

ML inference on Mobile with ONNX Runtime

Emma Ning - Senior Product Manager, Microsoft

Yufeng Li - Senior Software Engineer, Microsoft

Guoyu Wang - Senior Software Engineer, Microsoft

Agenda

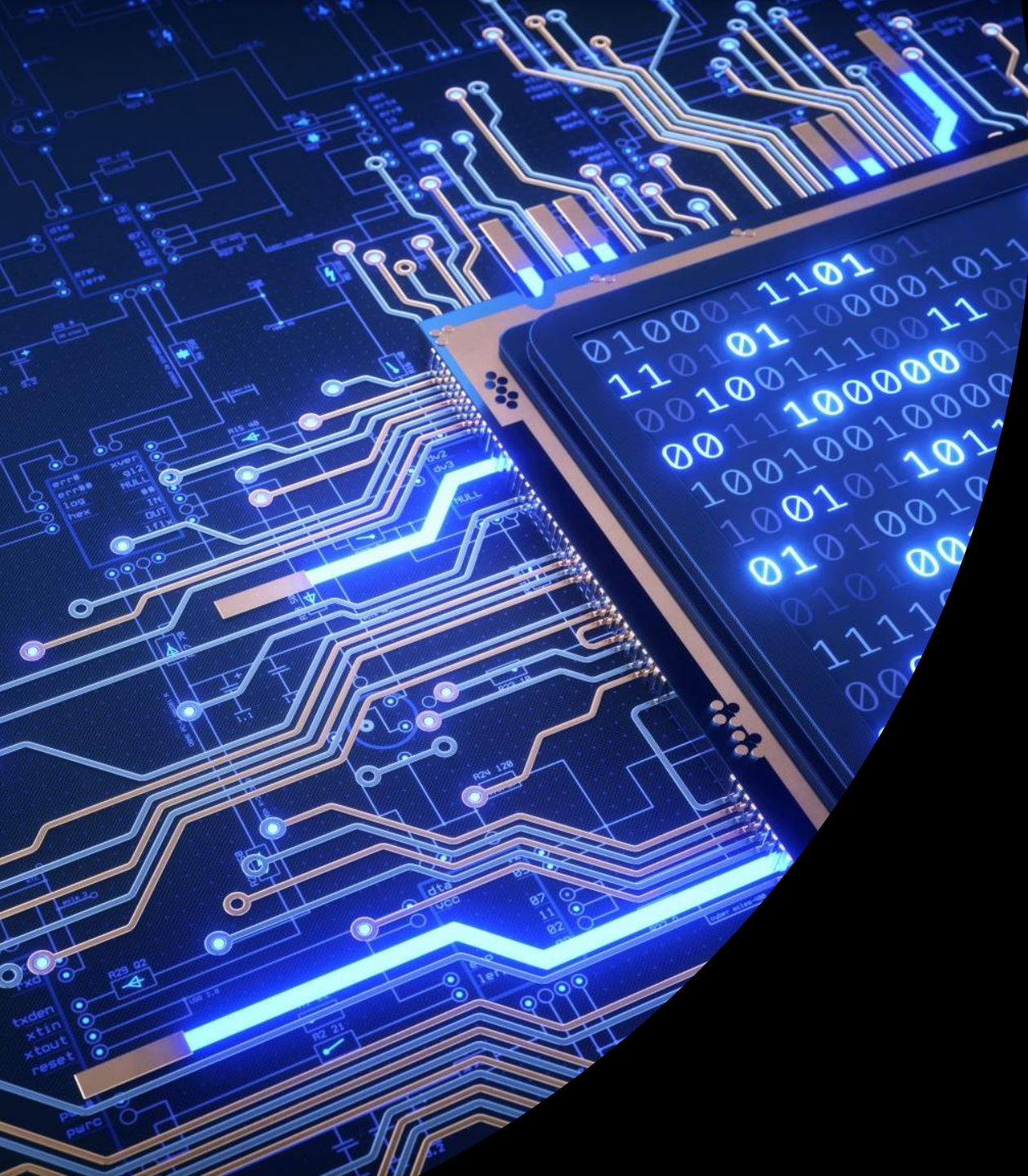


ONNX & ONNX Runtime Overview

ONNX Runtime Quantization

ONNX Runtime Mobile

Demo



ONNX & ONNX Runtime Overview

Mobile inference in Microsoft Office

Applications

X



Model Complexity

X

Transformers RNN CNN LR LightGBM ...

Platforms Complexity

X



Inferencing Complexity

TensorFlow Pytorch CoreML Keras ...

Challenges

Consistency across
all endpoints

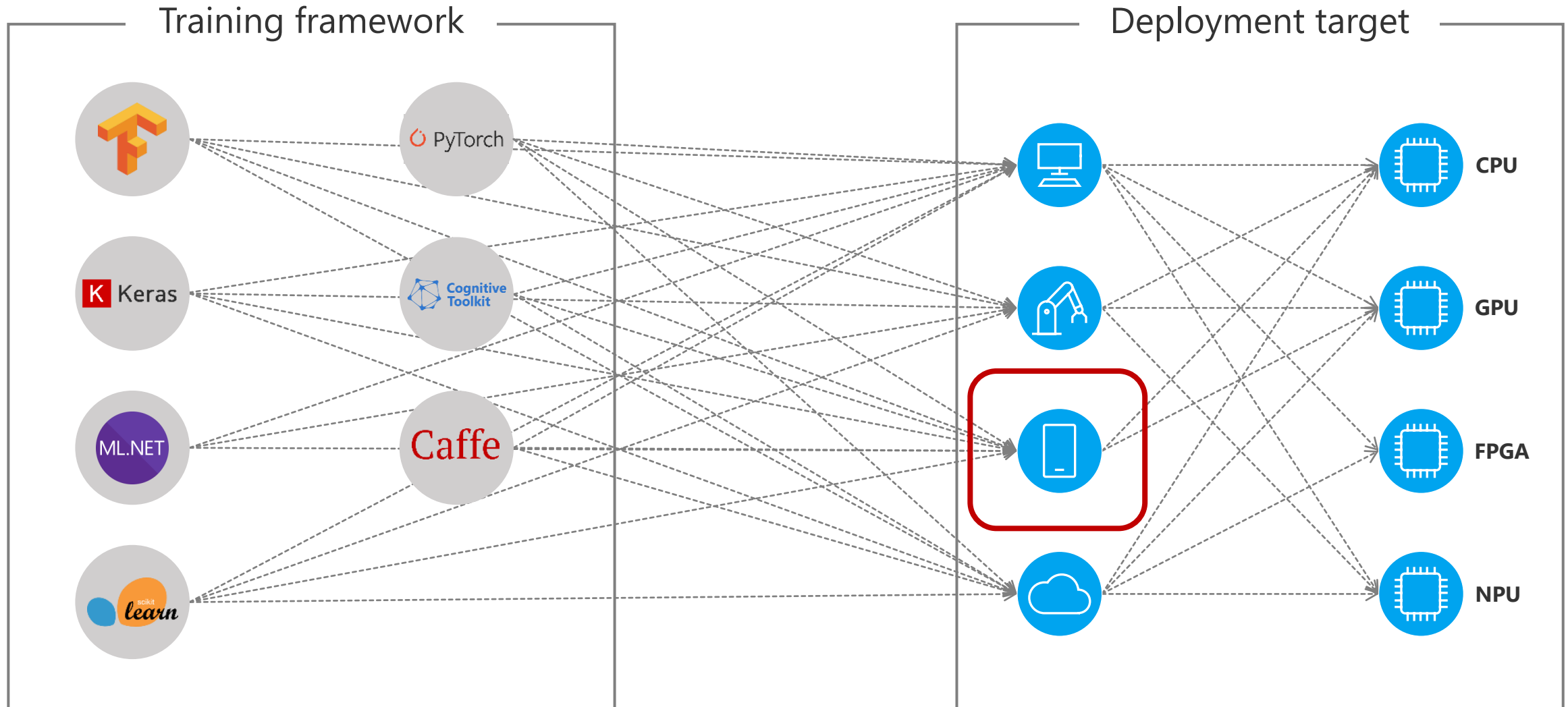
Build & validation
pipeline

Platform
fragmentation

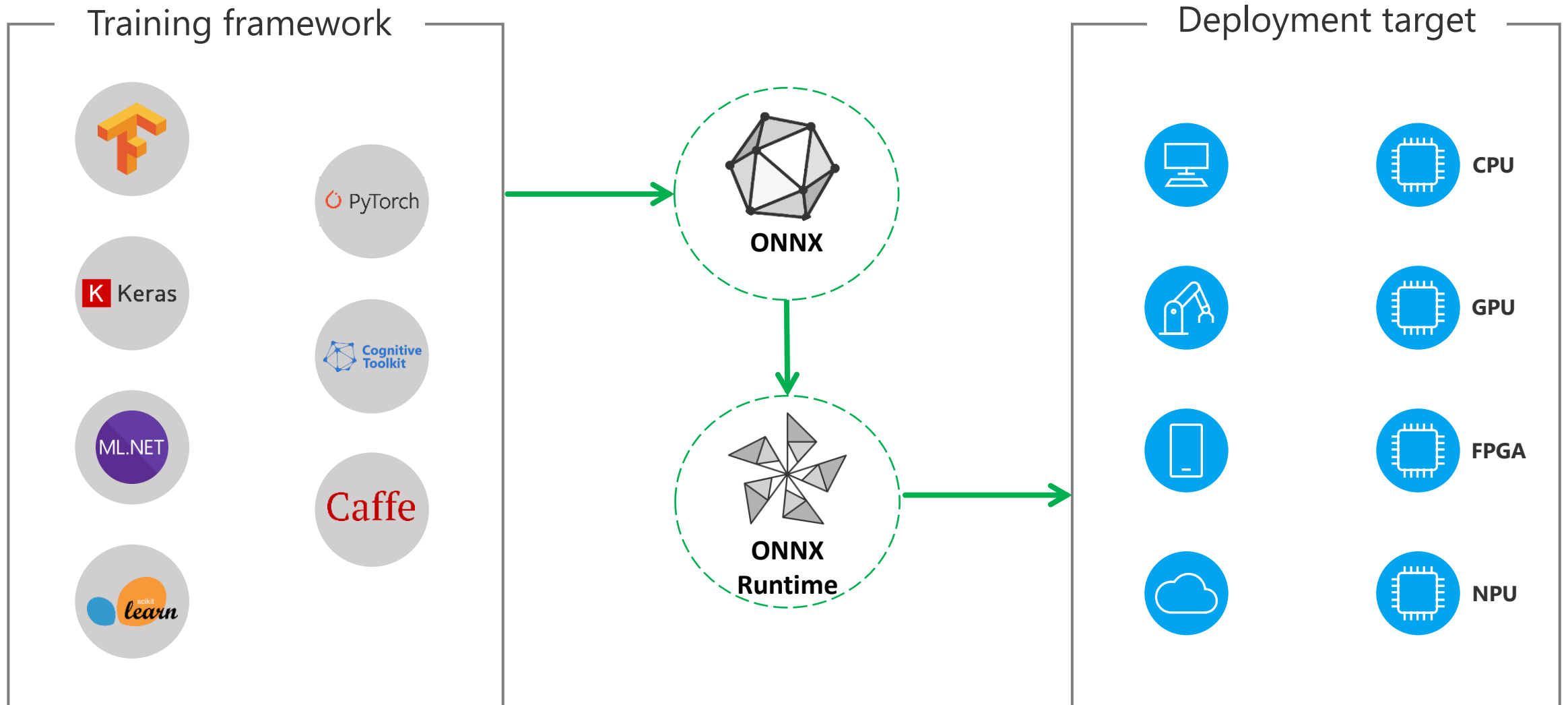
Space/Storage/Size

Need a unified inference solution
providing consistency development experience, high performance and lightweight inference

