DEVCON

Low latency inference for Transformers on Cloud

Sheik Mohamed Imran
AI Solutions Specialist - Intel
25-26 March, 2022

> In deep learning applications, inference accounts for up to 90% of compute cost.

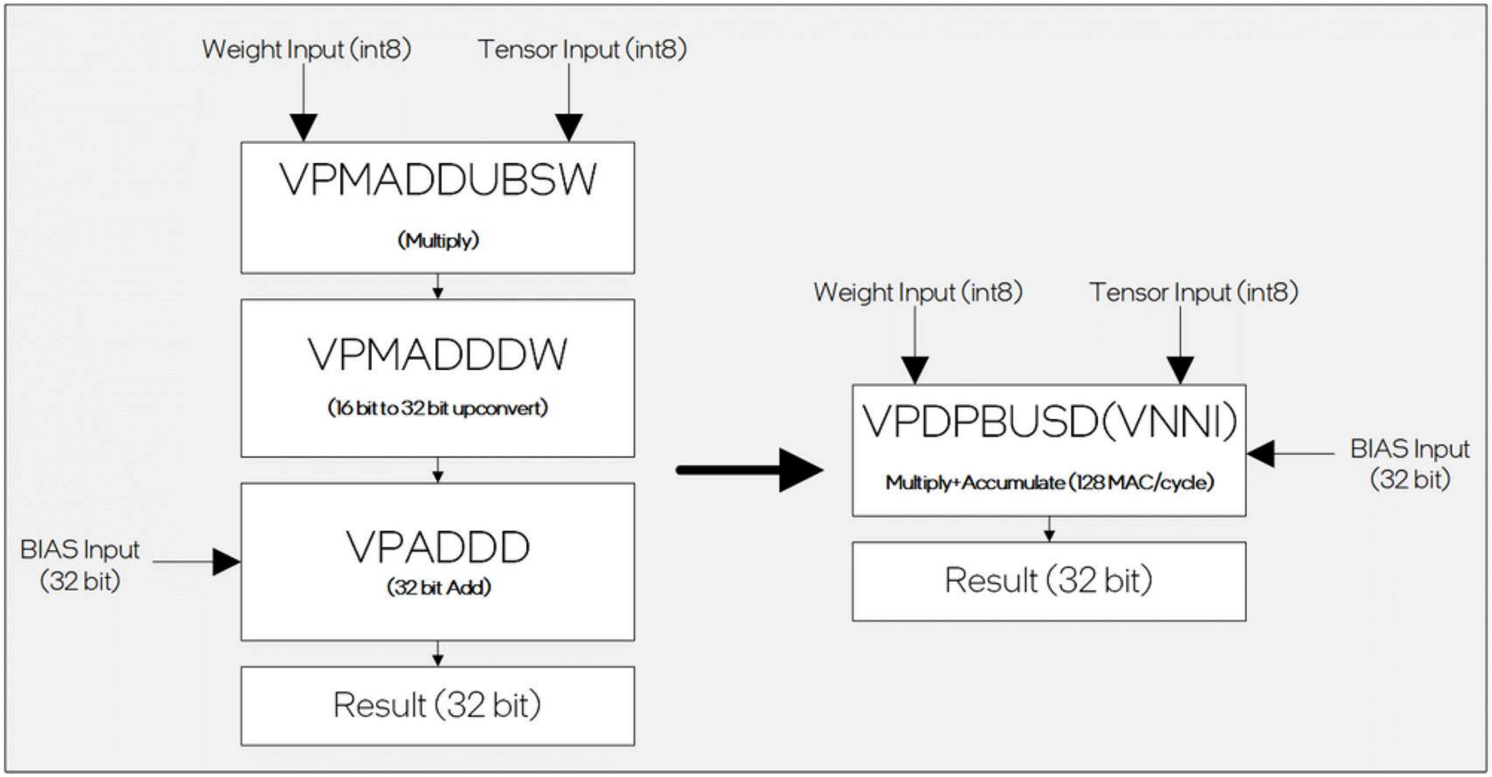
AWS (Source)

ONNX (Open Neural Network Exchange)

