深度學習系統與實現

LAB5
Deployment of Trained Models

Dept. of Computer Science and
Information Engineering
National Chiao Tung University

LAB 5

- Dataset: Food11
- Model: Modified Model
- Accuracy: 91+ on evaluation dataset
- Realized with TensorRT

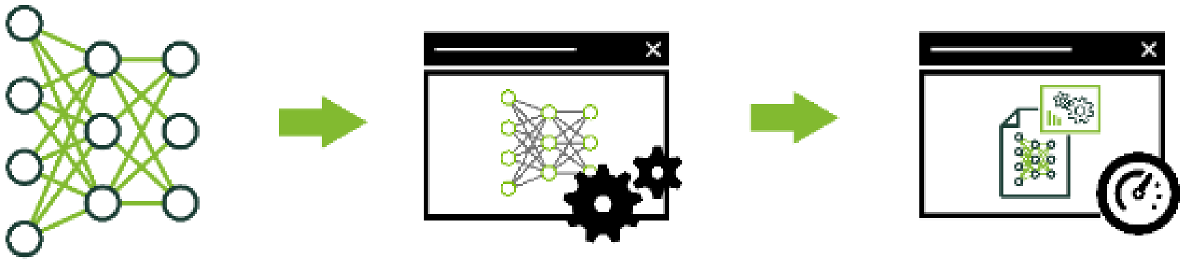
Outline

- Tool
- TensorRT workflow
- LAB 5-1
- LAB 5-2
- Report Spec
- Grading

Tool

- TensorRT
  - Offical Docs: https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html
  - Support two interfaces
    - Python API (you will also need pyCUDA)
    - C++ API
  - Using docker image are highly encouraged
    - https://docs.nvidia.com/deeplearning/sdk/tensorrt-container-release-notes/running.html
    - Be aware of each release's TensorRT and CUDA version