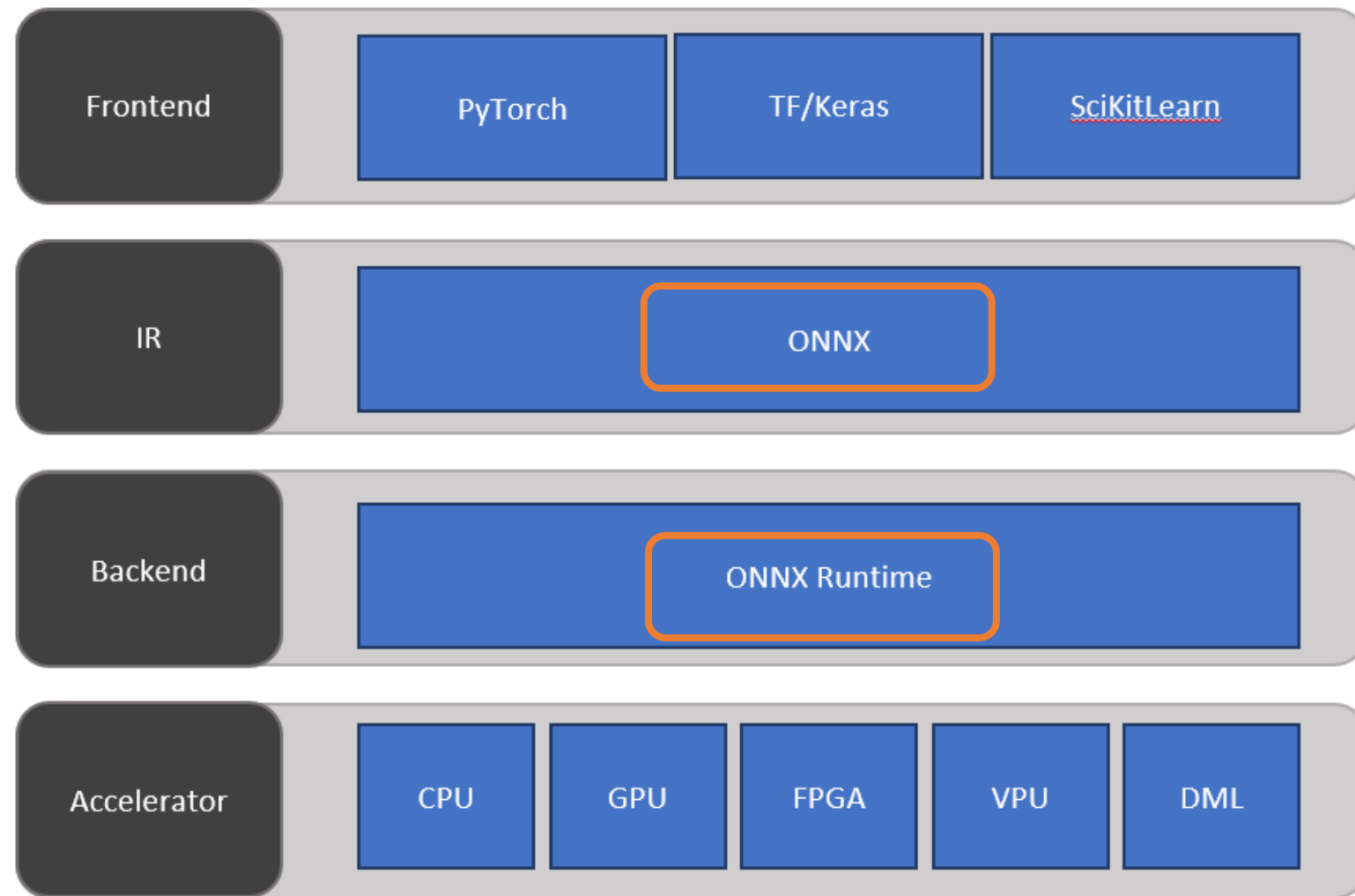
ONNX/ONNX Runtime - Consistency development experience



ONNX/ONNX Runtime - Consistency development experience



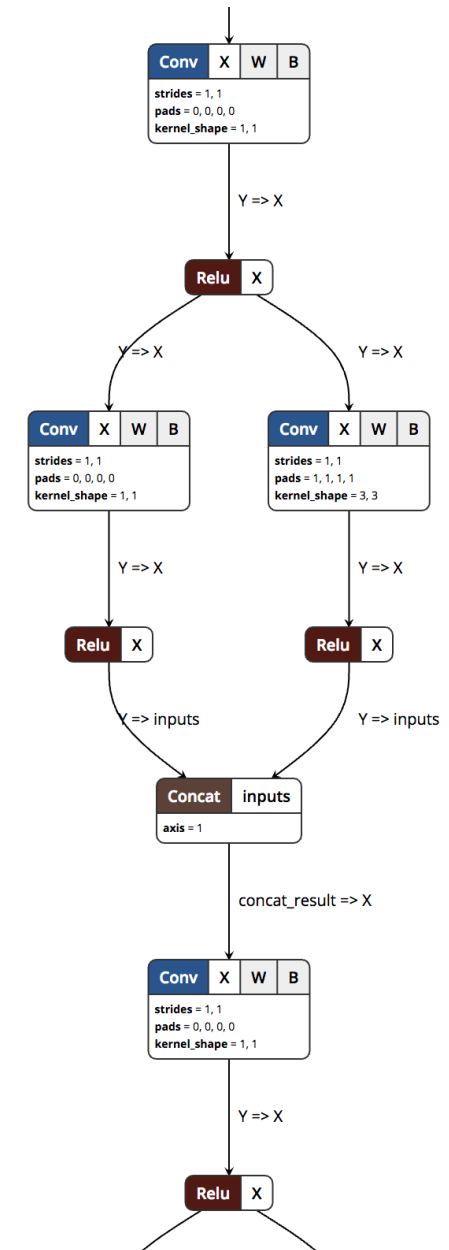
AI Software Stack for Inference with ONNX

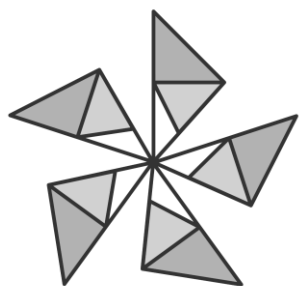


ONNX

- Open Neural Network Exchange

- A specification that defines a standard format for ML models
- Consisting of:
 - common Intermediate Representation
 - full operator spec
- Supports both DNN and traditional ML
- Backward compatible with comprehensive versioning





ONNX RUNTIME

high-performance engine for machine learning models

Flexible

Supports full ONNX-ML spec
(v1.2-1.7)

Supports CPU, GPU, VPU

C#, C, C++, Java, JS and
Python APIs

Cross Platform

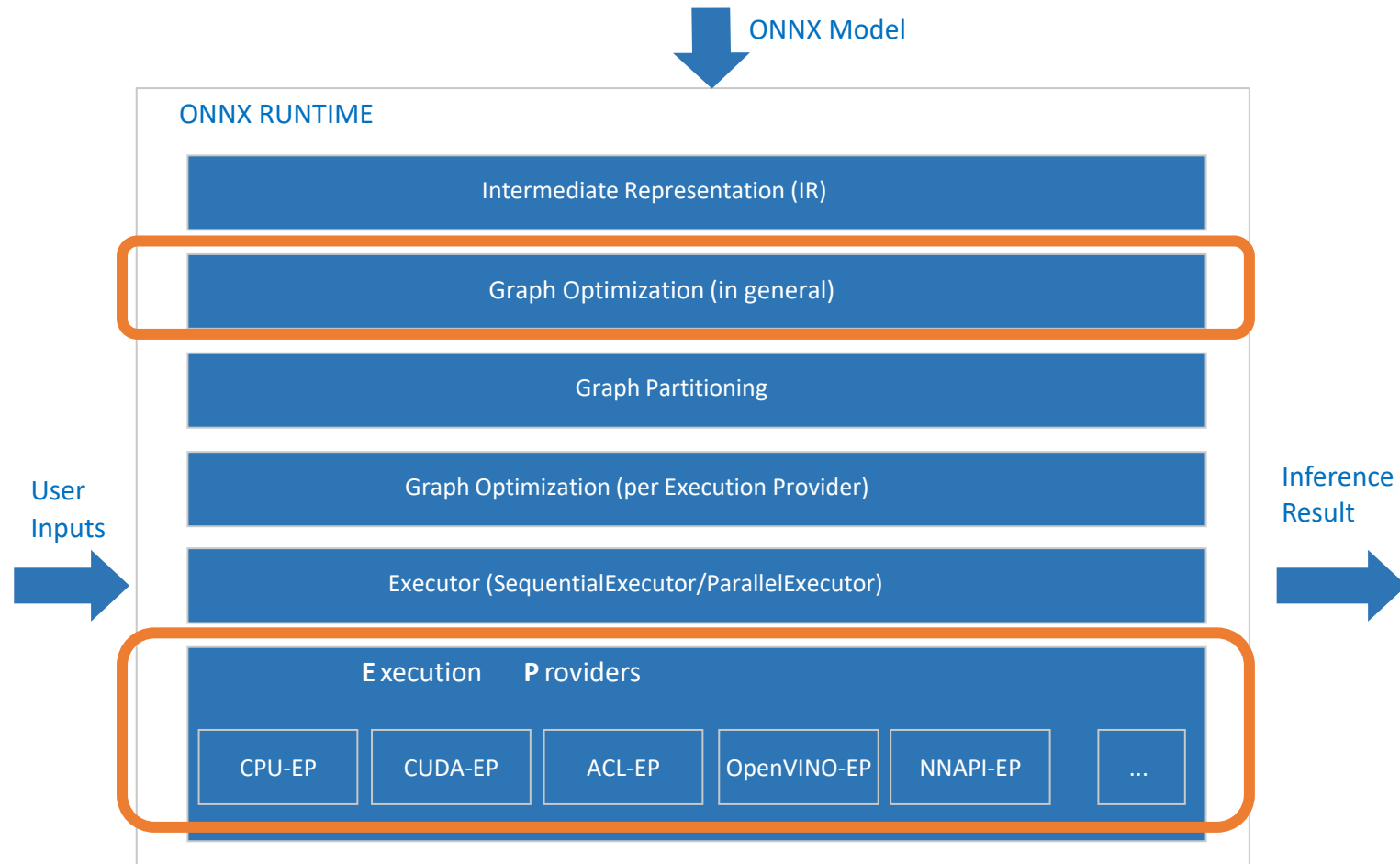
Works on
-Mac, Windows, Linux,
Android, IOS
-x86, x64, ARM