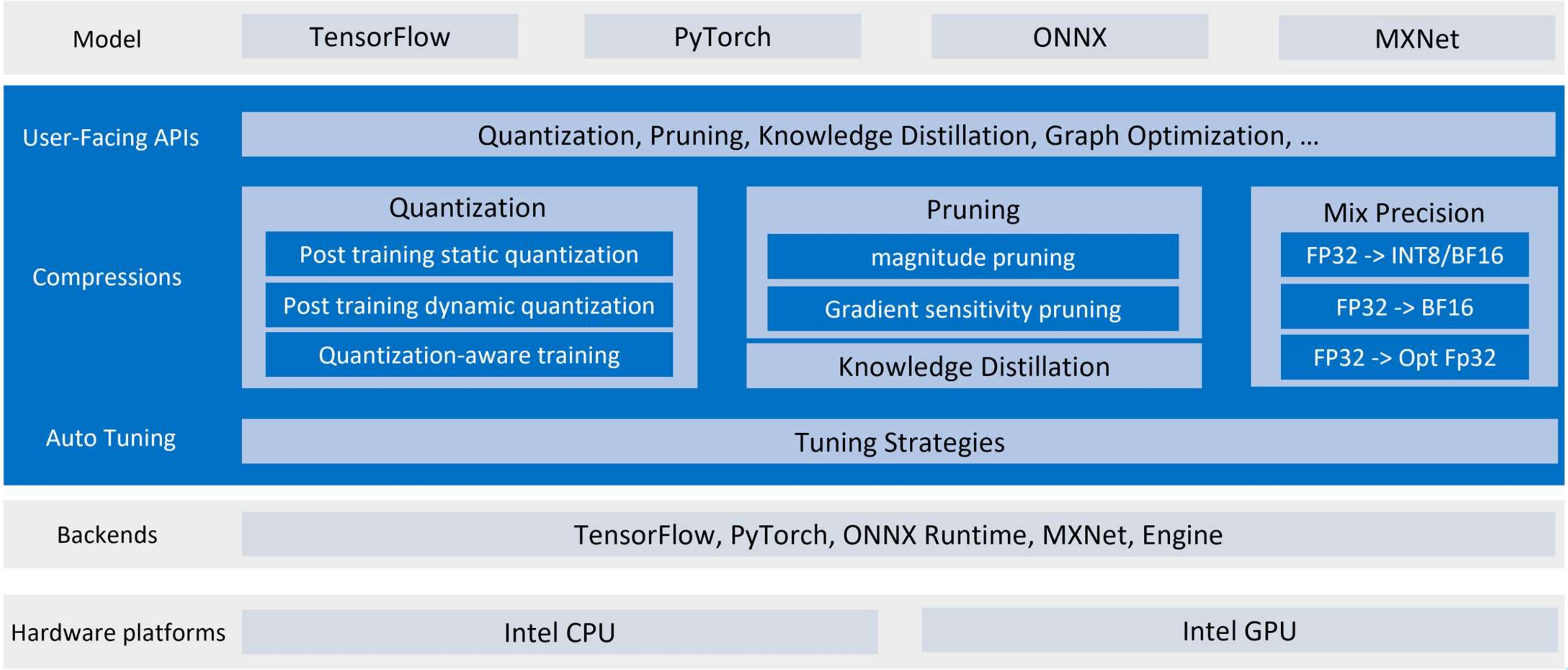
An open format to represent both deep learning and traditional models

- Developed and maintained by community of partners (Microsoft, Facebook & AWS)
- Designed to offer interoperability across different frameworks.
- ONNXRuntime – Runtime library to maximize performance on Intel hardware for ONNX inference.

Intel® Deep Learning Boost - VNNI



Intel® Neural Compressor - Architecture



Model

TensorFlow

PyTorch

ONNX

MXNet

User-Facing APIs

Quantization, Pruning, Knowledge Distillation, Graph Optimization, …

Compressions

Quantization

- Post training static quantization
- Post training dynamic quantization
- Quantization-aware training

