Deployment of ML Models

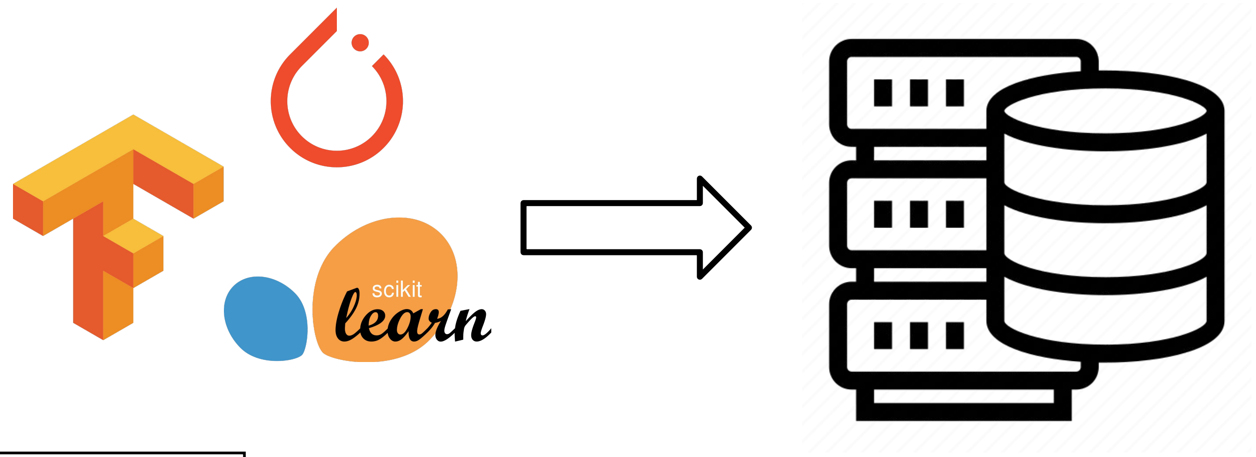
Marcin Walkowski 2022

Plan for today

- Model Deployment
  - Persistence
  - Optimization
  - Serving
- Organizational - projects and plans for the upcoming months

Model Deployment

Model Deployment



Persistence

Persistence

- We want to persist our trained model

```python
# TF2 code
# Save the weights
model.save_weights('./model_weights/my_model_weights')

# Create a new model instance
model = create_model()

# Restore the weights and evaluate
model.load_weights('./model_weights/my_model_weights')
loss, acc = model.evaluate(test_images, test_labels, verbose=2)
```

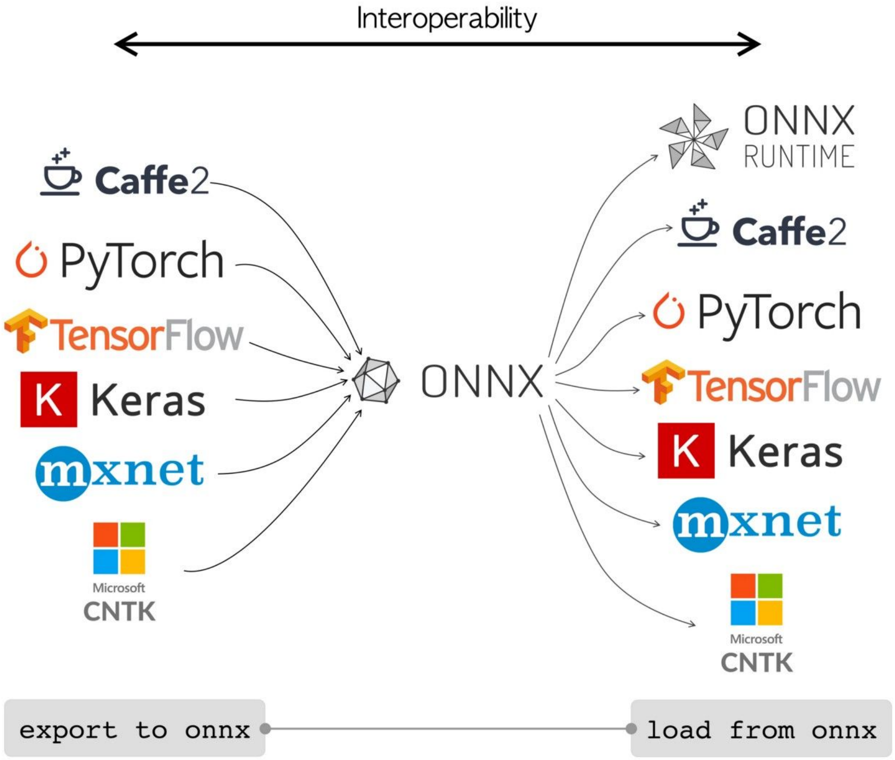
```python
# TF2 code
# Save the entire model as a SavedModel.
model.save('models/my_model')

# Load the entire model and evaluate
new_model = tf.keras.models.load_model('saved_model/my_model')
loss, acc = model.evaluate(test_images, test_labels, verbose=2)
```