Also built-in to Windows 10
natively (WinML)

Extensible

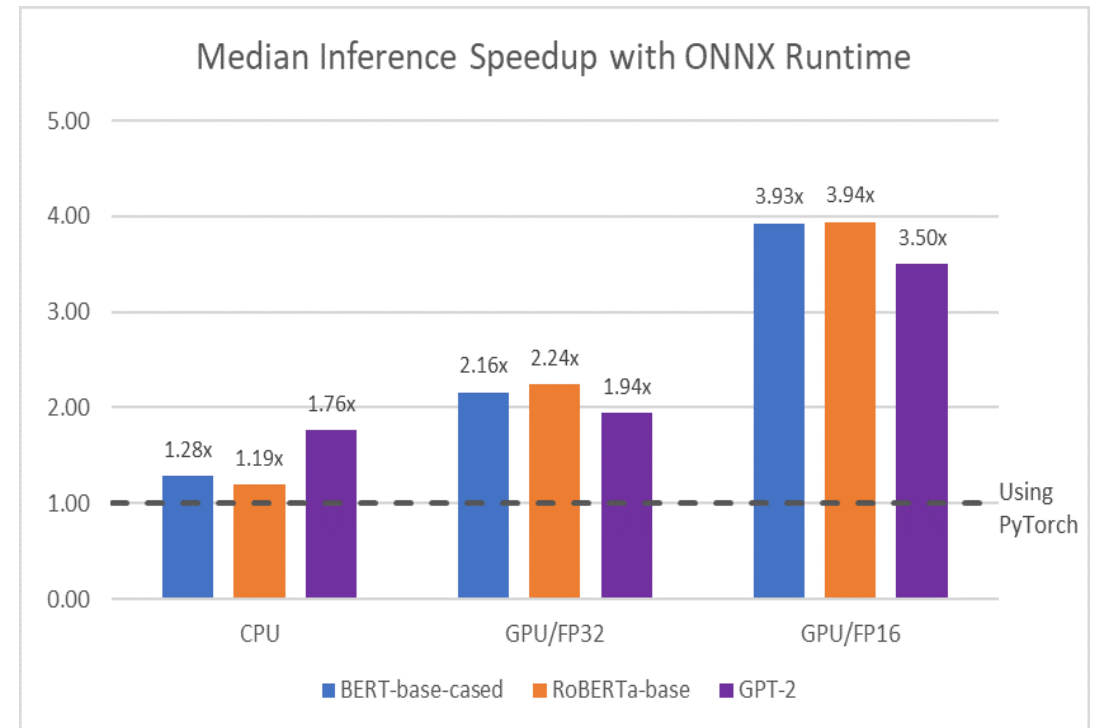
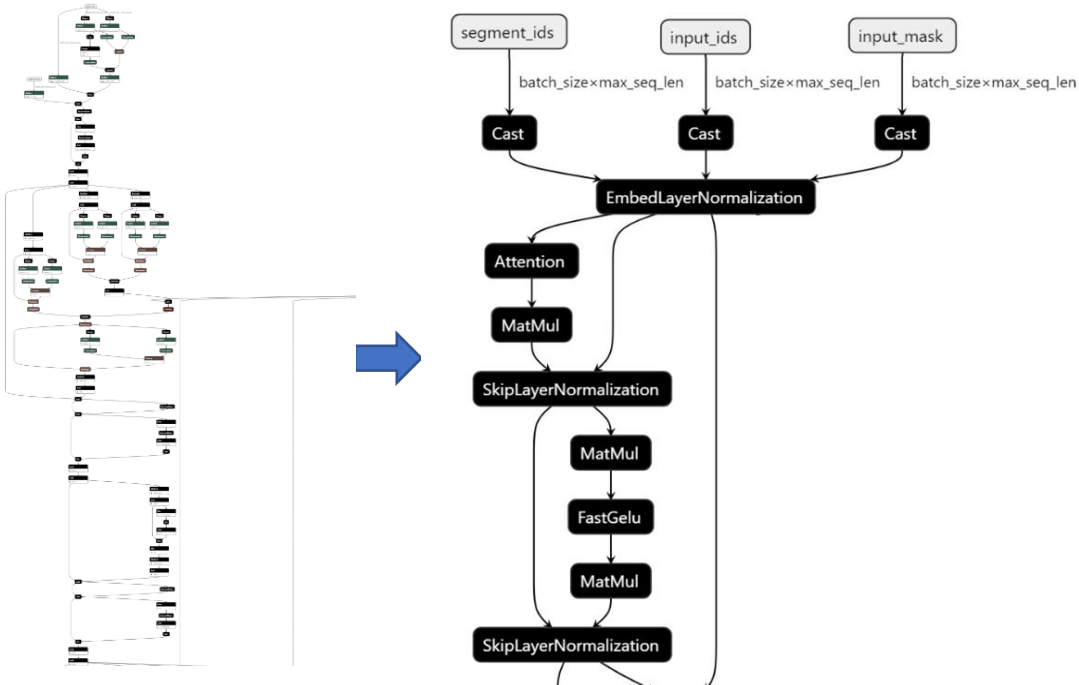
Extensible architecture to
plug-in custom operators,
optimizers, and hardware
accelerators

ONNX Runtime – High performance



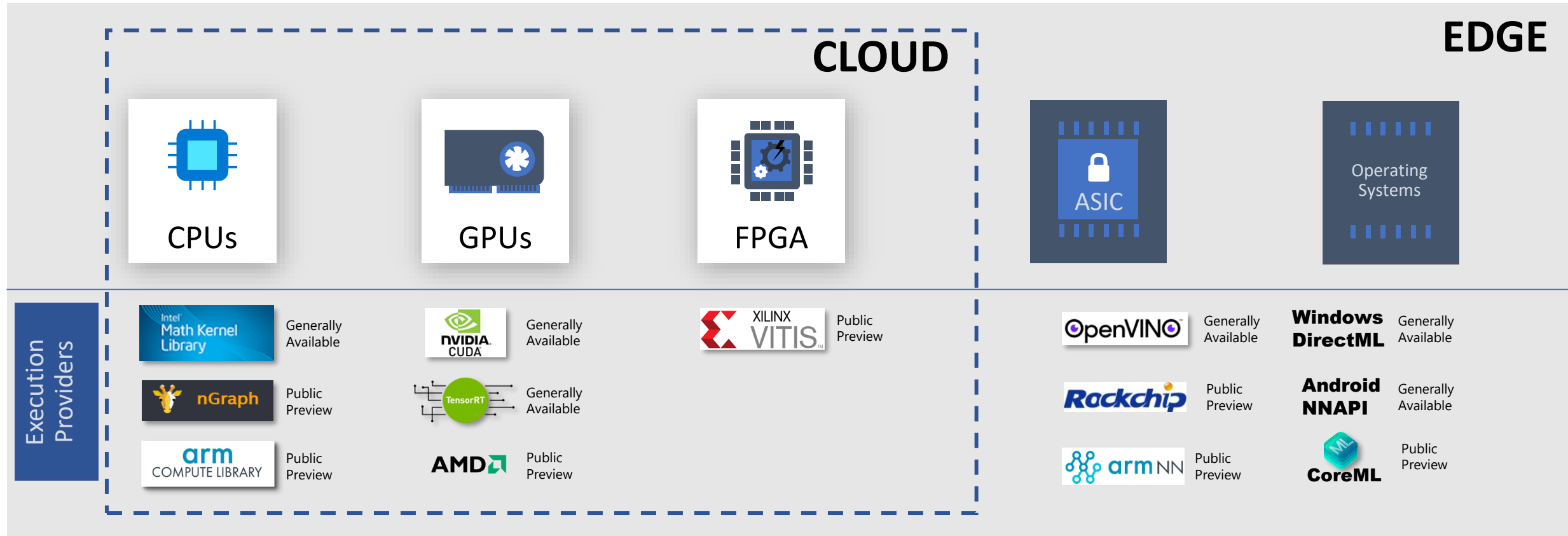
ONNX Runtime – High performance

- Transformer acceleration with ONNX Runtime
 - Graph optimization in ONNX Runtime
 - Kernel optimizations for both CPU and GPU