Pruning

- magnitude pruning
- Gradient sensitivity pruning

Knowledge Distillation

Mix Precision

- FP32 -> INT8/BF16
- FP32 -> BF16
- FP32 -> Opt Fp32

Auto Tuning

Tuning Strategies

Backends

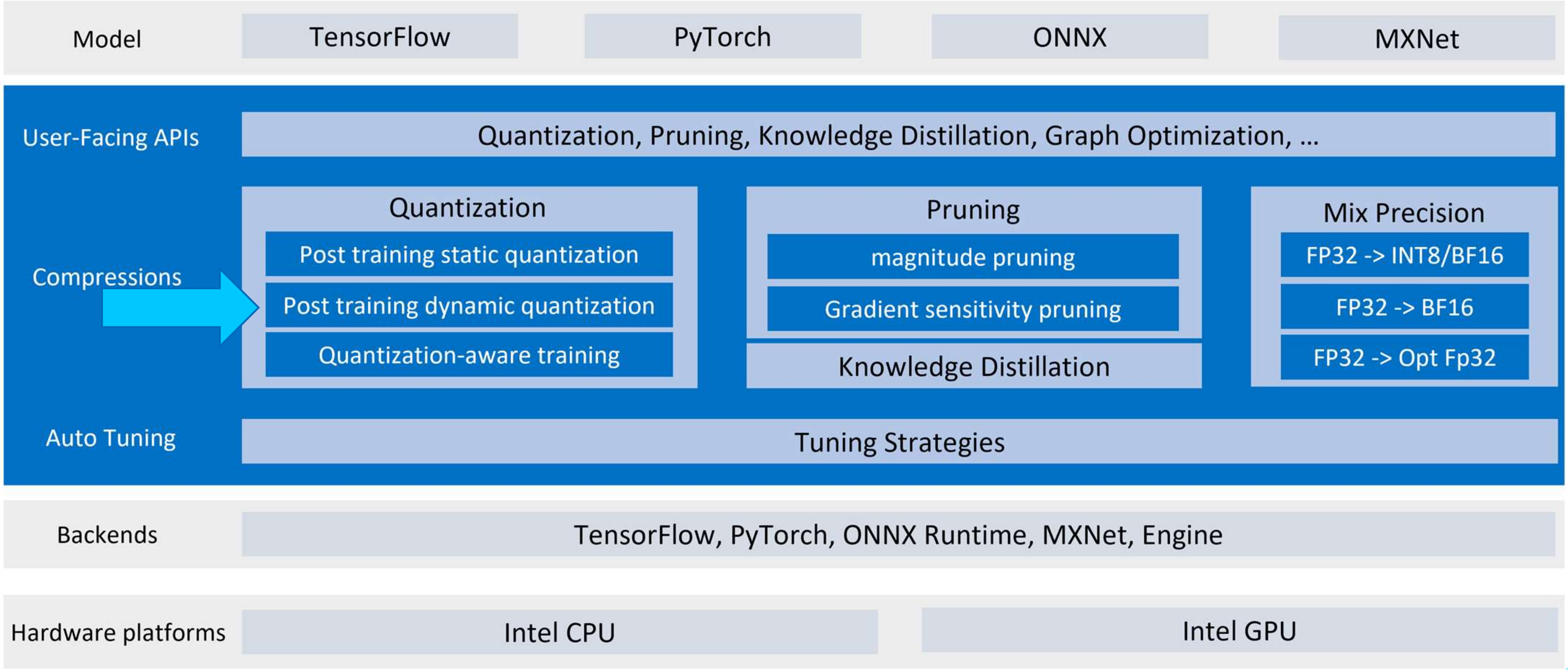
TensorFlow, PyTorch, ONNX Runtime, MXNet, Engine

Hardware platforms

Intel CPU

Intel GPU

Intel® Neural Compressor - Architecture



Model: TensorFlow, PyTorch, ONNX, MXNet

User-Facing APIs: Quantization, Pruning, Knowledge Distillation, Graph Optimization, …

Compressions:

- Quantization
  - Post training static quantization
  - Post training dynamic quantization
  - Quantization-aware training
- Pruning
  - magnitude pruning
  - Gradient sensitivity pruning
- Knowledge Distillation
- Mix Precision
  - FP32 -> INT8/BF16
  - FP32 -> BF16
  - FP32 -> Opt Fp32