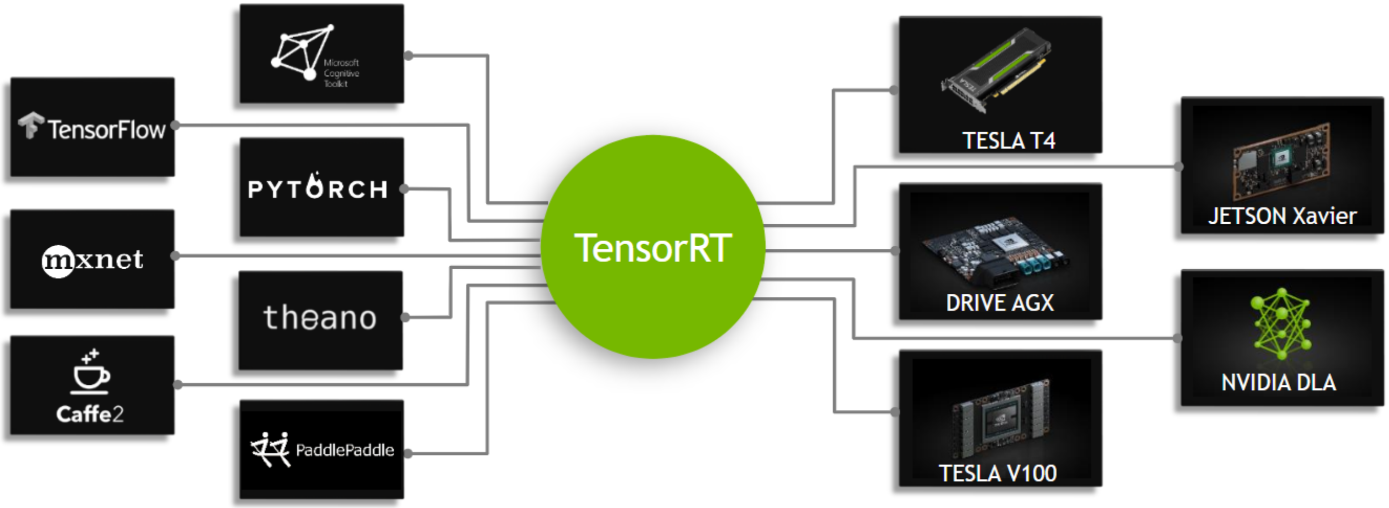
TensorRT Workflow



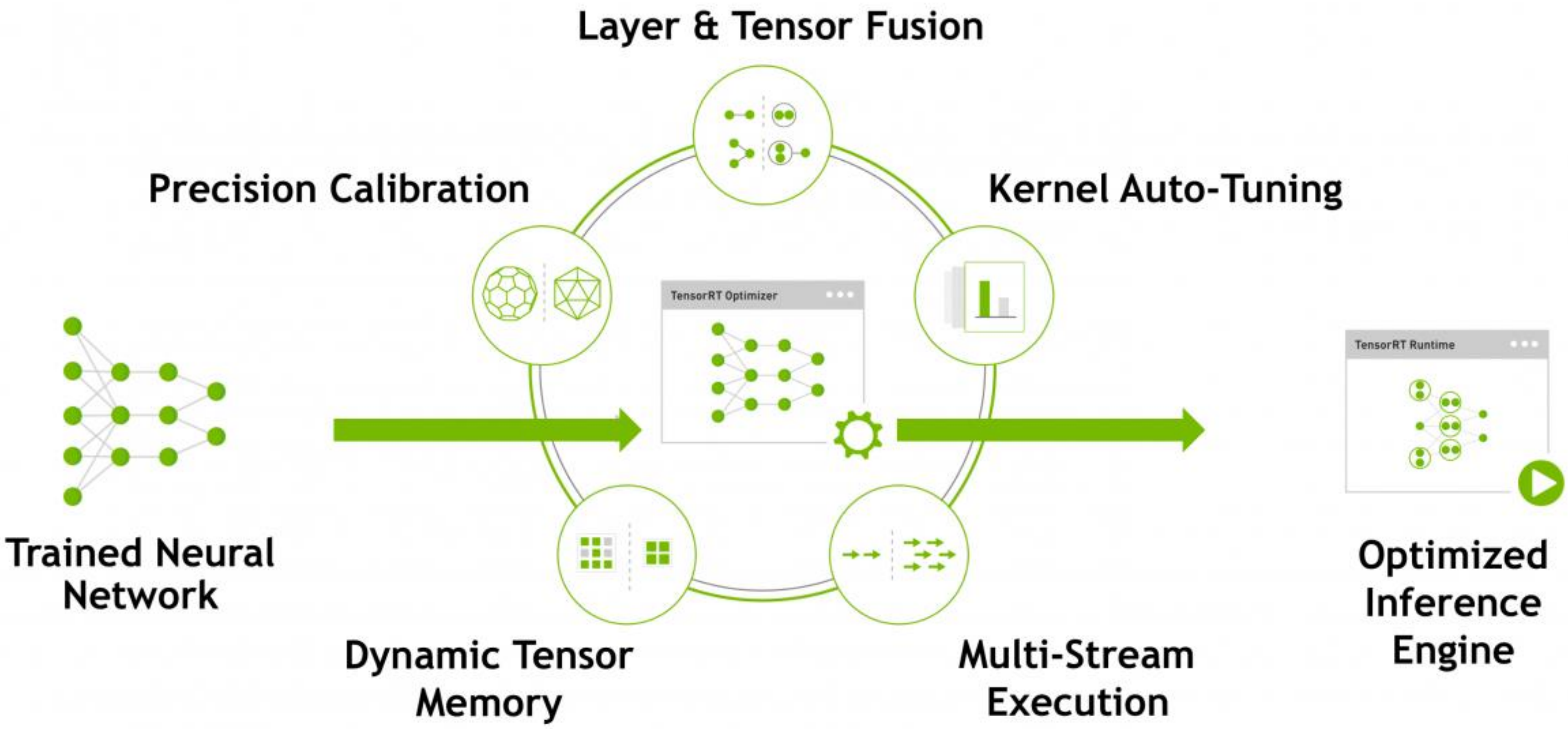
Trained Neural Network

TensorRT Optimizer

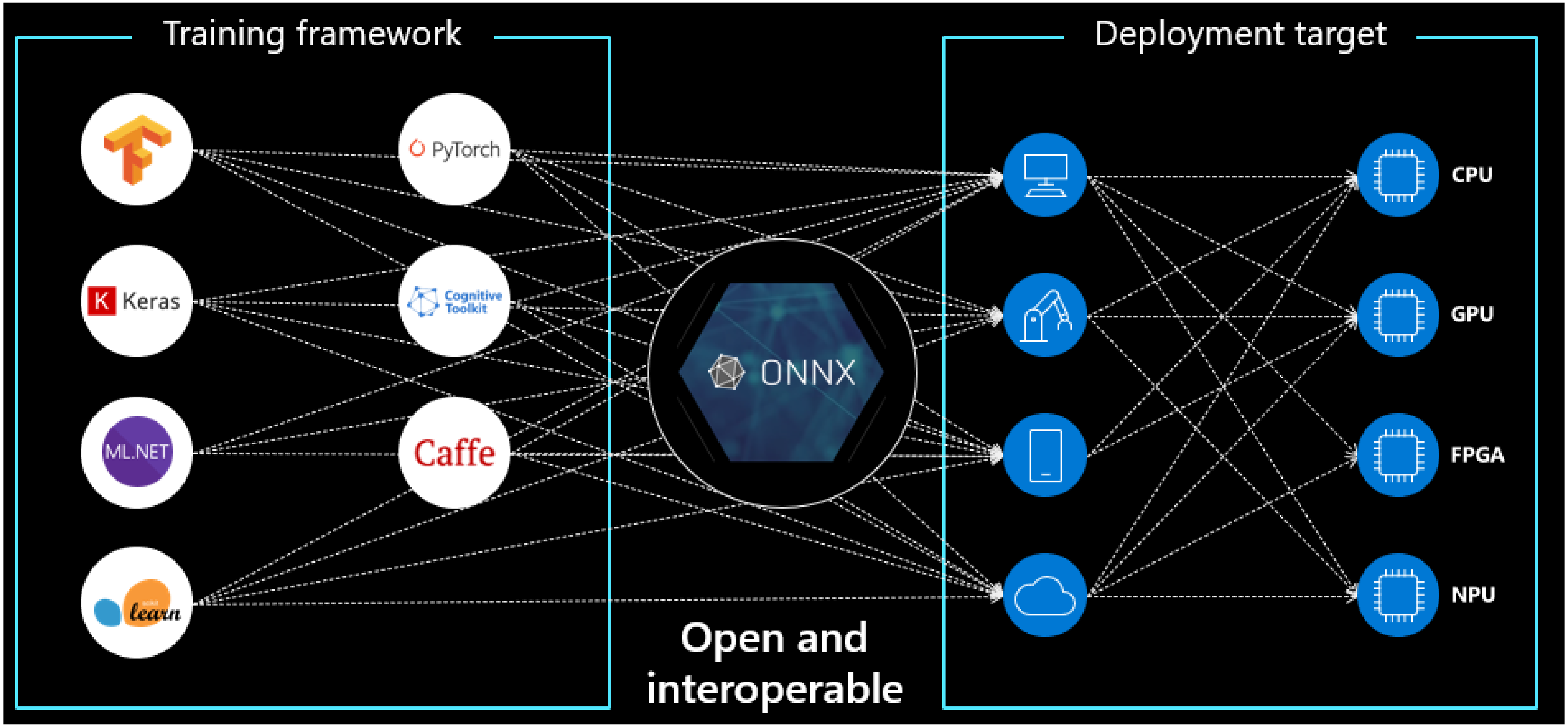
TensorRT Runtime Engine



TensorRT Optimizer



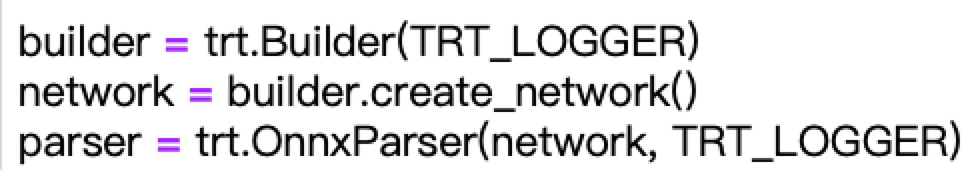
Open Neural Network Exchange (ONNX)



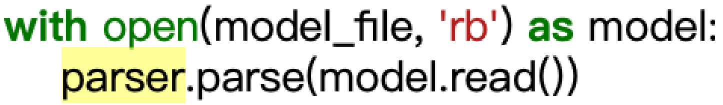