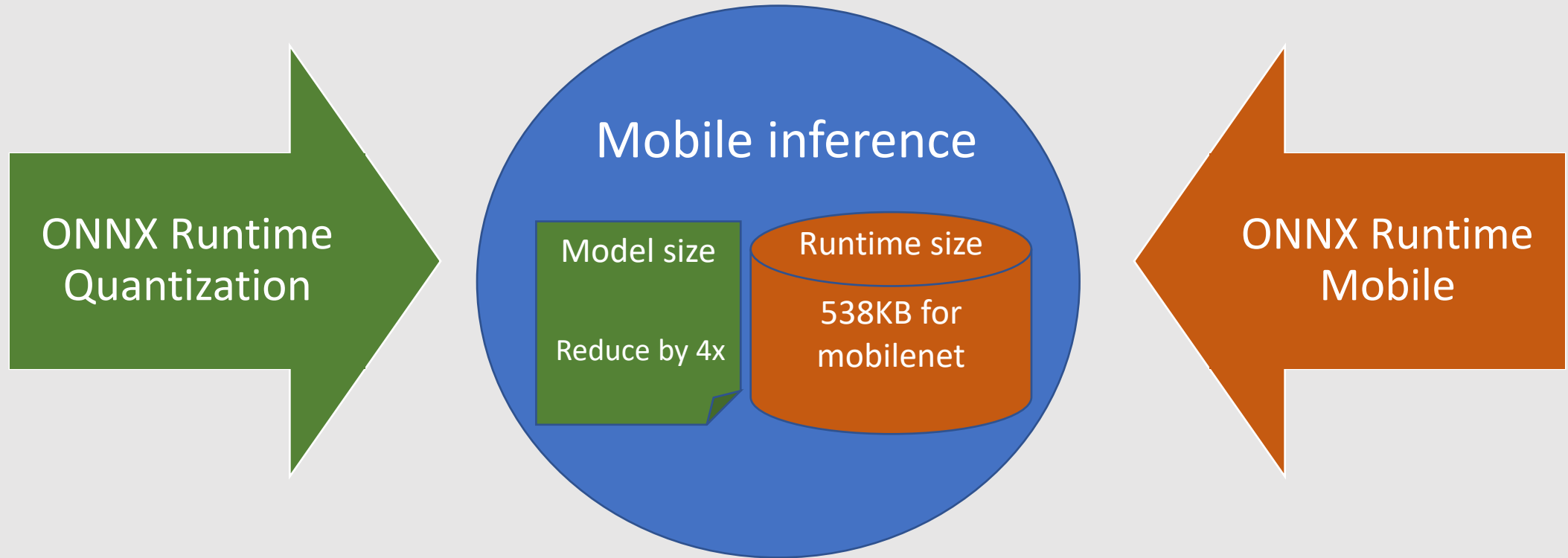


ONNX Runtime – High performance

- Hardware accelerators for different target hardware



ONNX Runtime – Lightweight inference



ONNX Runtime Ecosystem

Up to 17x performance gains seen by Microsoft services

Millions of devices running ONNX Runtime

20 Billion+ requests handled in prod

Extensive hardware support:



Microsoft Advertising



Azure ML



Power BI



Azure Kinect DK



Azure Media Services



Visual Studio Code



SQL Server



Azure Stack Edge



PowerApps

Model inferencing with ONNX Runtime

Train models with various
frameworks

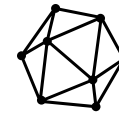


Convert into ONNX with
ONNX Converters



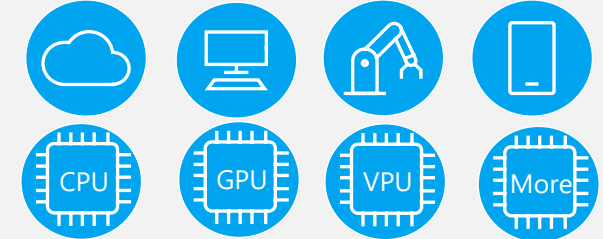
HW accelerated inference with
ONNX Runtime

Frameworks



ONNX Model

Devices



```
import torch
import torch.onnx

model = torch.load("model.pt")

sample_input = torch.randn(1, 3, 224, 224)


torch.onnx.export(model, sample_input,
                  "model.onnx")
```

PyTorch

```
import onnxruntime

session =
onnxruntime.InferenceSession("model.onn
x")

results =
session.run([], {"input": input_data})
```



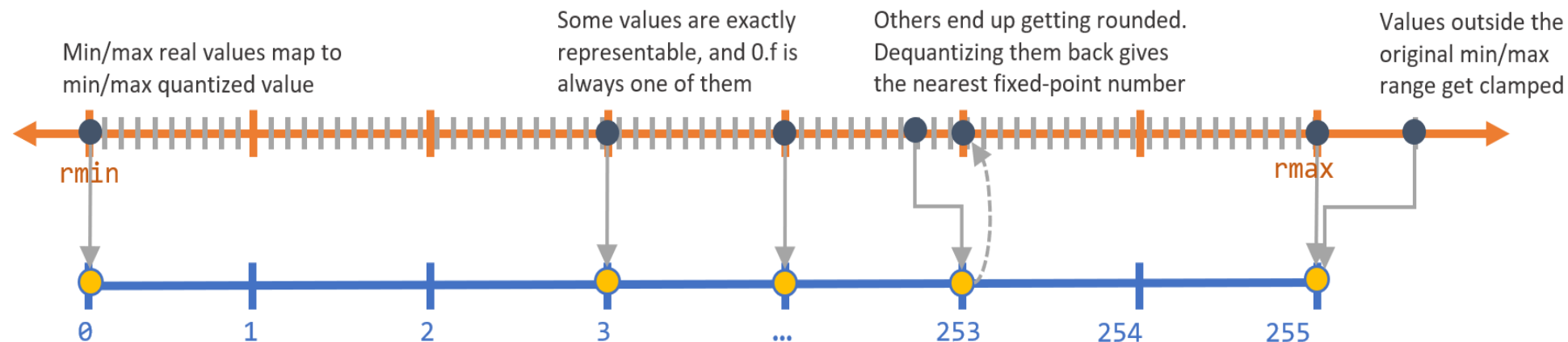


ONNX Runtime Quantization
Faster, smaller model inference

What is Quantization

Approximate floating-point numbers with lower bits, e.g., int8

$$\text{VAL_fp32} = (\text{VAL_quantized} - \text{Zero_point}) * \text{Scale}$$

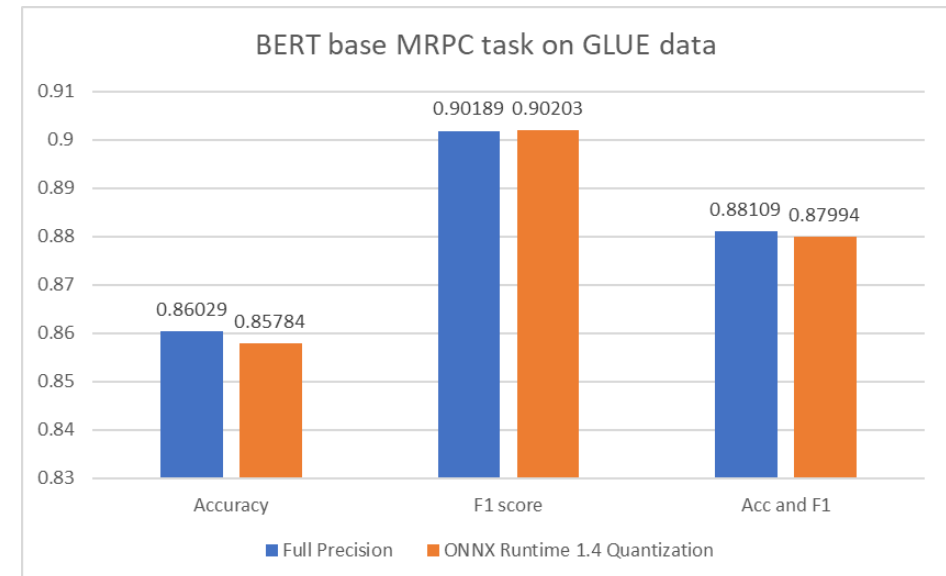
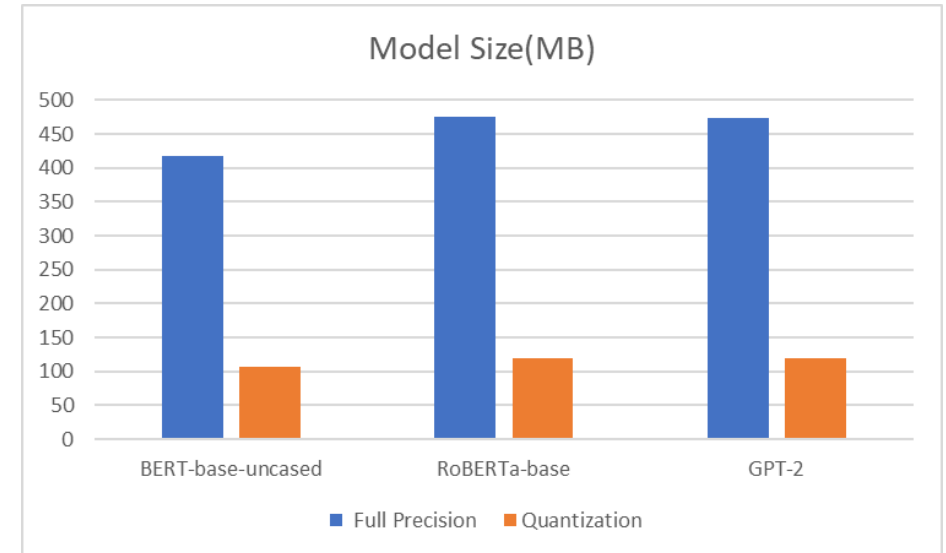