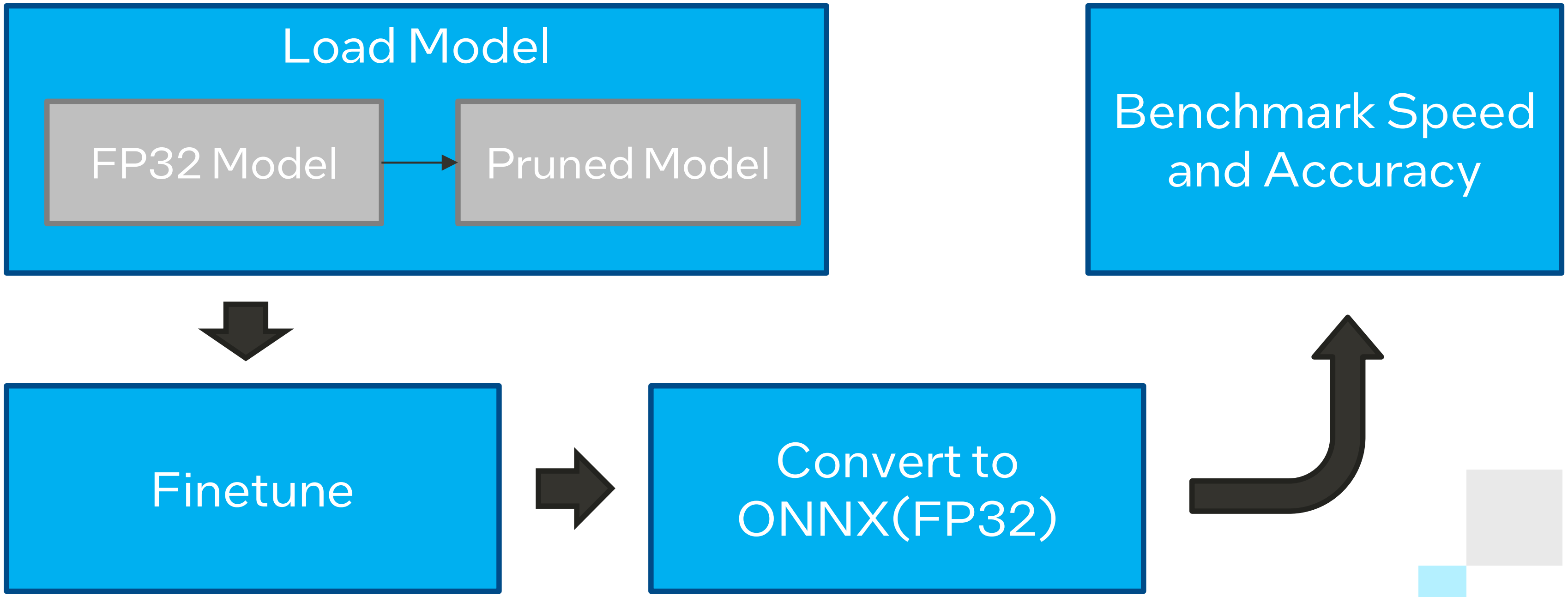
Auto Tuning: Tuning Strategies

Backends: TensorFlow, PyTorch, ONNX Runtime, MXNet, Engine

Hardware platforms: Intel CPU, Intel GPU

Demo Overview

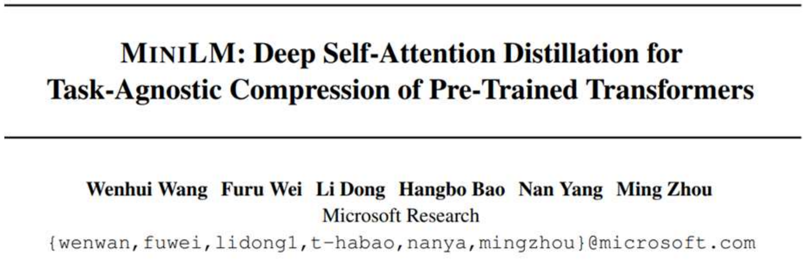
Part 1 – Finetune full & pruned models and run benchmarks.



Demo Overview

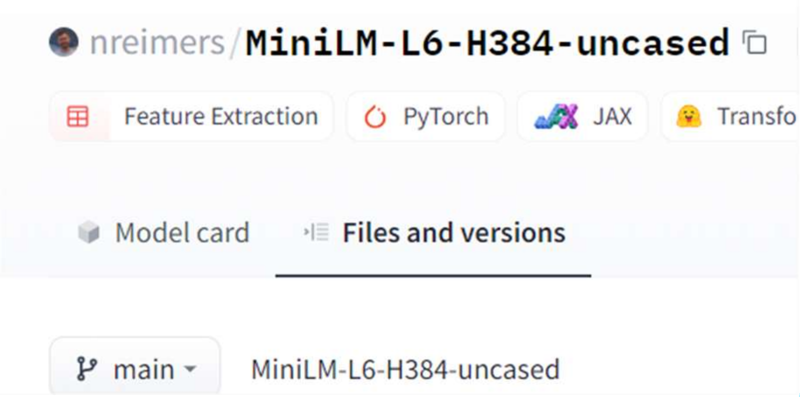
Model details:

- MiniLM – distilled approach created by Microsoft Research.
- The 12 Layer version from the paper shared in microsoft/MiniLM-L12-H384-uncased.
- This is a 6-layer version of that model, by keeping only every second layer.



MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers

Wenhui Wang Furu Wei Li Dong Hangbo Bao Nan Yang Ming Zhou
Microsoft Research
{wenwan,fuwei,lidong1,t-habao,nanya,mingzhou}@microsoft.com

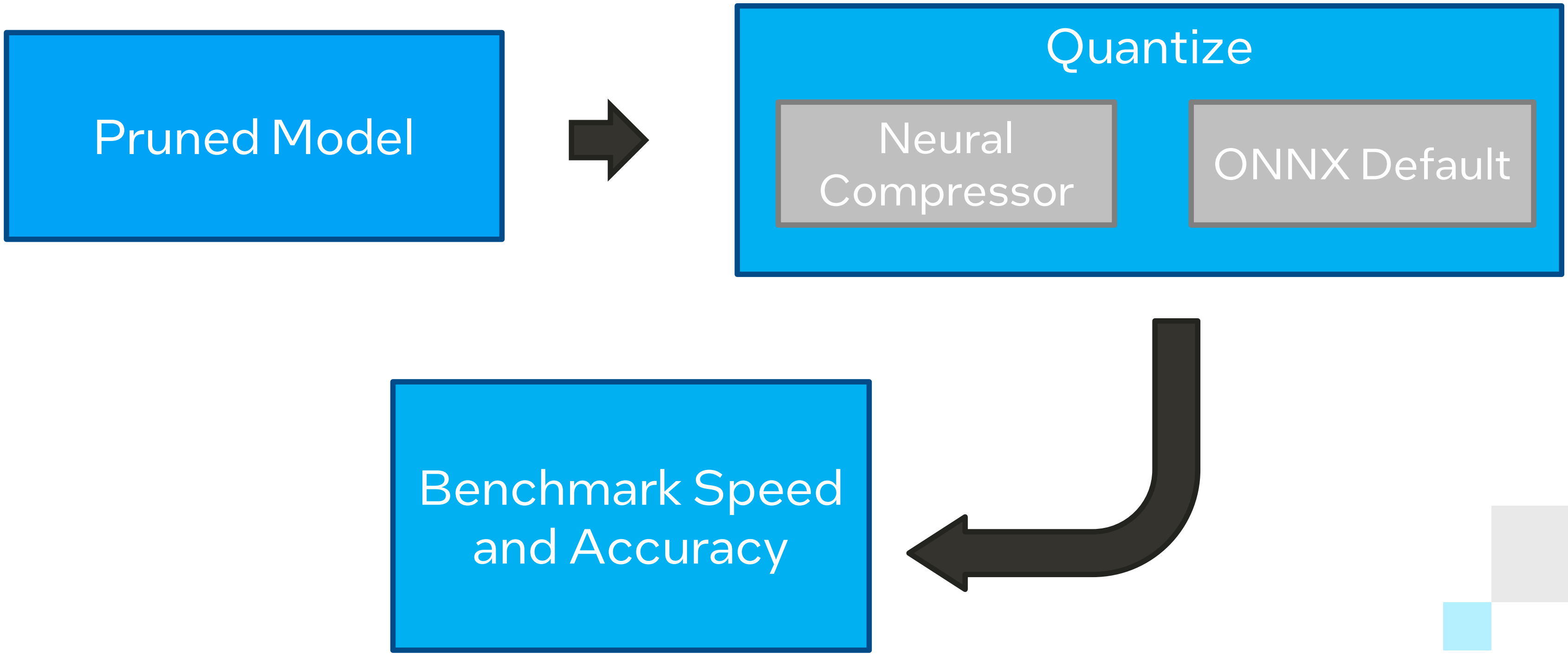


nreimers/MiniLM-L6-H384-uncased

MiniLM-L6-H384-uncased

Demo Overview

Part 2 – Covert pruned models to INT8 and run benchmarks.