Credit: https://microsoft.github.io/ai-at-edge/docs/onnx/

TensorRT workflow (ONNX Parser)

- Prepare builder, network definition, and parser
  - We use TensorRT ONNX parser here

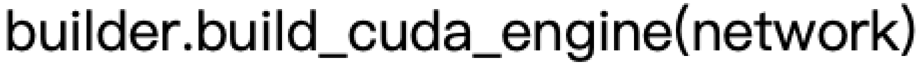


```
builder = trt.Builder(TRT_LOGGER)
network = builder.create_network()
parser = trt.OnnxParser(network, TRT_LOGGER)
```



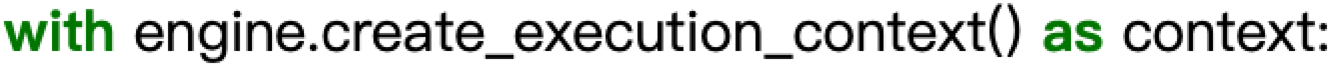
```
with open(model_file, 'rb') as model:
    parser.parse(model.read())
```

- Build engine via the network from ONNX parser



```
builder.build_cuda_engine(network)
```

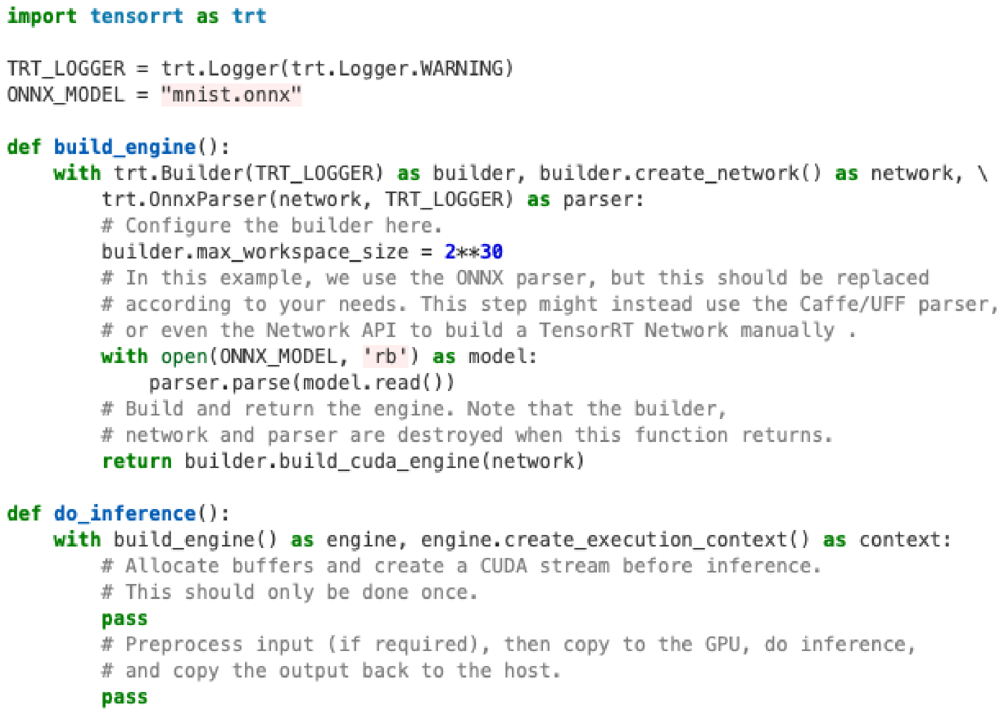
- Create context from engine



```
with engine.create_execution_context() as context:
```