Why is Quantization

- Reduce Model Size
- Reduce memory footprint
- Inference Efficiency

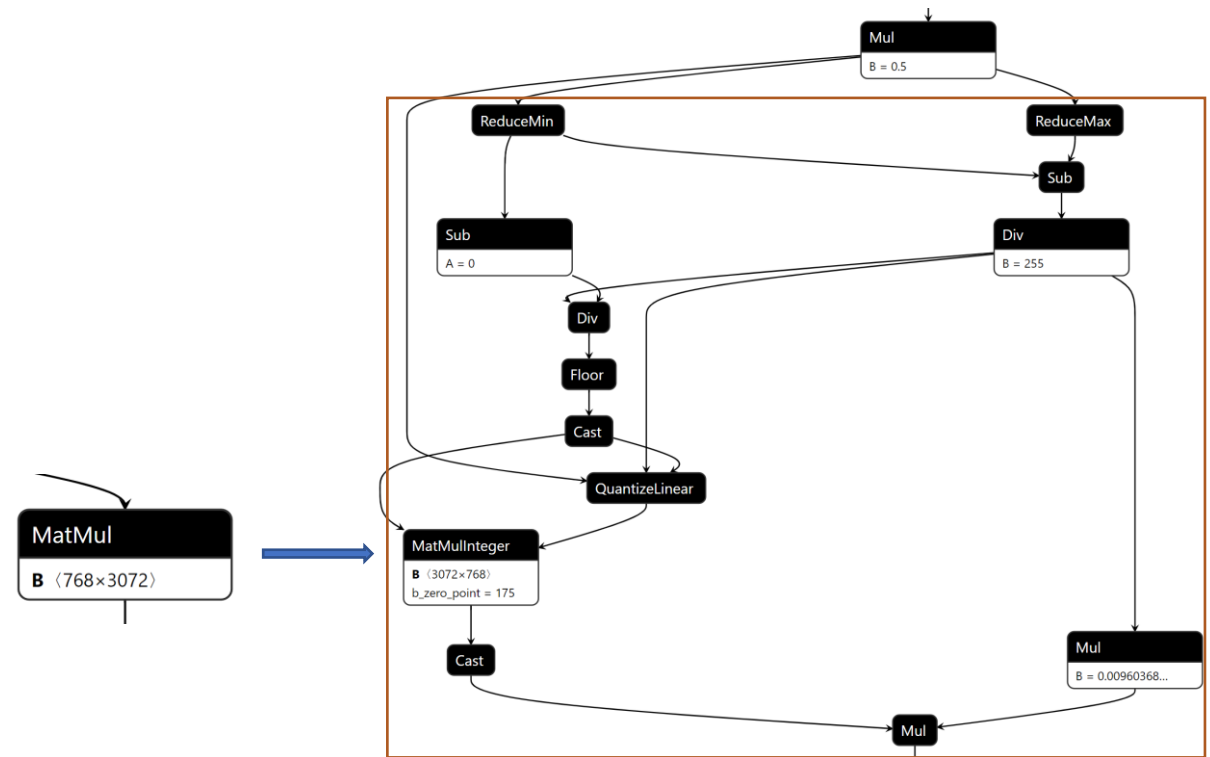


! Memory and compute resource constraint on devices



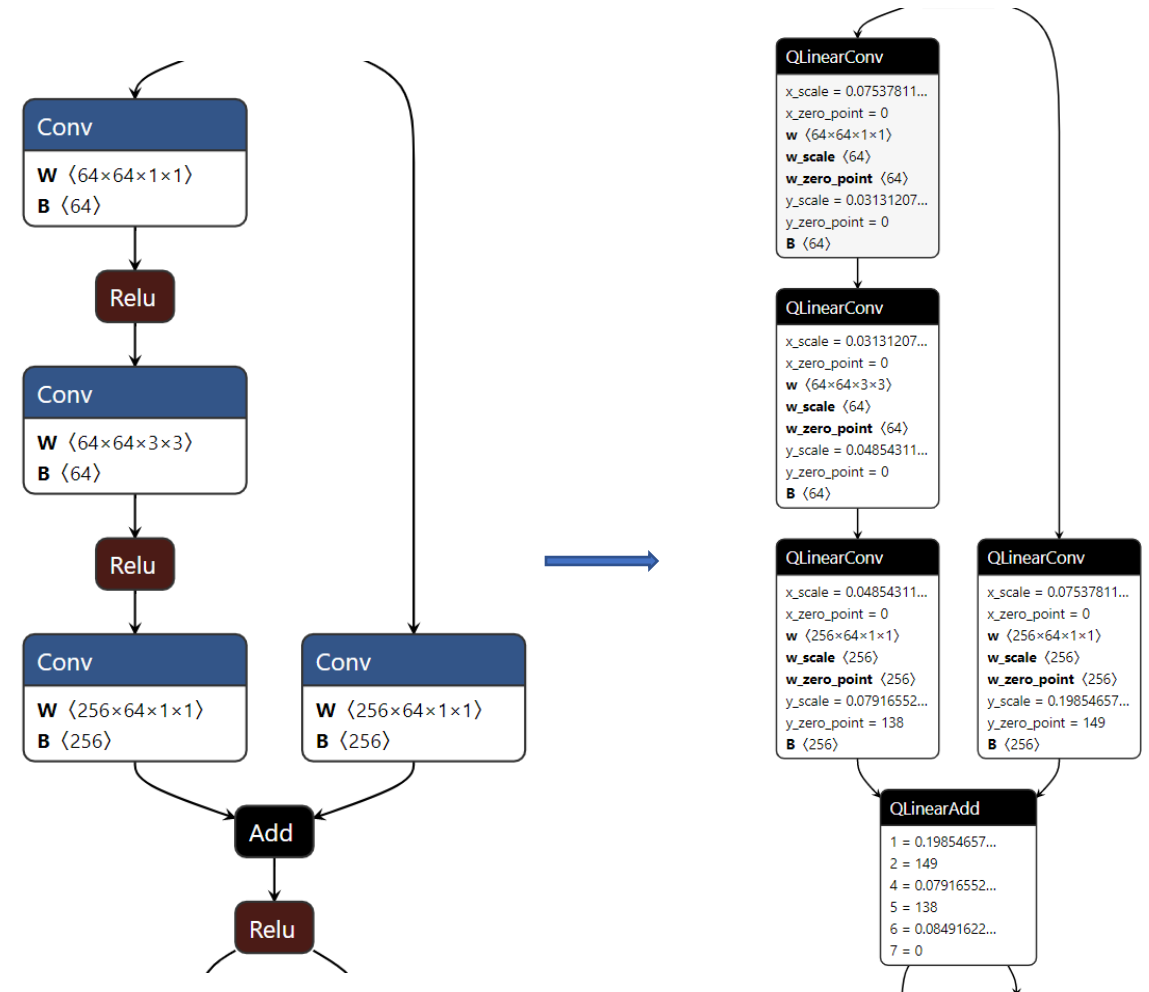
Dynamic Quantization

- Weights are quantized ahead of time
- Activations are quantized during inference dynamically
- Suitable for weights dominate models, like transformer-based model
- Not used on GPU and accelerators in practice



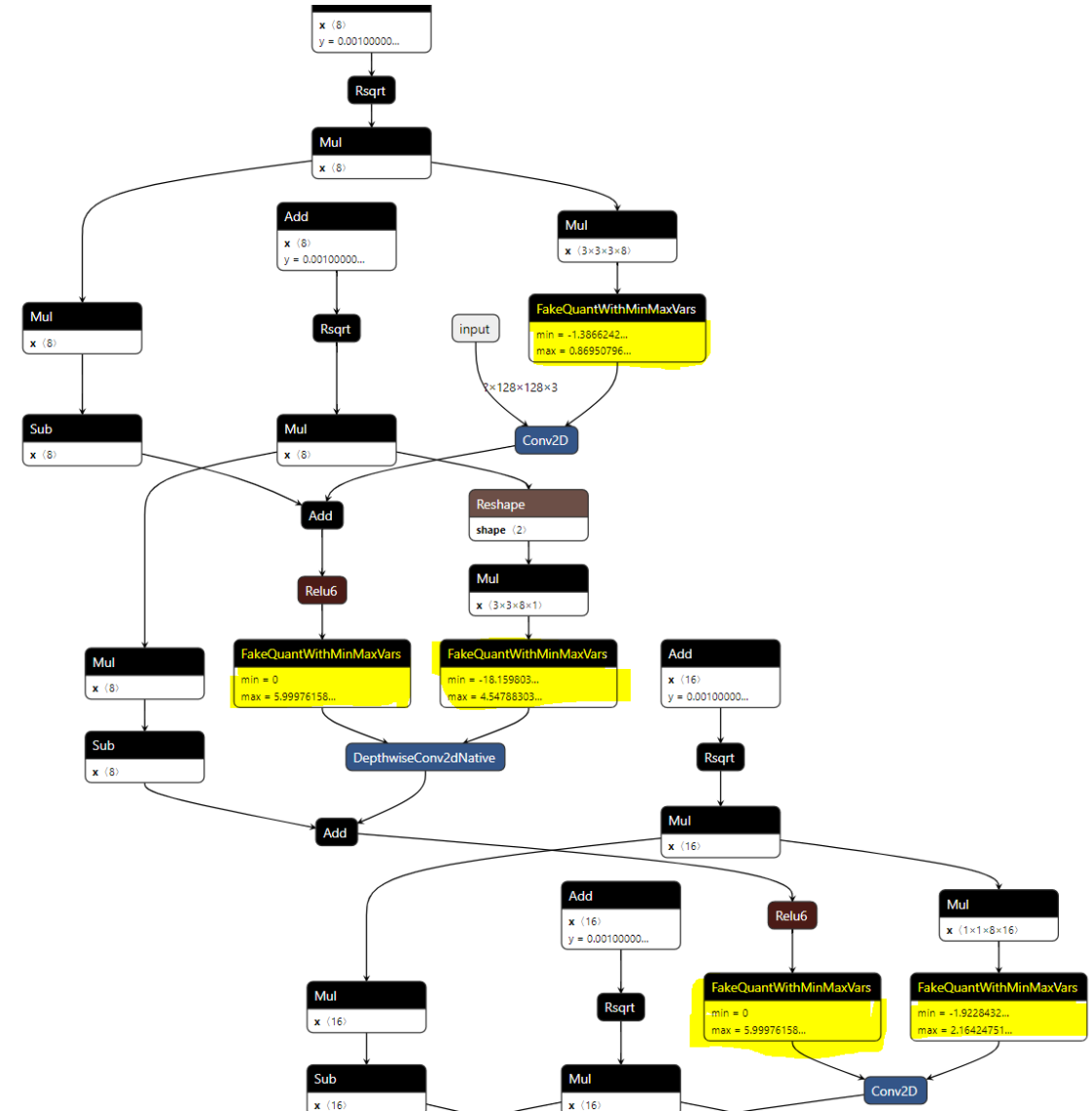
Static Quantization

- weights are quantized ahead of time
- scale and zerobpoint of activations computed with calibration data ahead of time
- Suitable for activation dominate models