Pruned Model

Quantize

Neural Compressor

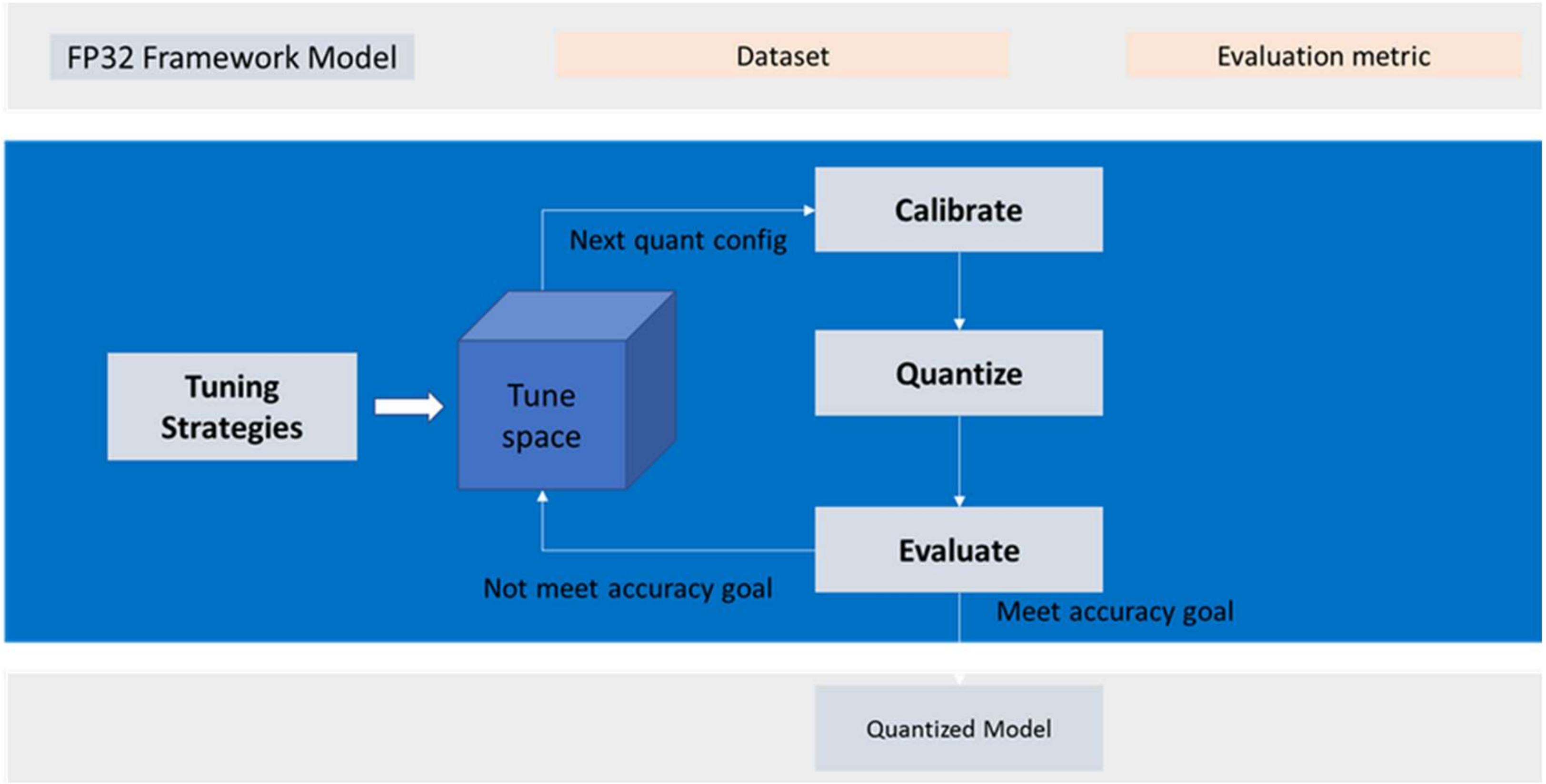
ONNX Default

Benchmark Speed and Accuracy

Intel® Neural Compressor – Quantization Workflow

Input

Output



FP32 Framework Model

Dataset

Evaluation metric

Tuning Strategies

Tune space

Next quant config

Calibrate

Quantize

Evaluate

Not meet accuracy goal

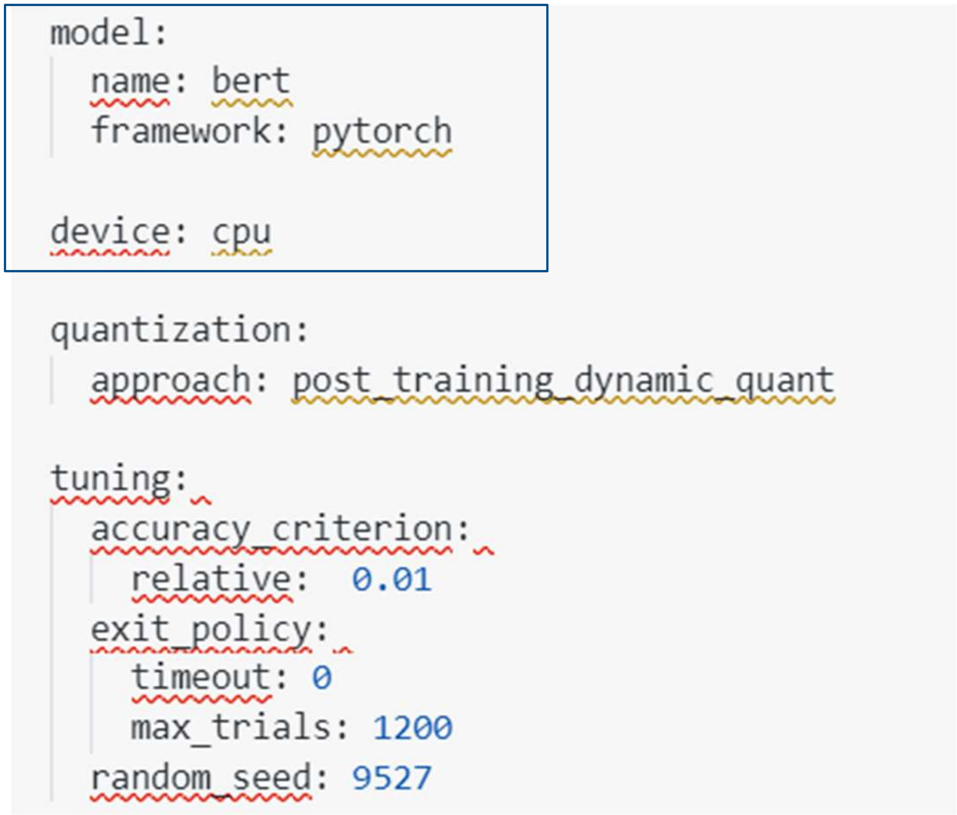
Meet accuracy goal

Quantized Model

Intel® Neural Compressor - Configurations

Model Configurations

- “model” and “device” sections are mandatory
- “name” and “framework” fields are mandatory