Perform engine on CUDA devices

- □ Prepare Input/output on CPU/GPU
  - Based on test image size and batch size
- □ Copy input to GPU
- □ Perform inference
- □ Copy output to CPU



```python
import tensorrt as trt

TRT_LOGGER = trt.Logger(trt.Logger.WARNING)
ONNX_MODEL = "mnist.onnx"

def build_engine():
    with trt.Builder(TRT_LOGGER) as builder, builder.create_network() as network, \
        trt.OnnxParser(network, TRT_LOGGER) as parser:
        # Configure the builder here.
        builder.max_workspace_size = 2**30
        # In this example, we use the ONNX parser, but this should be replaced
        # according to your needs. This step might instead use the Caffe/UFF parser,
        # or even the Network API to build a TensorRT Network manually .
        with open(ONNX_MODEL, 'rb') as model:
            parser.parse(model.read())
        # Build and return the engine. Note that the builder,
        # network and parser are destroyed when this function returns.
        return builder.build_cuda_engine(network)

def do_inference():
    with build_engine() as engine, engine.create_execution_context() as context:
        # Allocate buffers and create a CUDA stream before inference.
        # This should only be done once.
        pass
        # Preprocess input (if required), then copy to the GPU, do inference,
        # and copy the output back to the host.
        pass
```

Important Parameters

- WorkspaceSize
  - GPU temporary memory which the engine can use at execution time.
- MaxBatchSize
  - The maximum batch size which can be used at execution time, and also the batch size for which the engine will be optimized.

LAB 5-1(50%)

- Train target model to target accuracy
  - Use any technique to train the model to target accuracy
  - Target Accuracy = 91%↑
- Serialize to ONNX
  - Serialize the model to ONNX
  - The serialize detail please reference the link below
    - https://pytorch.org/docs/stable/onnx.html#functions

LAB 5-2(50%)

- Inference on TensorRT
  - Use the ONNX from 5-1
- Batch Size Adjustment
  - Set the batch size to [1, 2, 4, 8, 16, 32, 64]
  - Show the latency and FPS in plot graph

Report

- Environment Setup
  - GPU、CUDA Version、TensorRT Version
  - Baseline Setup(batch size, pre-processing etc...)
- Baseline
  - Accuracy and Inference speed
- Result
  - TensorRT accuracy on evaluation dataset
  - Batch size vs Speedups
- Conclusion and Insights
- Anything you want to say

Grading

- □ LAB 5-1 (50%)
  - □ Train model to target accuracy (30%)
  - □ Serialize to ONNX (20%)
- □ LAB 5-2 (50%)
  - □ Inference on TensorRT (40%)
  - □ Batch size adjustment(10%)
- □ Bonus (10%)
  - □ Further Optimization
    - □ Multiple stream on TensorRT
    - □ Mixed precision inference
    - □ et cetera...
- □ Submission to E3:
  - Source code + report
  - zip format (ex: dllab_lab5_{group id}.zip)
- □ Deadline : 2020/05/25 23:55

Total:

110

Reference

- PyTorch ONNX API
  - https://pytorch.org/docs/stable/onnx.html
- TensorRT:
  - https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html
- TensorRT API Documentation
  - https://docs.nvidia.com/deeplearning/sdk/tensorrt-api/c_api/index.html