Persistence - formats

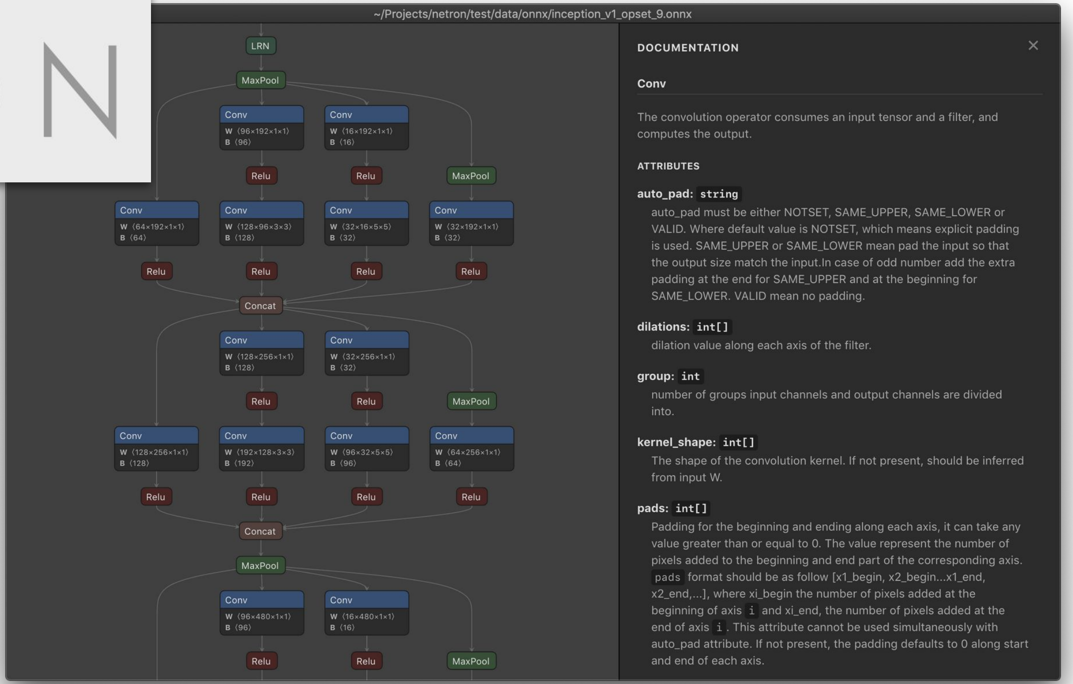
Persistence – **O**pen **N**eural **N**etwork **E**xchange

Persistence - **O**pen **N**eural **N**etwork **E**xchange



Source: https://towardsdatascience.com/onnx-preventing-framework-lock-in-9a798fb34c92

Persistence – visualization



Source:
https://medevel.com/netron-open-source-visualizer-for-deep-learning-machine-learning-and-neural-network-models/

Persistence - checkpoints

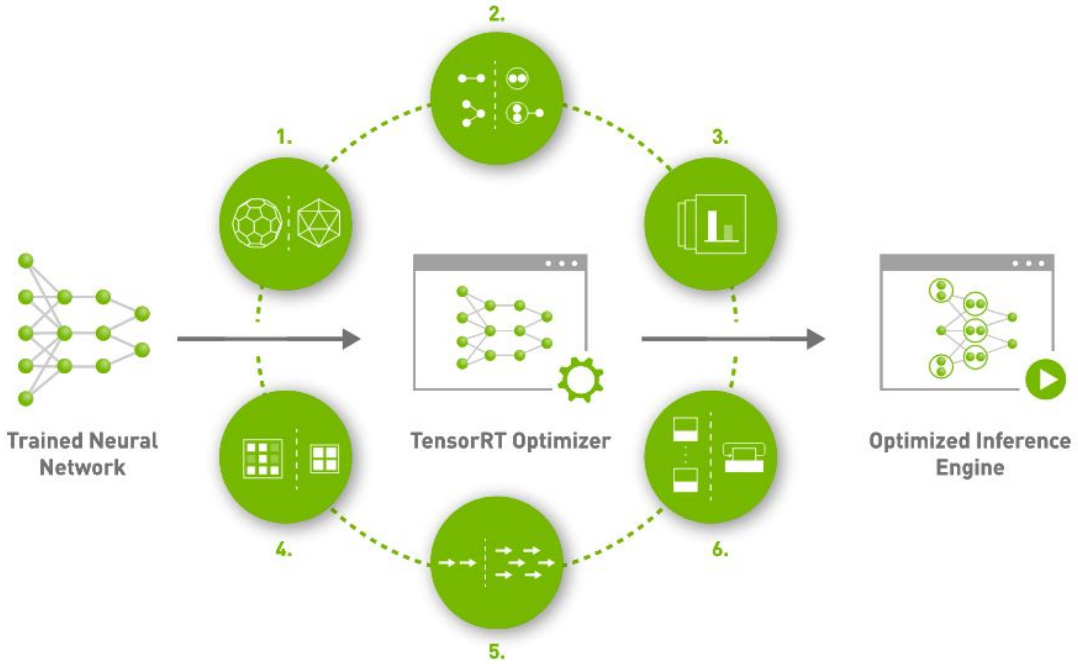
```
# TF2 code
# Create a callback that saves the model's weights
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath='path/to/checkpoints',
                                                 save_weights_only=True,
                                                 verbose=1)

# Train the model with the new callback
model.fit(train_images,
          train_labels,
          epochs=10,
          validation_data=(test_images, test_labels),
          callbacks=[cp_callback])
```