- Possible values for framework are “tensorflow”, “mxnet”, “pytorch”, “pytorch_ipex”, “onnxrt_integerops” and “onnxrt_qlinearops”

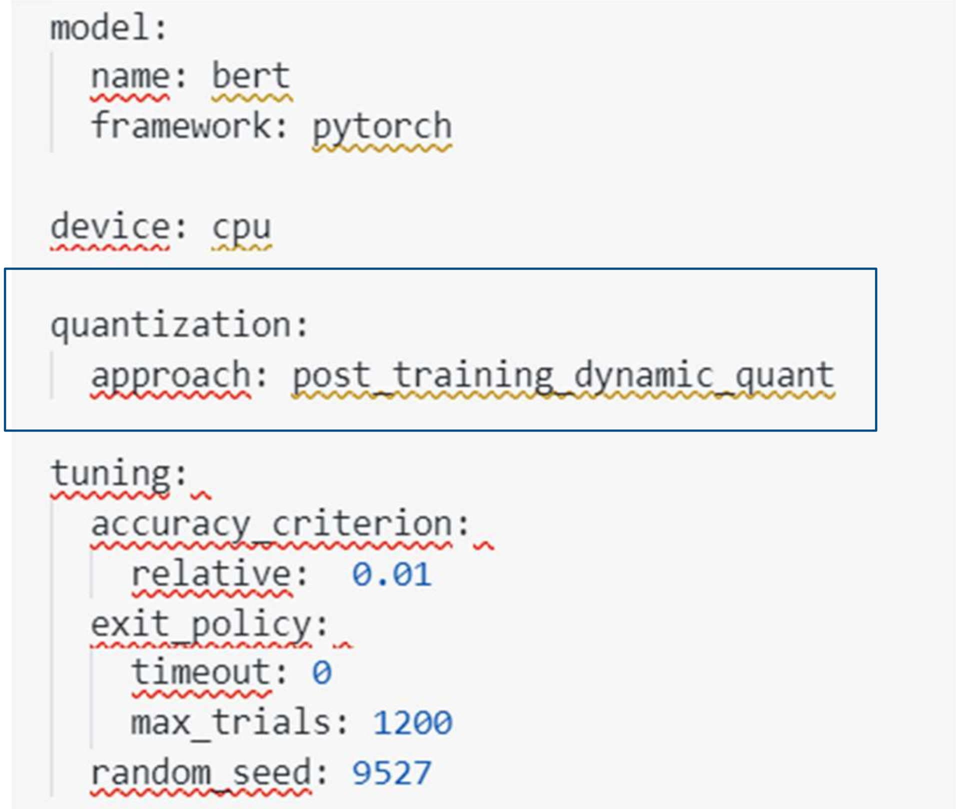


```yaml
model:
  name: bert
  framework: pytorch

device: cpu

quantization:
  approach: post_training_dynamic_quant

tuning:
  accuracy_criterion:
    relative:  0.01
  exit_policy:
    timeout: 0
    max_trials: 1200
  random_seed: 9527
```

Intel® Neural Compressor - Configurations

Quantization Configurations

- "approach" – to specify quantization method to be used post training (static/dynamic) quantization, or quantization aware training.

