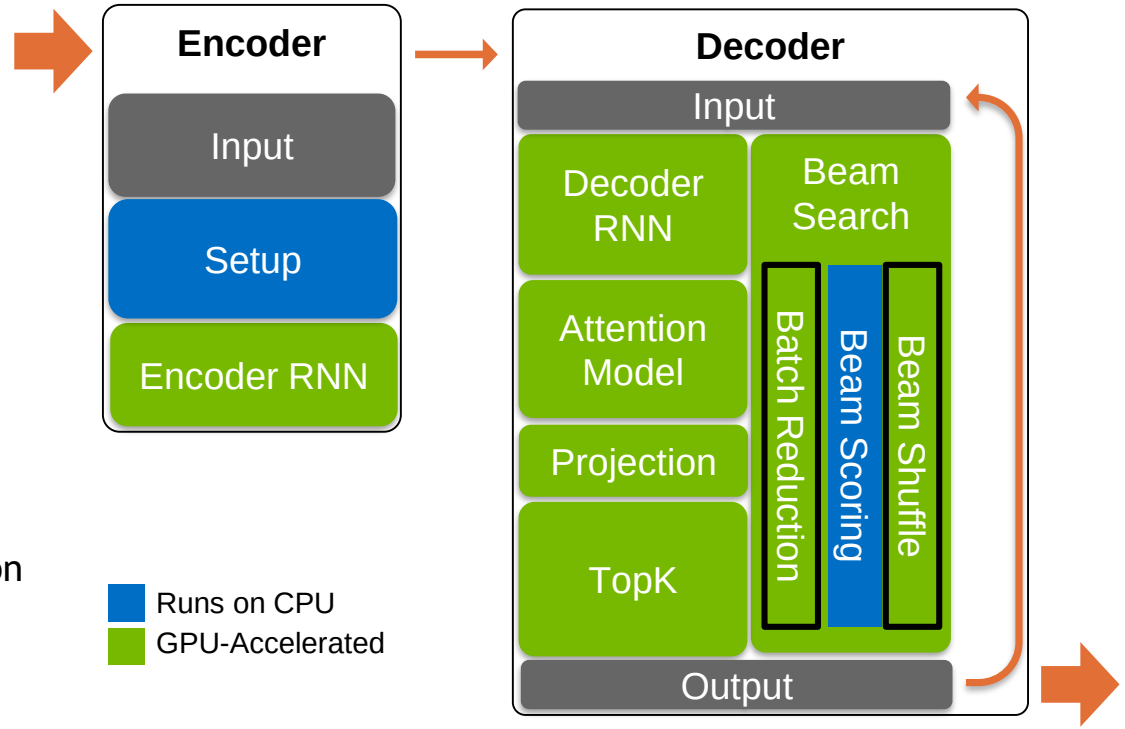
Deploy highly-optimized language translation apps in production environments

Running 3 binded engines
(Encoder, Generator, Suffle Engine)

Modular Network Merge

Support NMT layers such as
Gather, Softmax, Batch GEMM and Top K

Get started with NMT sample in documentation



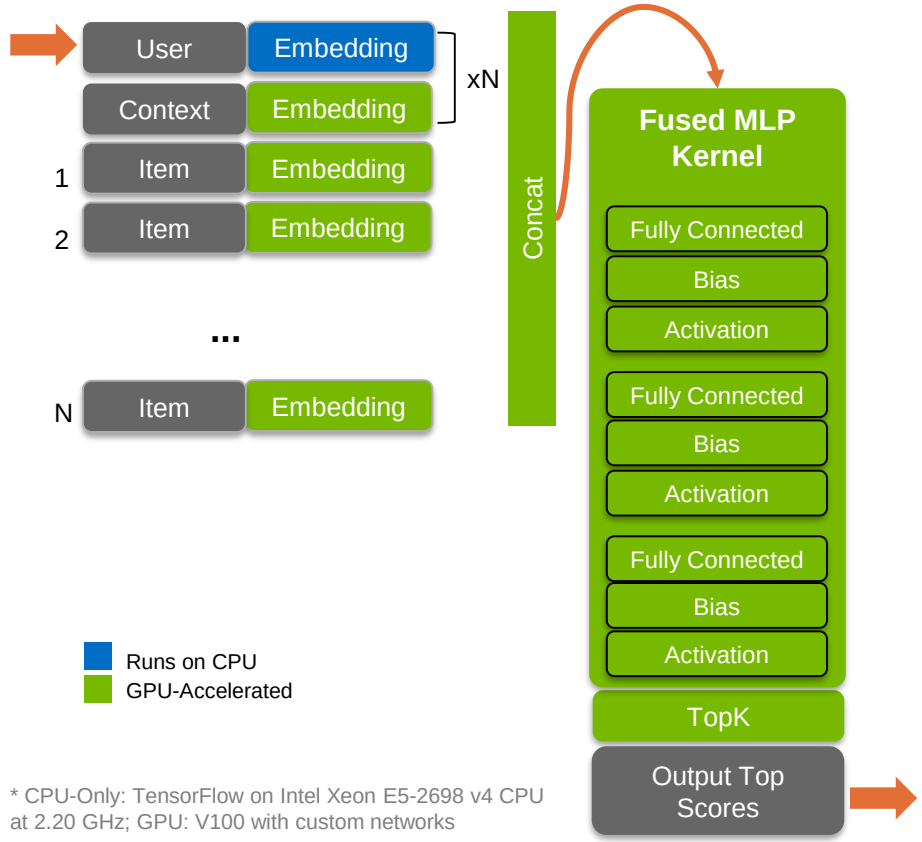
* CPU-Only: TensorFlow on SKL 6140 18 core, FP32
GPU: V100, TensorRT 5, FP16; Sorted data, Batch=128, English to German