Optimization

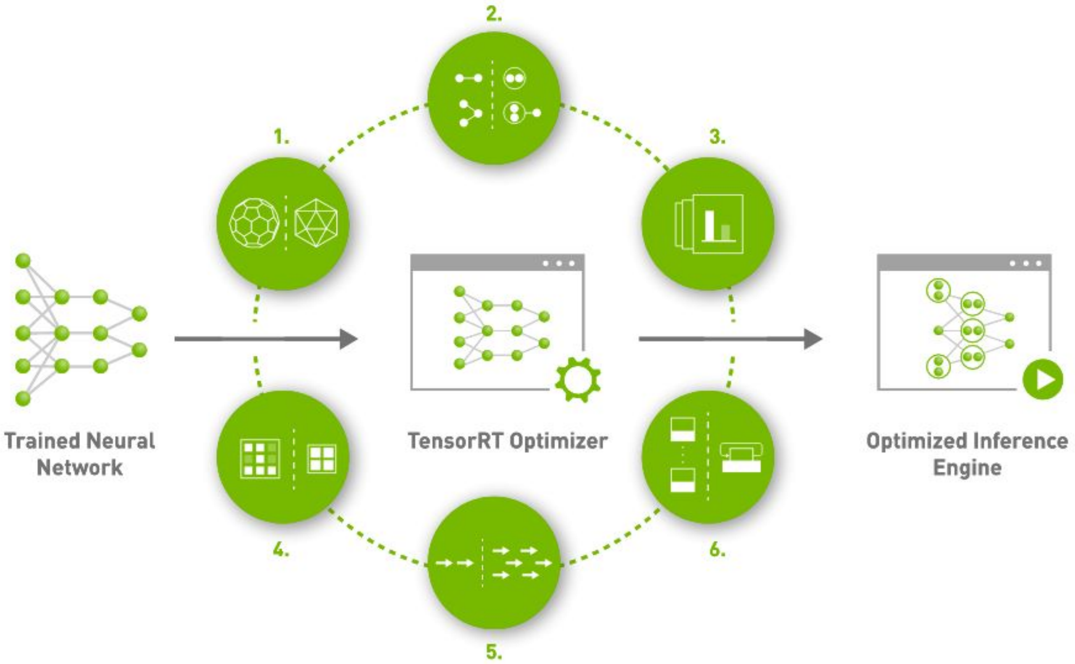
Optimization - NVIDIA TensorRT

- Reduced Precision
- Layer and Tensor Fusion
- Kernel Auto-Tuning
- Dynamic Tensor Memory
- Multi-Stream Execution
- Time Fusion



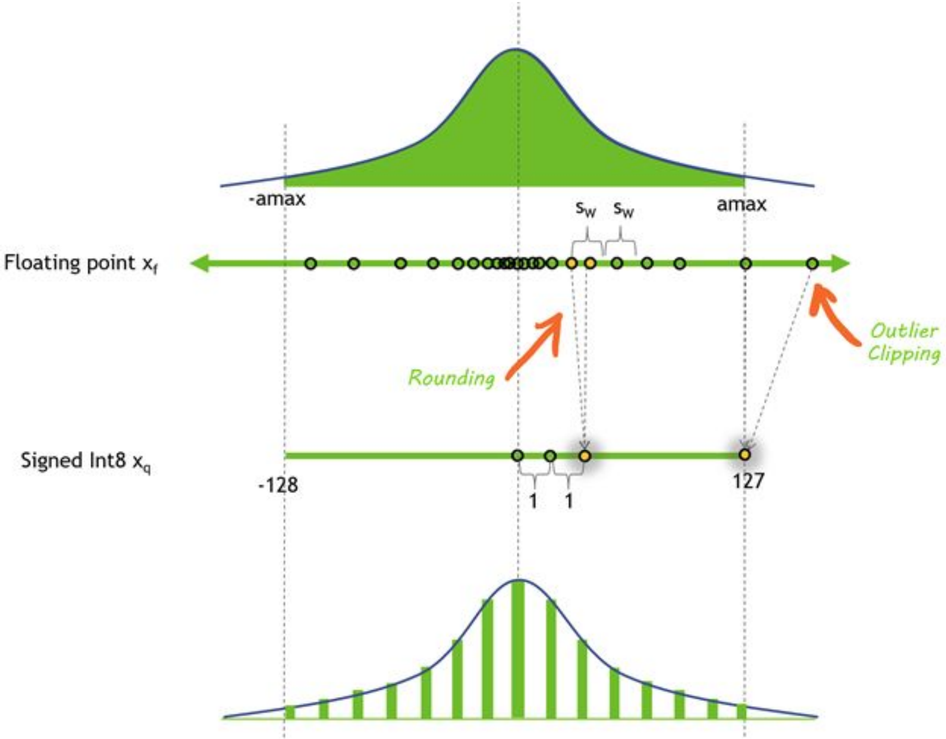
Optimization - NVIDIA TensorRT

- **Reduced Precision**
- Layer and Tensor Fusion
- Kernel Auto-Tuning
- Dynamic Tensor Memory
- Multi-Stream Execution
- Time Fusion