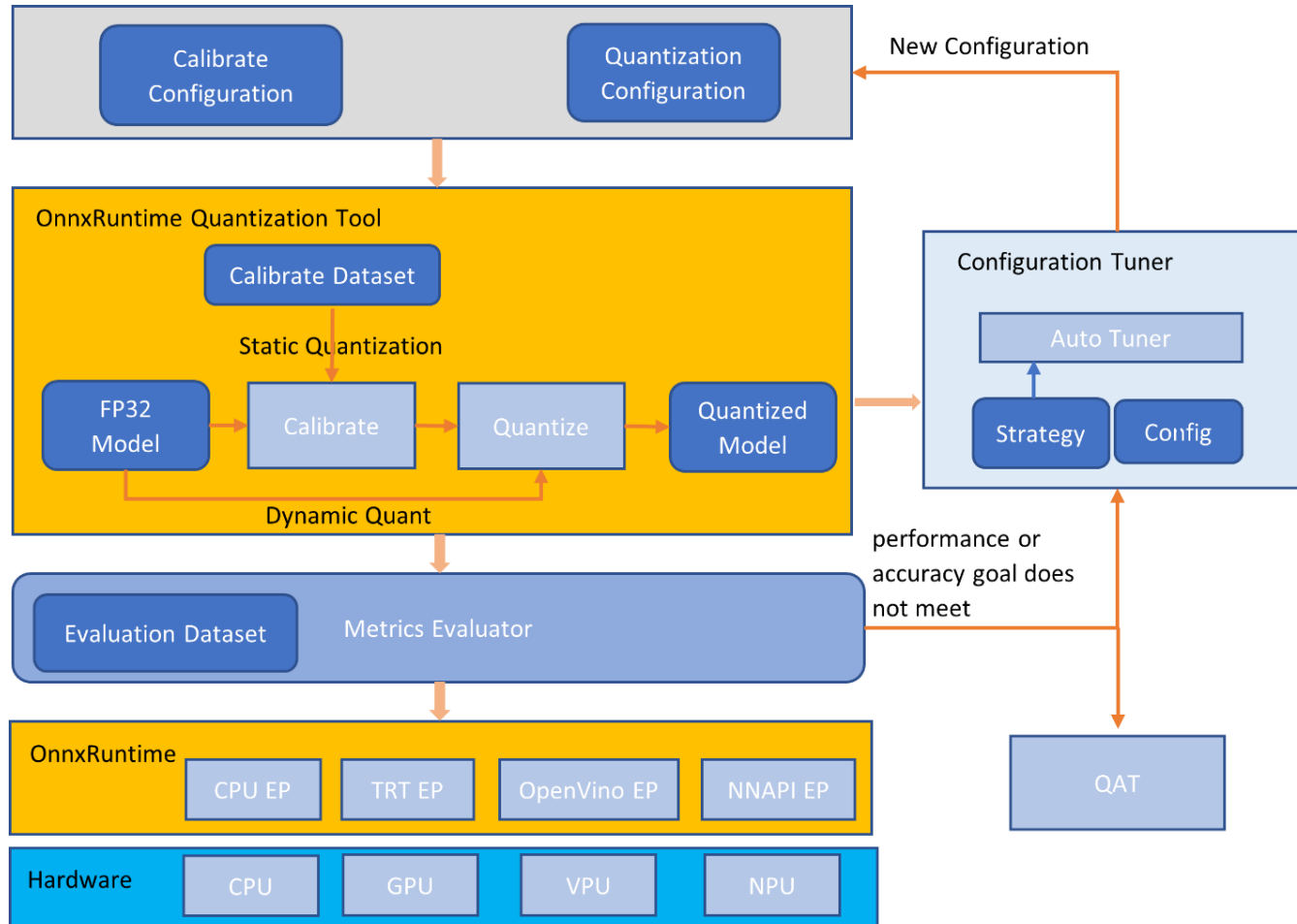


Quantization-Aware Training

- Required when post training quantization does not provide adequate accuracy
- Re-train the model with simulated quantization
 - Both TensorFlow and PyTorch support

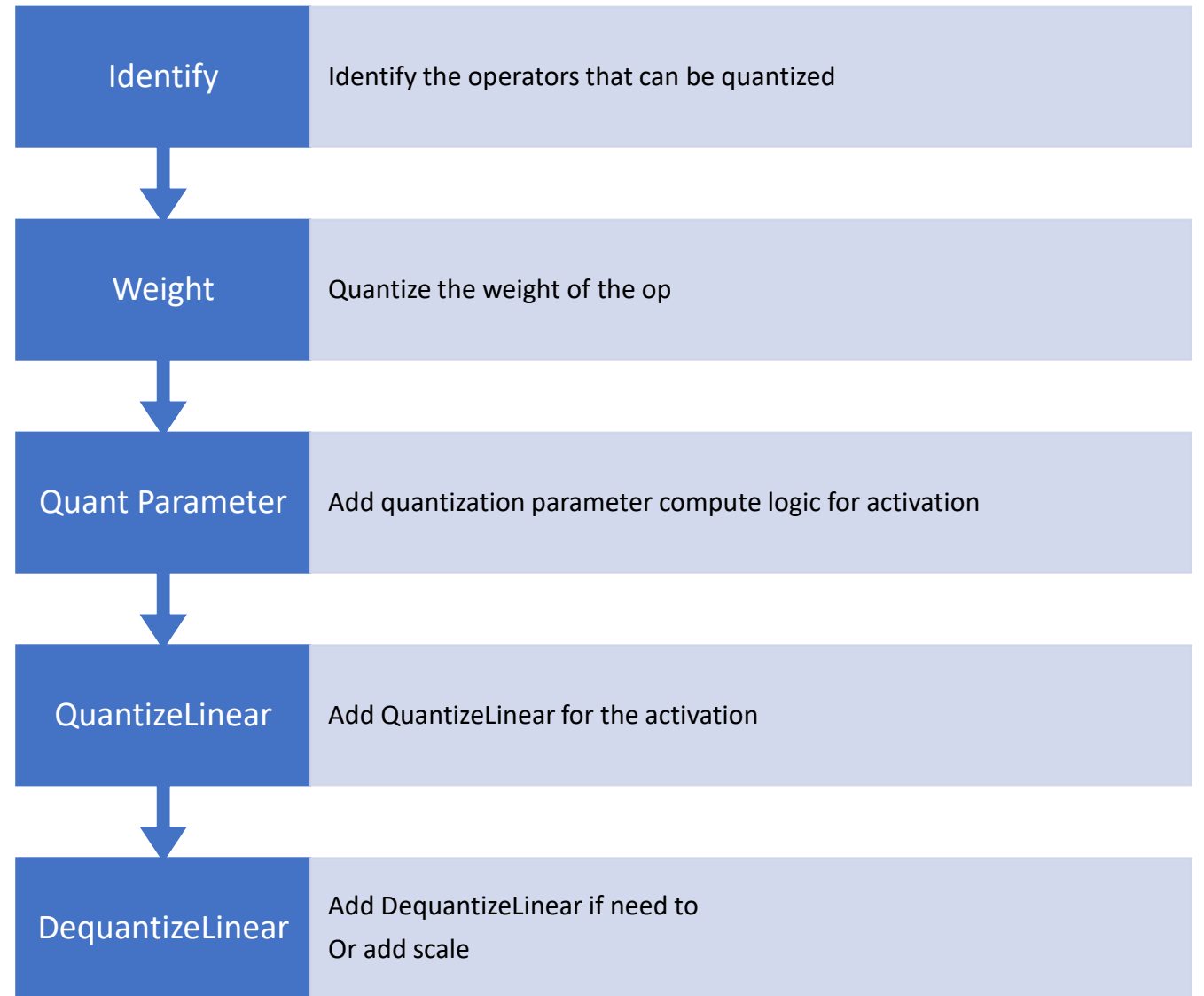


Quantization Workflow and Stack



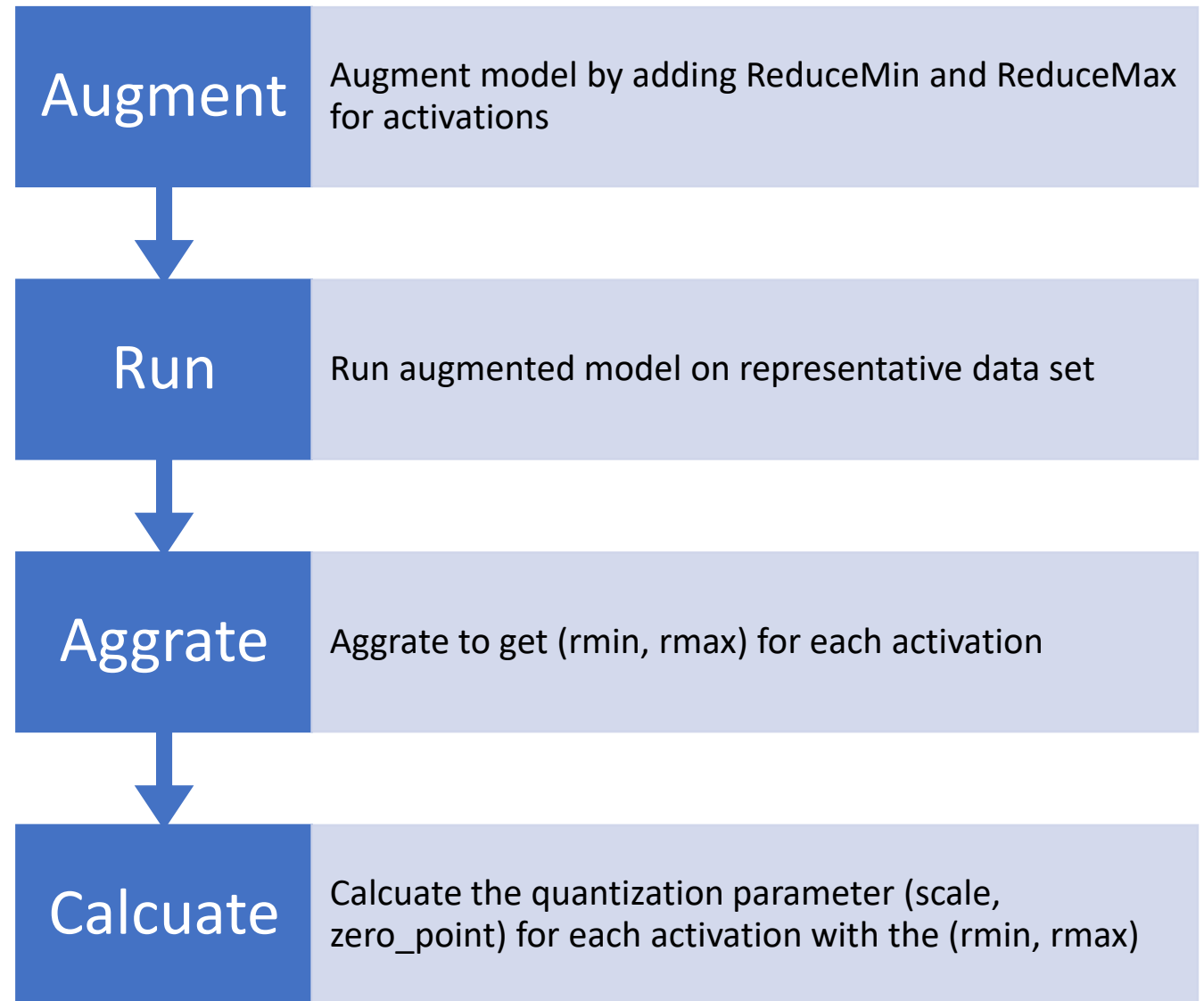
Quantization Tool

- Quantize a post-training fp32 model to 8-bit model
- Support dynamic, static and QAT
- Support QDQ(Quantize/Dequantize), a new format of static quantization and QAT