Recommendation Inference

Deploy multi-layer perceptron (MLP) based recommendation apps in production

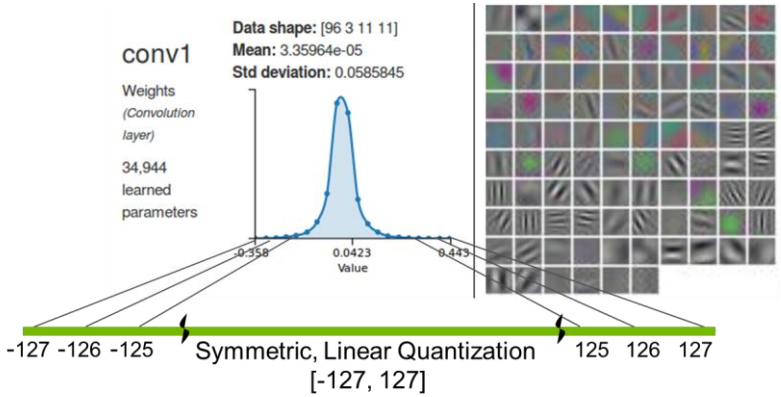
Predict events (click, purchase, interest) accurately based on input items (user, query, observed activity)

Get started with examples on MNIST character recognition and movie recommender system in the documentation



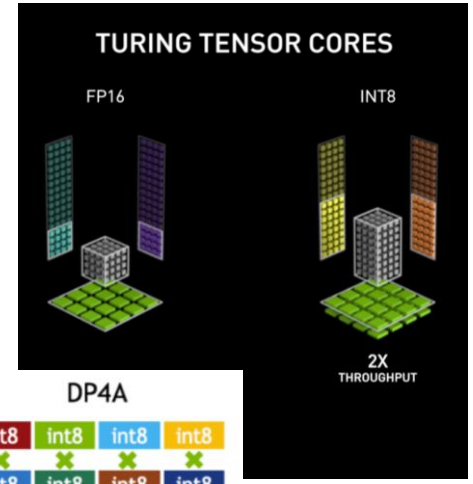
* CPU-Only: TensorFlow on Intel Xeon E5-2698 v4 CPU at 2.20 GHz; GPU: V100 with custom networks

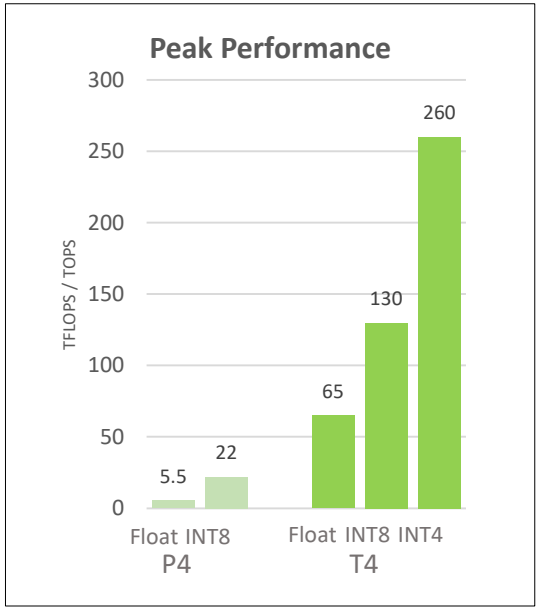
Inferencing with Low precision



I8_weight = Round_to_nearest_int(scaling_factor * F32_weight)

scaling_factor = 127.0f / max(abs(all_F32_weights_in_the_filter))