Optimization - Reduced Precision

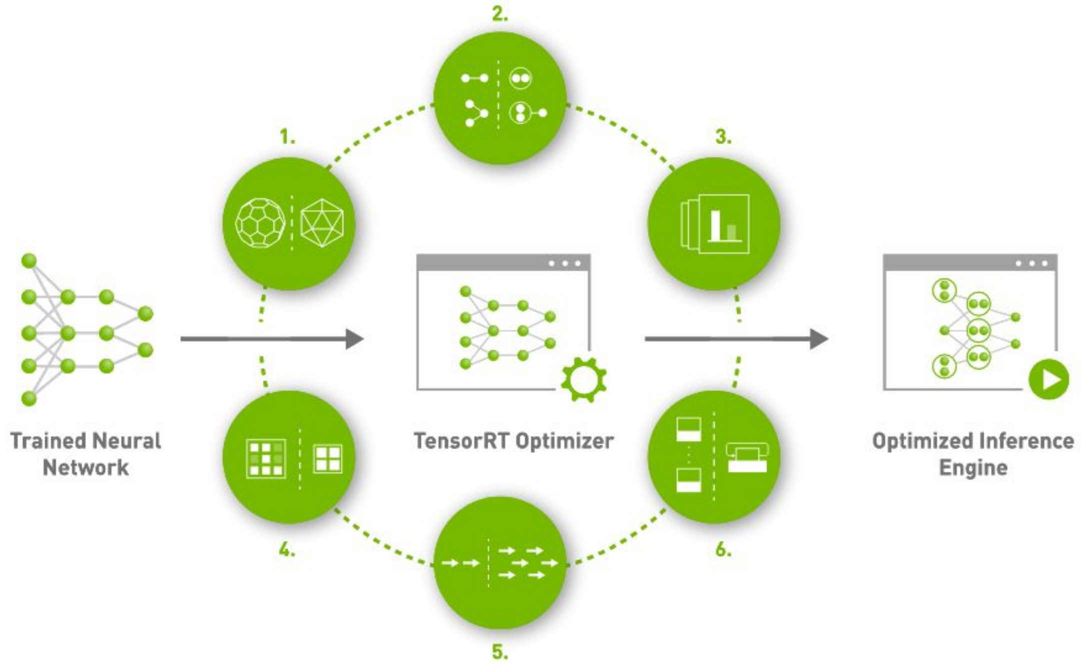
- FP16 is fast
- INT8 may be faster



Source: https://developer.nvidia.com/blog/achieving-fp32-accuracy-for-int8-inference-using-quantization-aware-training-with-tensorrt/