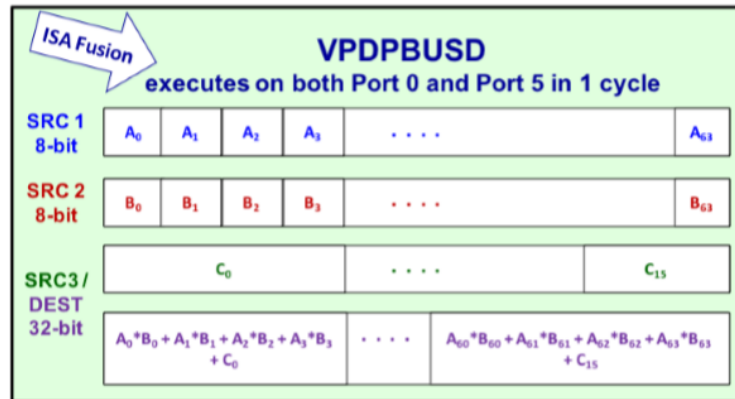
Calibration Tool

- For static quantization
- Uses a small representative data set to calculate the quantization parameters (scale, zero_point) for tensors to quantize
- Support min-max and entropy-based calibration

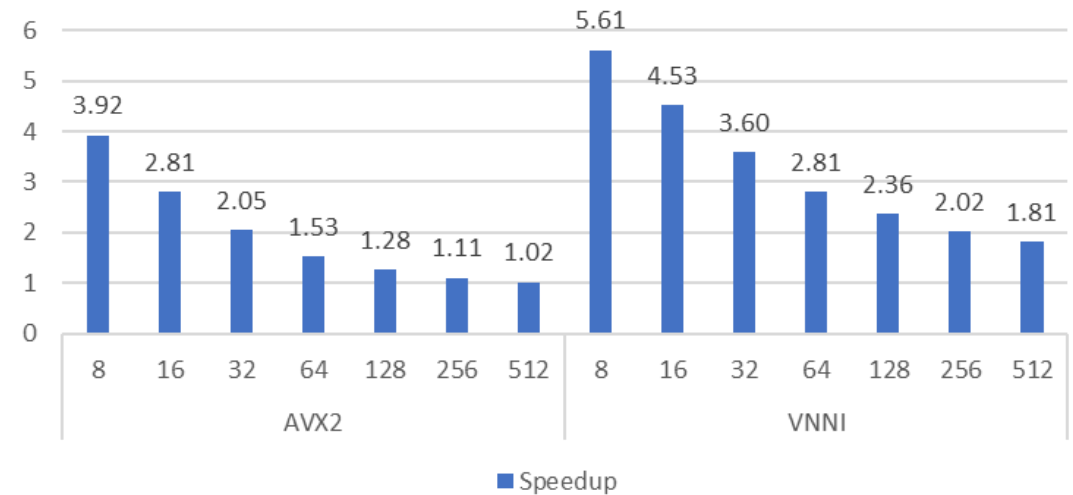


HW Optimization – x86-64

- Old arch mainly benefit from cache
 - Not designed for int8 computation
 - Performance improvement depends on model and input size
- Cascade Lake / Ice Lake supports AVX-512 Vector Neural Network Instructions(VNNI)

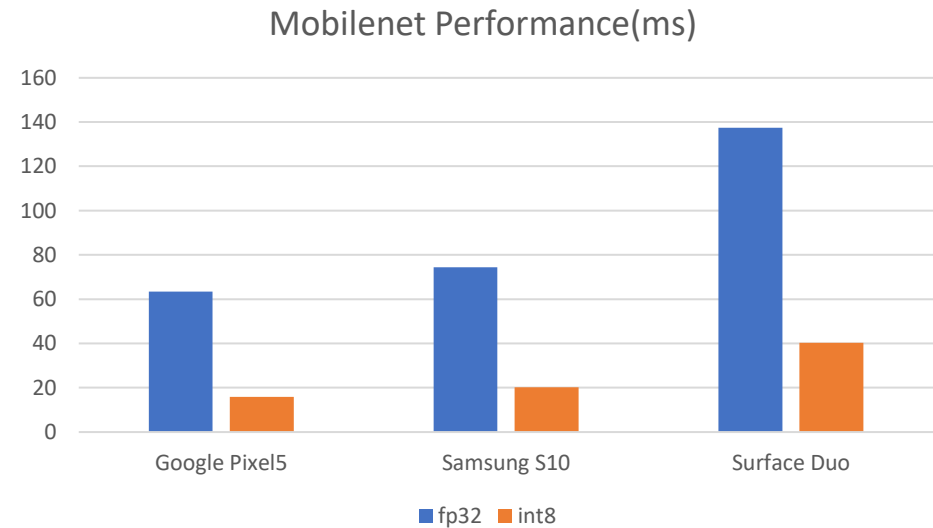
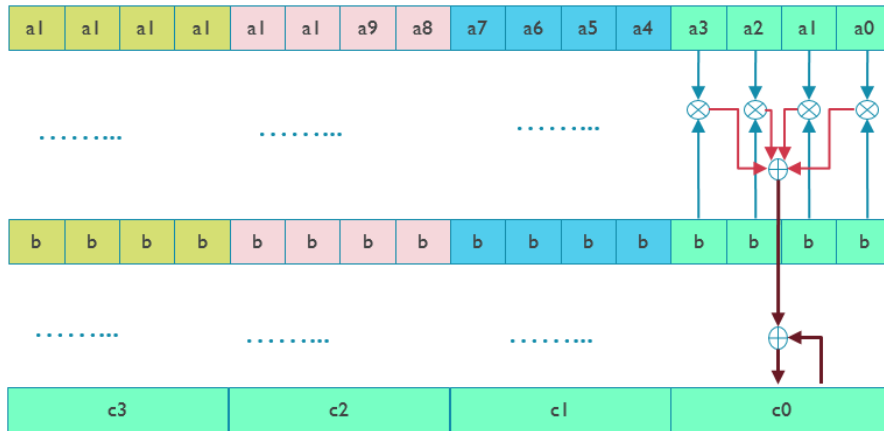


BERT Base Quantization Speedup for different sequence length



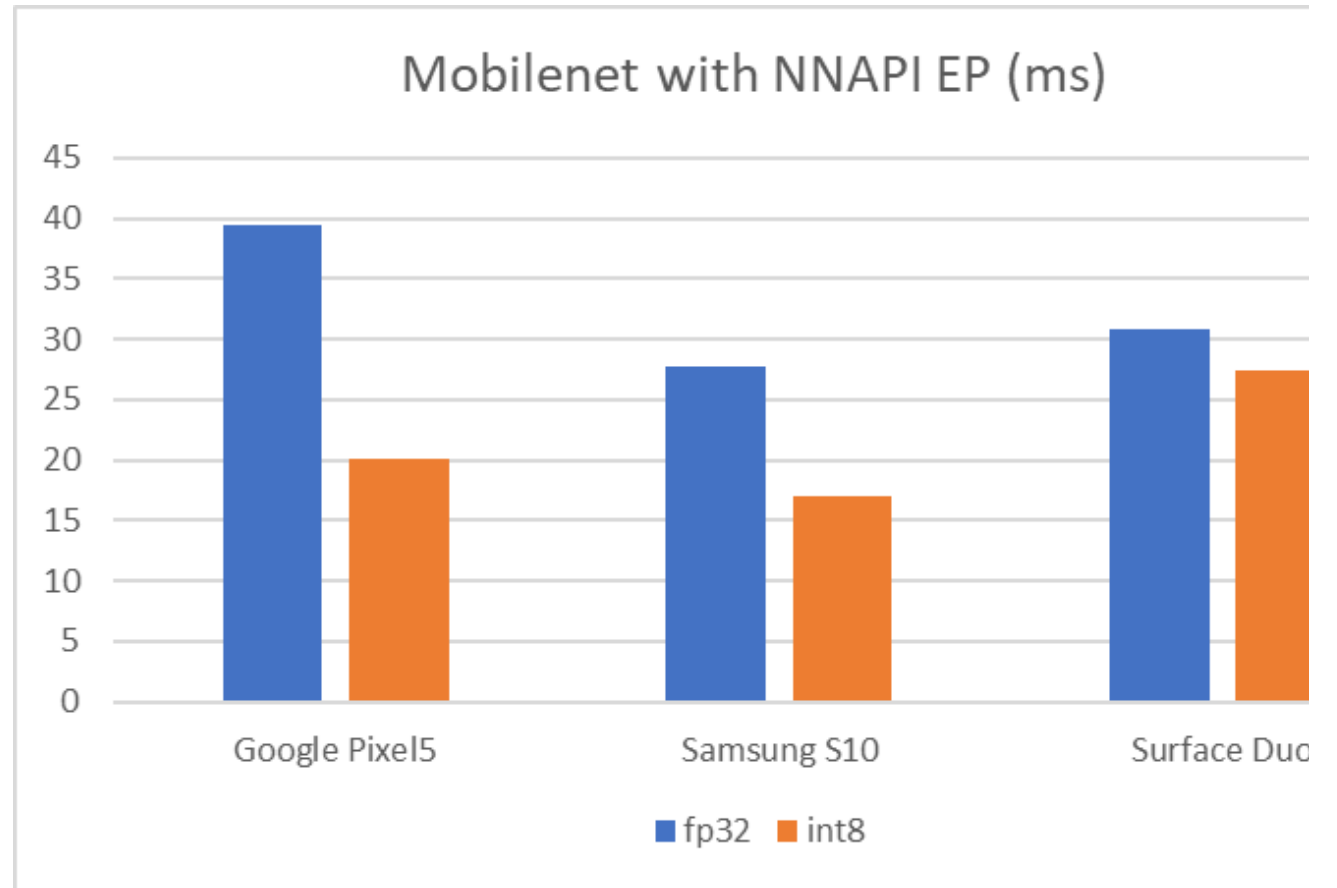
HW Optimization – ARM CPU

- Newer Arm cores support dot product instructions
 - Armv8.2-A optionally
 - Armv8.4-A onwards mandatorily
 - Improve quantization performance by up to 4x