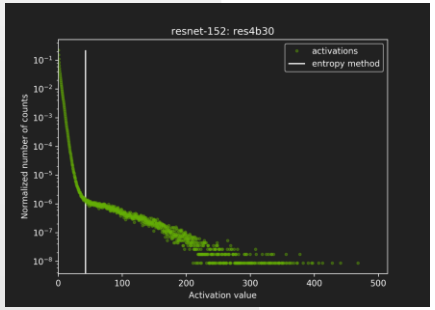


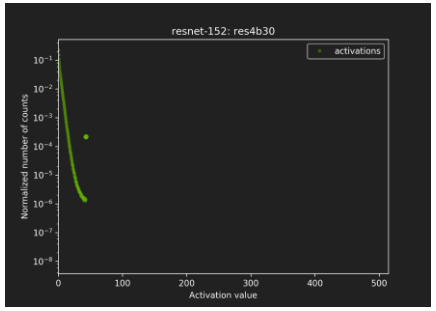
Post Training Quantization

Minimize information loss between FP32 and INT8 inference on calibration dataset

```
calibration_files = create_calibration_dataset(calib_dataset_file)
    # Process 10 images at a time for calibration
    # This batch size can be different from MaxBatchSize (1 in this example)
batchstream = calibrator.ImageBatchStream(10, calibration_files, sub_mean_chw)
int8_calibrator = calibrator.PythonEntropyCalibrator(["input_1"], batchstream)
```

100 mini-batches are sufficient to determine quantization parameter (~500 samples)






```
engine = trt.lite.Engine(framework="uff",
                         path=uff_file,
                         max_batch_size=batch_size,
                         max_workspace_size=(256 << 20),
                         input_nodes={"input_1": input_shape},
                         output_nodes=[outputlayer_name],
                         preprocessors={"input_1": sub_mean_chw},
                         postprocessors={"activation_3/Sigmoid": postprocessing},
                         data_type=trt.infer.DataType.INT8,
                         calibrator=int8_calibrator,
                         #profiler=G_PROFILER,
                         logger_severity=trt.infer.LogSeverity.INFO)
#Serialize to .plan
engine.save(serialized_plan_int8)
```

New INT8 APIs and Optimizations

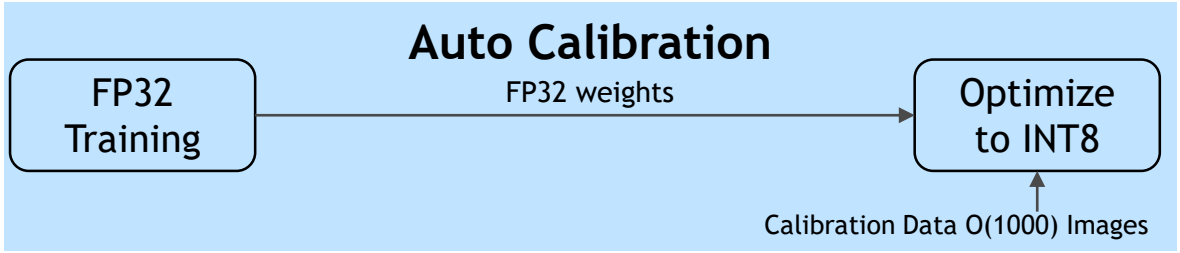
Maximize throughput at low latency with mixed precision compute in production

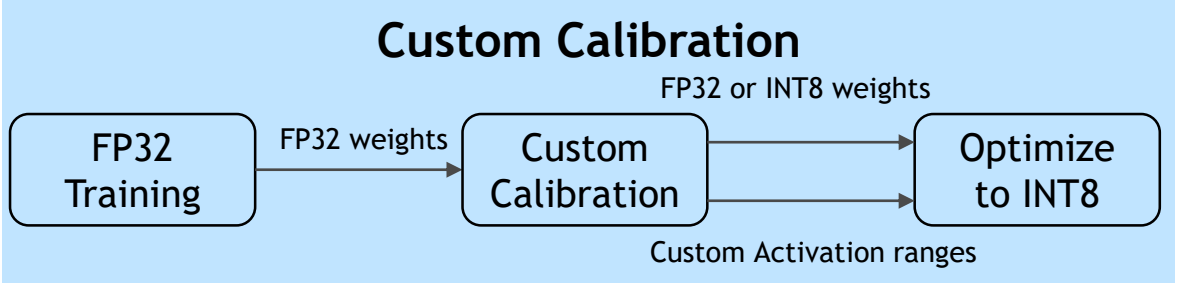
Apply INT8 quantization aware training or custom calibration algorithms with new APIs