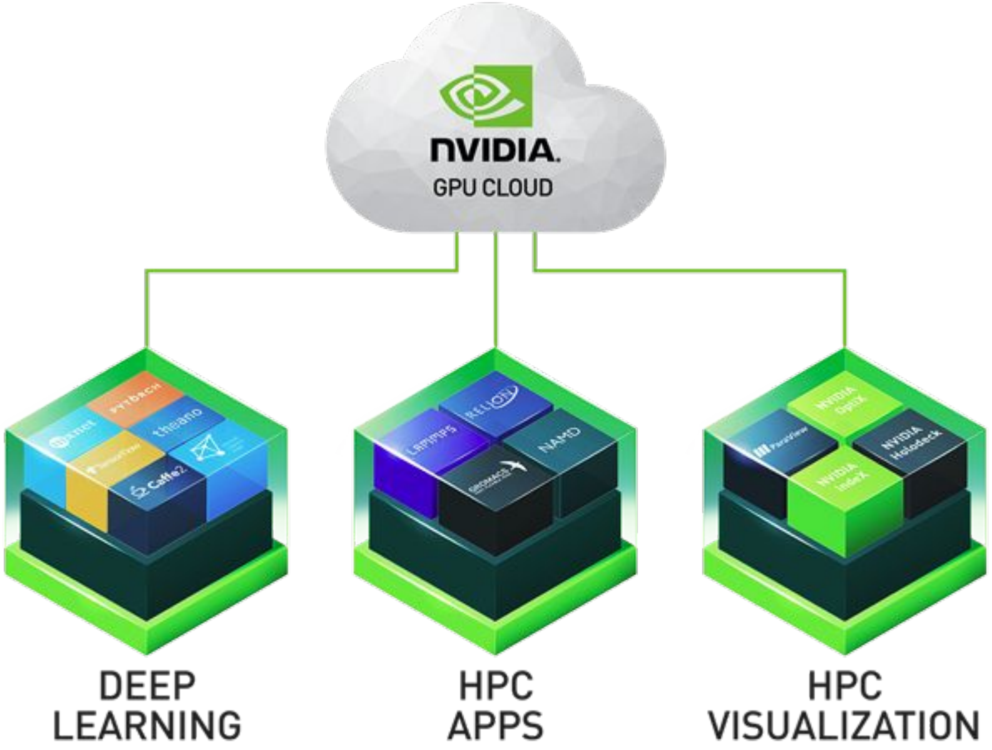
Optimization - NVIDIA TensorRT

- Reduced Precision
- Layer and Tensor Fusion
- Kernel Auto-Tuning
- Dynamic Tensor Memory
- Multi-Stream Execution
- Time Fusion



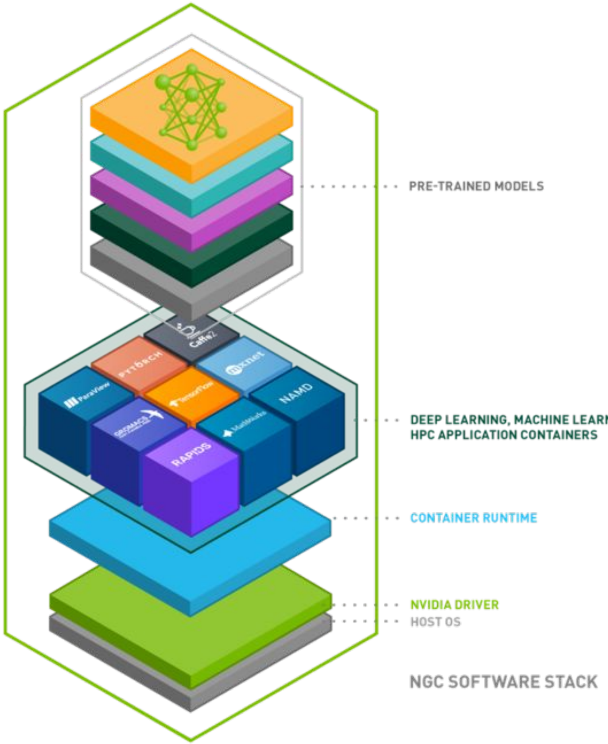
NVIDIA GPU Cloud

- https://ngc.nvidia.com/



NVIDIA GPU Cloud

- https://ngc.nvidia.com/



Deployment

Deployment - service

- Easy solution
- Problematic to scale

```python
# Python-like code
from flask import Flask
import tensorflow as tf

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json['data']

    model = tf.keras.models.load_model('saved_models/my_model')

    prediction = model.predict(data)

    return jsonify({'prediction': list(prediction)})


if __name__ == '__main__':
    app.run(port=8080)
```

Deployment - NVIDIA Triton Inference Server

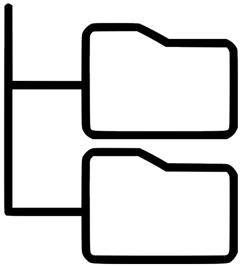
- AI Inference Server
- Open-source software
- Can be deployed locally and on cloud
- GPU and CPU inference

NVIDIA Triton - structure

- Software - source code/NGC container
- Model repository

NVIDIA Triton - structure

- Software - source code/NGC container
- **Model repository**



NVIDIA Triton – model repo

```
<model-repository-path>/
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  ...
```

NVIDIA Triton - model repo

```
<model-repository-path>/
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  ...
```

NVIDIA Triton - model config

- For some models can be auto-generated
- Protocol Buffers format

```
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 ]
  }
]
```

NVIDIA Triton – model config – instance group

```
instance_group [
  {
    count: 2
    kind: KIND_CPU
  }
]
```

```
instance_group [
  {
    count: 1
    kind: KIND_GPU
    gpus: [ 0 ]
  },
  {
    count: 2
    kind: KIND_GPU
    gpus: [ 1, 2 ]
  }
]
```

```
instance_group [
  {
    count: 1
    kind: KIND_GPU
    gpus: [ 0, 1, 2 ]
    rate_limiter {
      resources [
        {
          name: "R1"
          count: 4
        },
        {
          name: "R2"
          global: True
          count: 2
        }
      ]
      priority: 2
    }
  }
]
```

NVIDIA Triton - model config - optimization

```
optimization { execution_accelerators {
  gpu_execution_accelerator : [ {
    name : "tensorrt"
    parameters { key: "precision_mode" value: "FP16" }
    parameters { key: "max_workspace_size_bytes" value: "1073741824" }
    }]
}}
```

```
dynamic_batching {
  preferred_batch_size: [ 4, 8 ]
  max_queue_delay_microseconds: 100
}
```

```
optimization { execution_accelerators {
  cpu_execution_accelerator : [ {
    name : "openvino"
  }]
}}
```