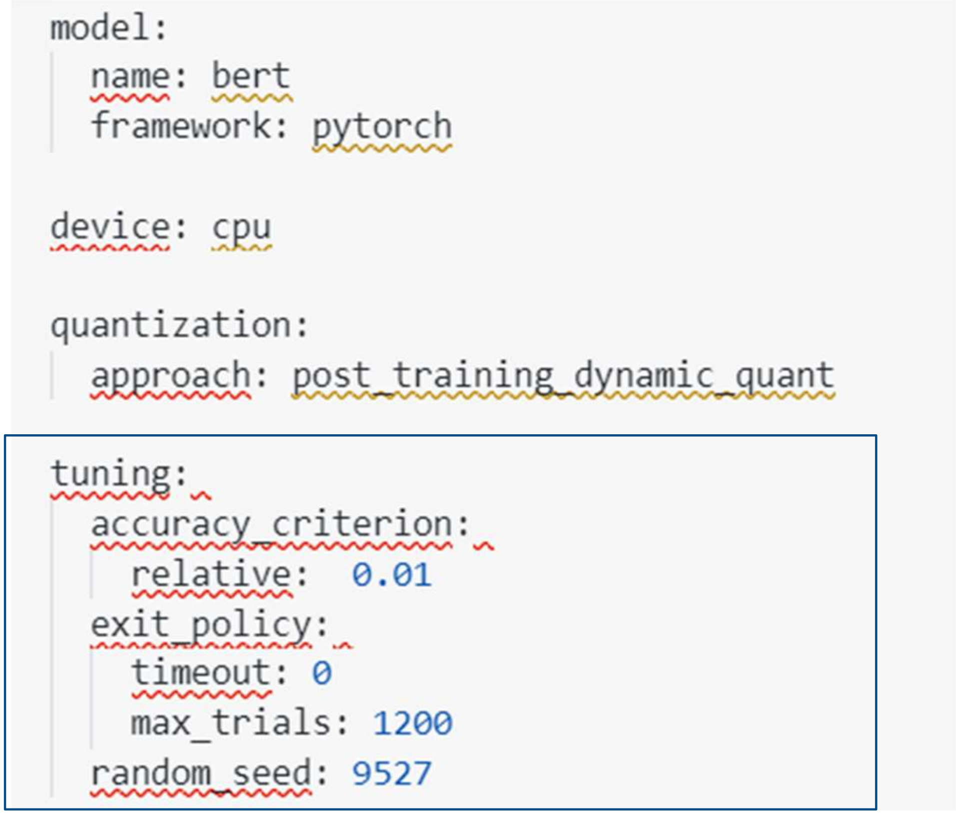


```yaml
model:
  name: bert
  framework: pytorch

device: cpu

quantization:
  approach: post_training_dynamic_quant

tuning:
  accuracy_criterion:
    relative:  0.01
  exit_policy:
    timeout: 0
    max_trials: 1200
  random_seed: 9527
```

Intel® Neural Compressor - Configurations

Tuning Configurations

- “accuracy_criterion” could have “relative” or “obsolete” as fields.
- Example shown here allows relative accuracy loss of 1%
- “exit_policy” decides when to exit tuning. Here “timeout” (specified in seconds) is set at 0 which means early stop.
- “max_trials” indicates the maximum number of iterations to be tried.
- “random_seed” for deterministic tuning.



```
model:
  name: bert
  framework: pytorch

device: cpu

quantization:
  approach: post_training_dynamic_quant

tuning:
  accuracy_criterion:
    relative:  0.01
  exit_policy:
    timeout: 0
    max_trials: 1200
  random_seed: 9527
```

Further reading

- https://github.com/intel/neural-compressor/blob/master/docs/dynamic_quantization.md
- https://community.intel.com/t5/Blogs/Tech-Innovation/Artificial-Intelligence-AI/Quantizing-ONNX-Models-using-Intel-Neural-Compressor/post/1355237
- https://pytorch.org/docs/stable/quantization.html
- https://github.com/intel/neural-compressor/blob/master/docs/tuning_strategies.md

Thank you