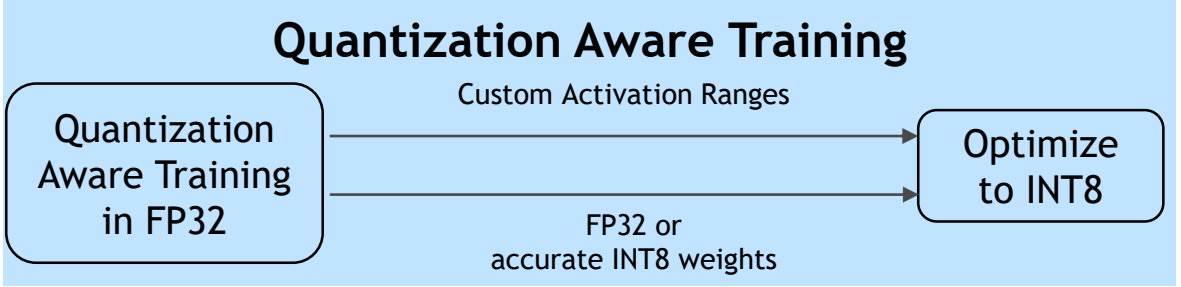
Control precision per-layer with new APIs

Optimizations for depth wise convolution operation

Getting Start with INT8 sample in the documentation

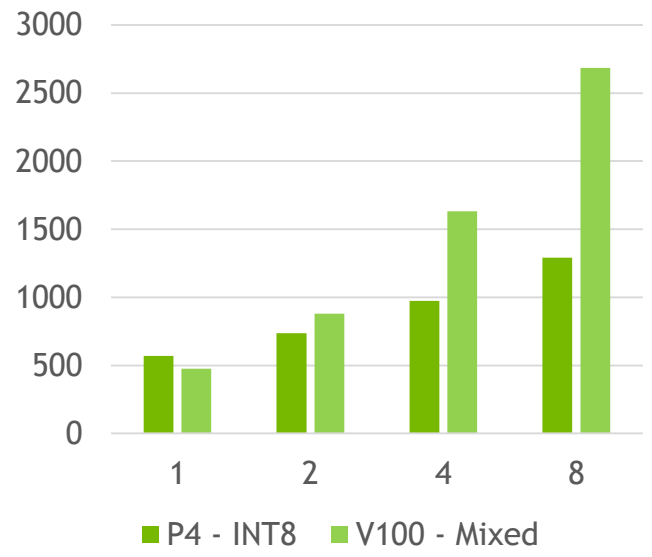






Inference Performance

Throughput



Performance/WATT

WIDELY ADOPTED

Baidu 百度 | BOSCH | Continental | DAIMLER | DarwinAI | DEEPMAP | Ford

here | 科大讯飞 iFLYTEK | IBM | JD.COM | 金山云 | AURORA | MAN

navya | SAP | SEARIDGE TECHNOLOGIES | SK telecom | 图森 tuSimple | verizon

VOLVO | wrnch AI | 欢聚时代 YY Inc. | ZENRIN | ZF | ZOOX

NVIDIA INFERENCE MOMENTUM

clarifai

Image Tagging

VALOESA

Video Analysis

SAP

Advertising Impact

Rakuten

Video Captioning

accenture

Cybersecurity

ebay

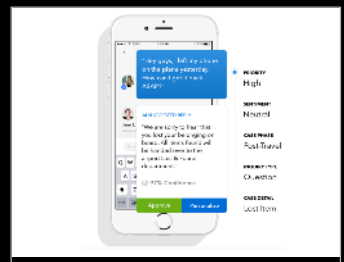
Visual Search

SHAZAM

Finding Music

Ford

Sports Performance



KLM

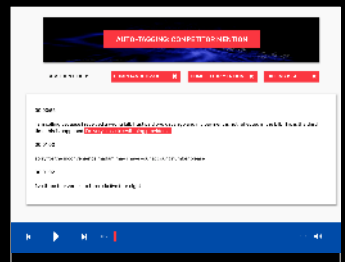
Customer Service

Bing

Visual Search

Avitas Systems
a GE venture

Industrial Inspection



DEEPGRAM

Voice Recognition

“ Using GPUs made it possible to **enable media understanding of the Twitter platform**, by drastically reducing media deep learning model training time and by allowing us to **derive real-time understanding of live videos** at inference time. ”

Nicolas Koumchatzky, Head of Twitter Cortex

Bing

“ At Microsoft, we are working hard to deliver the most advanced AI-powered services to our customers. Using NVIDIA GPUs in real-time inference workloads has improved Bing’s advanced search offerings, enabling us to reduce object detection latency for images. We look forward to working with NVIDIA’s next generation inference hardware and software to expand the way people benefit from AI products and services.”

-Jordi Ribas CVP, Bing and AI Products, Microsoft

Google Cloud

“ AI is becoming increasingly pervasive, and inference is a critical capability customers need to successfully deploy their AI models, so we’re excited to support NVIDIA’s Turing Tesla T4 GPUs on Google Cloud Platform soon.”

-Chris Kleban, product manager at Google Cloud

NVIDIA | AI CONFERENCE

SEOUL | NOVEMBER 7 - 8,2018

www.nvidia.com/ko-kr/ai-conference/