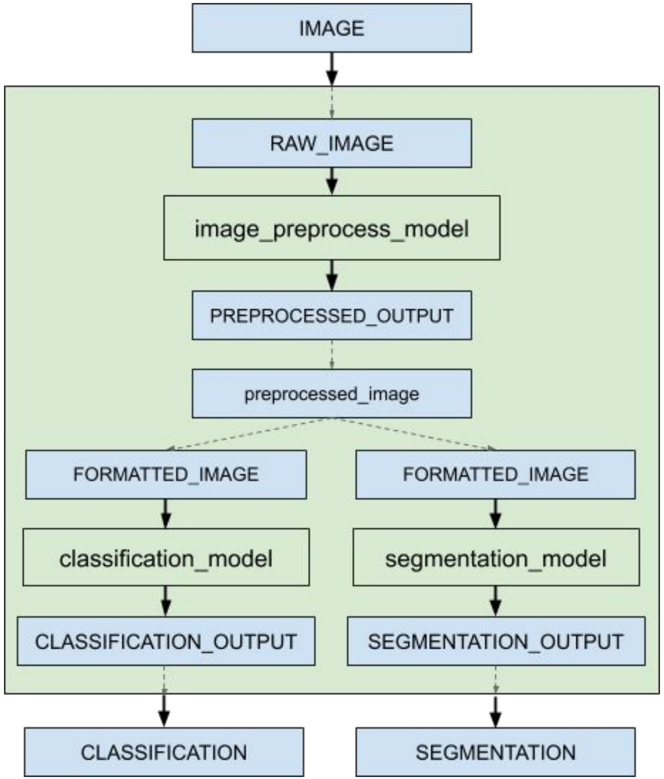
NVIDIA Triton - model config - optimization

```
optimization { execution_accelerators {
  gpu_execution_accelerator : [ {
    name : "tensorrt"
    parameters { key: "precision_mode" value: "FP16" }
    parameters { key: "max_workspace_size_bytes" value: "1073741824" }
    }]
}}
```

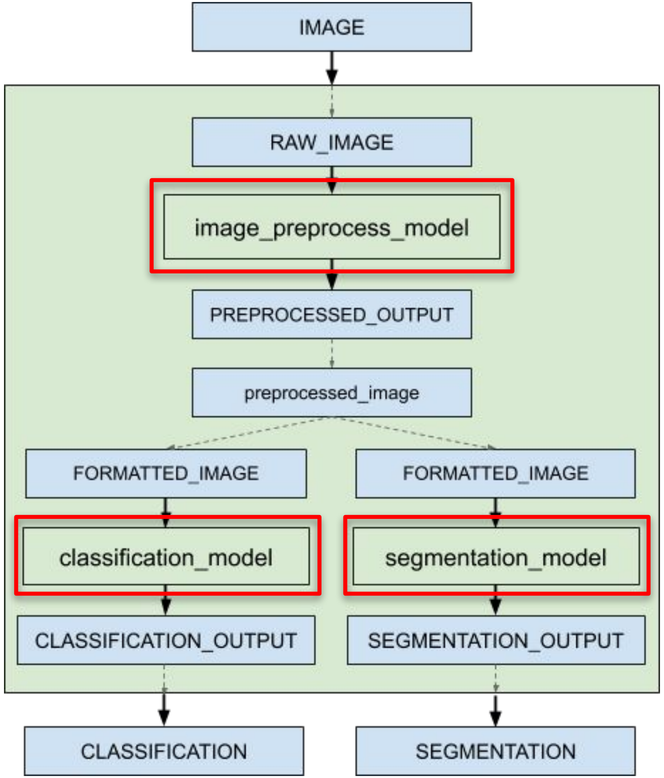
```
dynamic_batching {
  preferred_batch_size: [ 4, 8 ]
  max_queue_delay_microseconds: 100
}
```

```
optimization { execution_accelerators {
  cpu_execution_accelerator : [ {
    name : "openvino"
  }]
}}
```

NVIDIA Triton - model config - ensemble



NVIDIA Triton - model config - ensemble



NVIDIA Triton – benchmarks and metrics

Category	Metric	Description	Granularity	Frequency
GPU Utilization	Power Usage	GPU instantaneous power	Per GPU	Per second
	Power Limit	Maximum GPU power limit	Per GPU	Per second
	Energy Consumption	GPU energy consumption in joules since Triton started	Per GPU	Per second
	GPU Utilization	GPU utilization rate (0.0 - 1.0)	Per GPU	Per second
GPU Memory	GPU Total Memory	Total GPU memory, in bytes	Per GPU	Per second
	GPU Used Memory	Used GPU memory, in bytes	Per GPU	Per second
Count	Request Count	Number of inference requests received by Triton (each request is counted as 1, even if the request contains a batch)	Per model	Per request
	Inference Count	Number of inferences performed (a batch of "n" is counted as "n" inferences)	Per model	Per request
	Execution Count	Number of inference batch executions (see Count Metrics)	P	
Latency	Request Time	Cumulative end-to-end inference request handling time	P	
	Queue Time	Cumulative time requests spend waiting in the scheduling queue	P	
	Compute Input Time	Cumulative time requests spend processing inference inputs (in the framework backend)	P	
	Compute Time	Cumulative time requests spend executing the inference model (in the framework backend)	P	
	Compute Output Time	Cumulative time requests spend processing inference outputs (in the framework backend)	P	

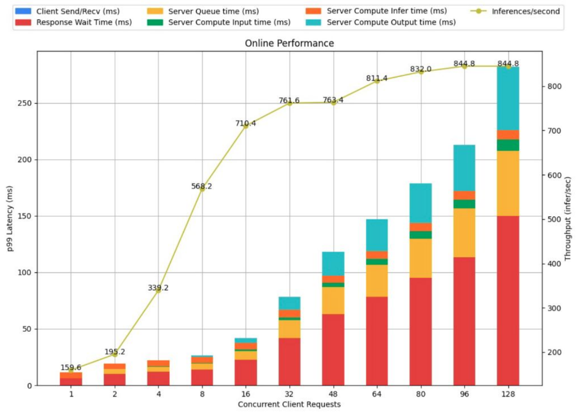
```
Usage: perf_analyzer [options]
==== SYNOPSIS ====

        --service-kind <"triton"|"tfserving"|"torchserve"|"triton_c_api">
        -m <model name>
        -x <model version>
        --model-signature-name <model signature name>
        -v

I. MEASUREMENT PARAMETERS:
        --async (-a)
        --sync
        --measurement-interval (-p) <measurement window (in msec)>
        --concurrency-range <start:end:step>
        --request-rate-range <start:end:step>
        --request-distribution <"poisson"|"constant">
        --request-intervals <path to file containing time intervals in microseconds>
        --binary-search
        --num-of-sequences <number of concurrent sequences>
        --latency-threshold (-l) <latency threshold (in msec)>
        --max-threads <thread counts>
        ...shold for stable measurement (in percentage)>
        ...rements for each profiling>
```

Detailed Report

Model Config: resnet50_libtorch_i0

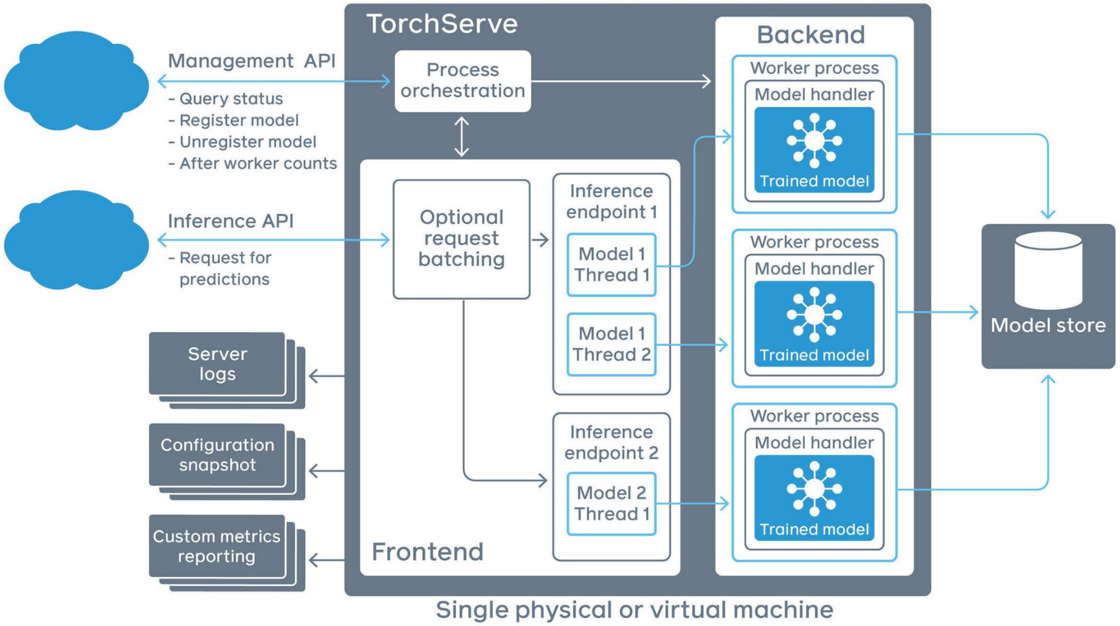


NVIDIA Triton – APIs and clients

- HTTP/REST
- gRPC
- Shared memory
- C API

Deployment - other solutions

- TorchServe
- Tensorflow Server
- BentoML
- Coretex
- KFServing
- ...



Deployment - edge devices

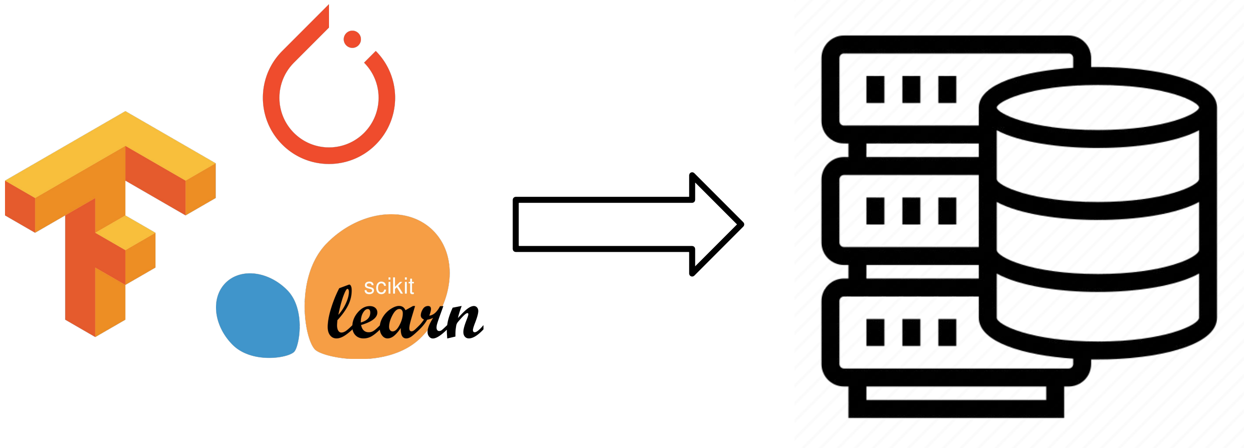
TensorFlow Lite

OpenVINO™


```python
# TF2 code
# Convert the model
converter = tf.lite.TFLiteConverter.from_saved_model('path/to/model')
tflite_model = converter.convert()

# Save the model
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Model Deployment



Resources

- NVIDIA GPU Cloud
- Coursera - MLOps Specialization

Resources

- NVIDIA GPU Cloud
- Coursera - MLOps Specialization

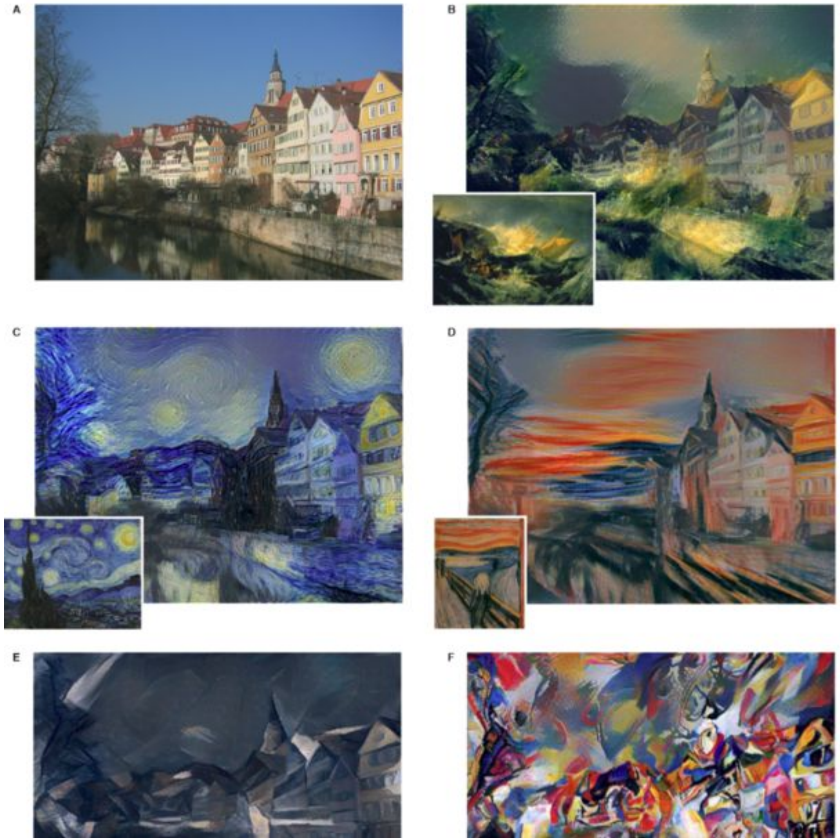
Thank you and Q&A

Działalność Koła Gradient w nadchodzących miesiącach

- Na okres końca stycznia-lutego regularne spotkania Gradientu zostają zawieszone. Będą odbywały się niezależne spotkania grup projektowych.
- Od marca planujemy rozpocząć prelekcji naszych członków/pracowników uczelni/gości z firm.
- 21 marca 2022 weźmiemy udział w Forum Organizacji i Kół Akademickich.

Pomysł na projekt - Real Time Neural Style Transfer

Source: https://medium.com/@chimezie.iwuanyanwu/real-time-style-transfer-caffa3393833



Pomysł na projekt - Real Time Neural Style Transfer

- Eksperymenty w celu poprawy jakości rezultatów
- end2end
  - Wdrożenie na klastrze z użyciem np. NVIDIA Triton
  - Przygotowanie klienta web
- Deadline 21 marca - prezentacja na FOKA
- Grupa projektowa 3-6 osób