HW Optimization – GPU, VPU, NPU

- TRT EP based on TensorRT for GPU
- OpenVino EP based on OpenVino for VPU
- NNAPI EP based Android Neural Network API (NNAPI) for NPU with Android





ONNX Runtime Mobile
Smaller runtime for mobile inference

Mobile Apps have size limit

- Mobile storage and cellular bandwidth are expensive commodities
 - Google Play store has 100MB limitation for APK size
 - Apple App store will warn user trying to download App over 200MB over the air
- Onnx Runtime full package for arm64 is ~8MB uncompressed and ~2.7MB compressed

ONNX Runtime Mobile



a variant of ONNX Runtime that minimizes binary size for mobile and edge scenarios

Same codebase as ONNX Runtime

Available since ONNX Runtime v1.5, Sept 2020



Includes only required operator kernels in the build

Can also reduce types supported by operator kernels



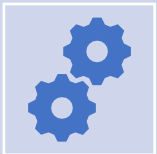
Custom format for the model file

ONNX Runtime Mobile



Runtime usage of ONNX Runtime Mobile is the same as regular ONNX Runtime

C, C++, and Java APIs are available



Moves the preparation of the execution from runtime to offline

Models are ready to execute after loading

Optimization, Graph resolve, type/shape inferencing and other preparations are performed at the time of conversion

Has a smaller binary size and smaller memory footprint

ORT format model

Created from an ONNX model

- Python script handles conversion

During conversion

- ONNX Runtime optimizations are applied
 - e.g. constant folding
- Nodes are assigned to kernels
 - No ONNX schema dependency
 - Significant binary size and memory usage saving

Uses google::flatbuffers

- `ai.onnx;11;AveragePool,Conv,Reshape,Shape,Softmax,Squeeze,Transpose`

Operator Kernel selection

Include only the operator kernels your model(s) will use

Configuration file specifies the kernels to include in the build

- Model conversion script will automatically generate configuration file when converting models
- Configuration file can also be manually created/edited

Example config

Reduced Type Support

Can limit types that operator kernels support

- Model conversion script can automatically detect required types on a per-operator basis
- Alternatively, can specify a global list of types to support

Model based type reduction generally reduces kernel binary size by 25 - 33%

Available in ONNX Runtime v1.7

- March 2021

Mobile Hardware Acceleration

Android Neural Networks API (NNAPI) Execution Provider

- Available in Onnx Runtime 1.6
- December 2020

iOS CoreML Execution Provider (preview)

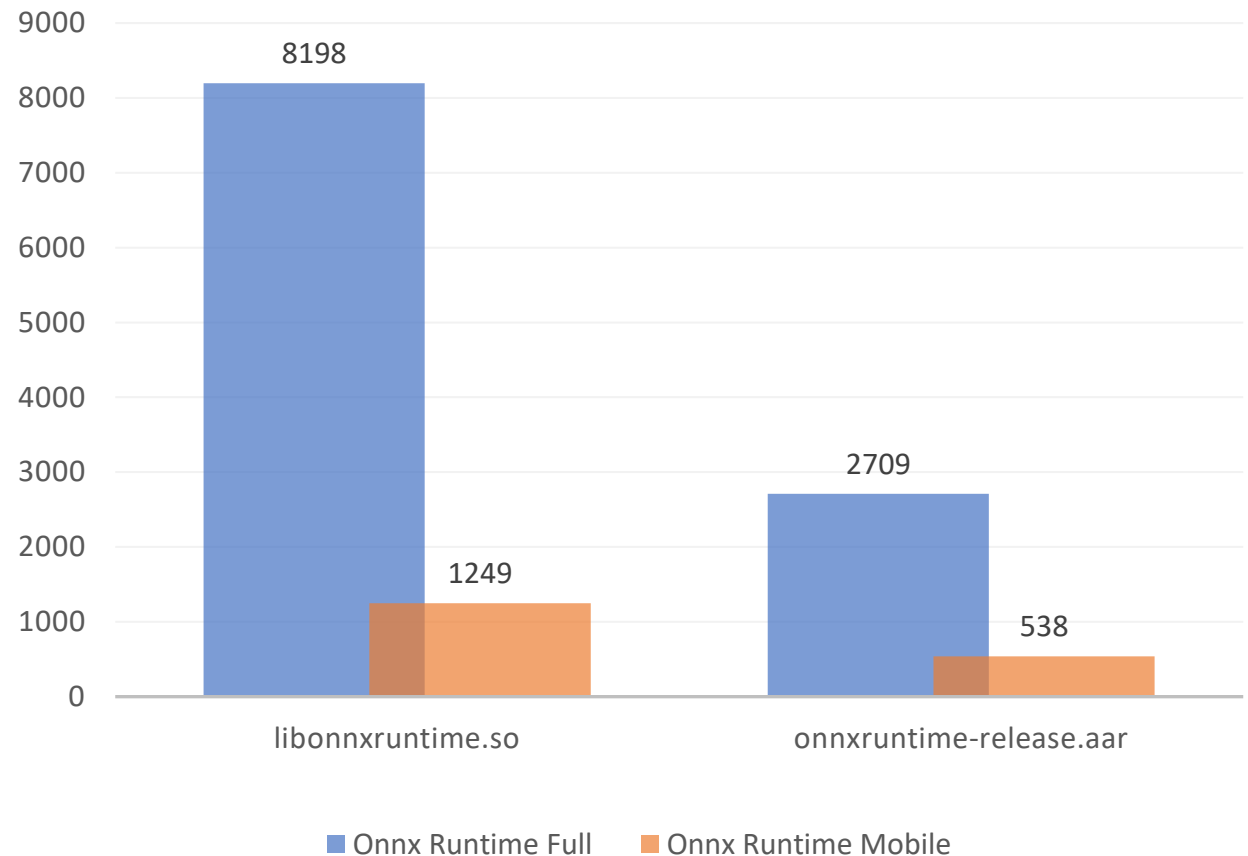
- Available Onnx Runtime 1.8
- May 2021

Binary Size (MobileNet V2)

ONNX Runtime Mobile is built with

- Only operator kernels and types used by MobileNet V2 fp32 and uint8 models included
- Exception disabled
- Traditional ML operators support disabled

Android arm64 Library Size (KB)



Office Text Prediction with ONNX Runtime Mobile

Model

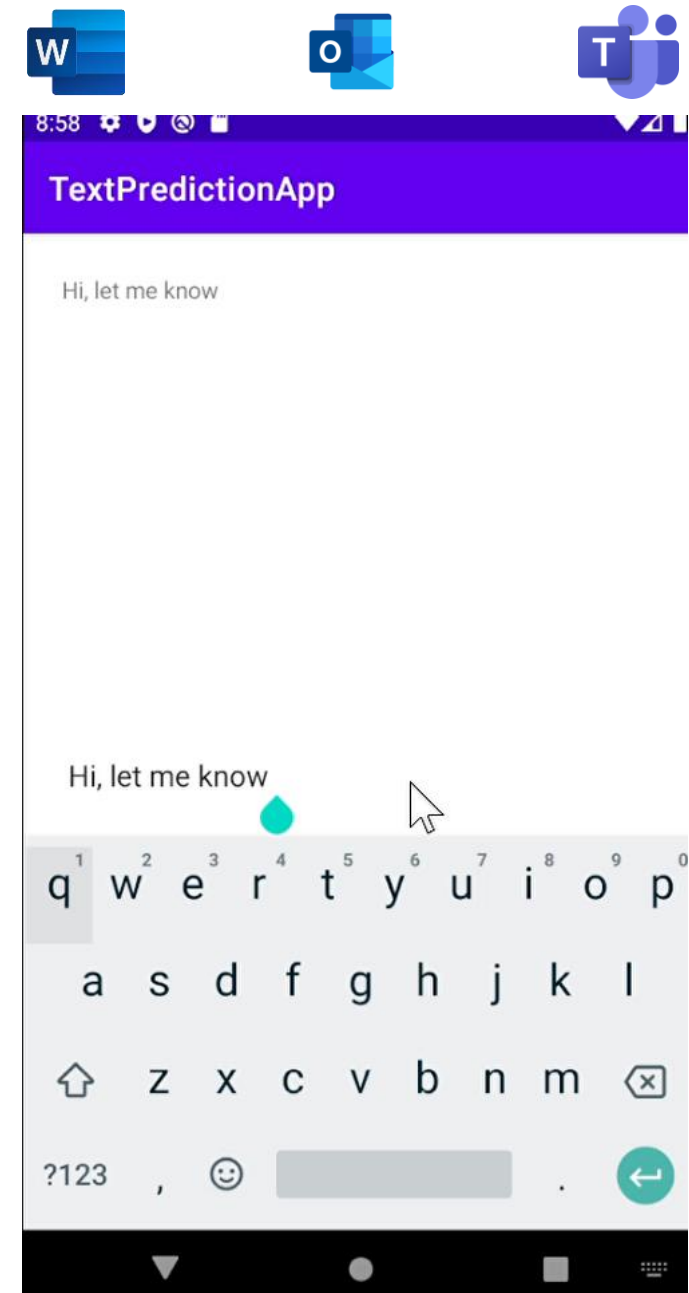
- Size of the model is 5MB
- ONNX Quantization (post training quantization)

Inference Engine

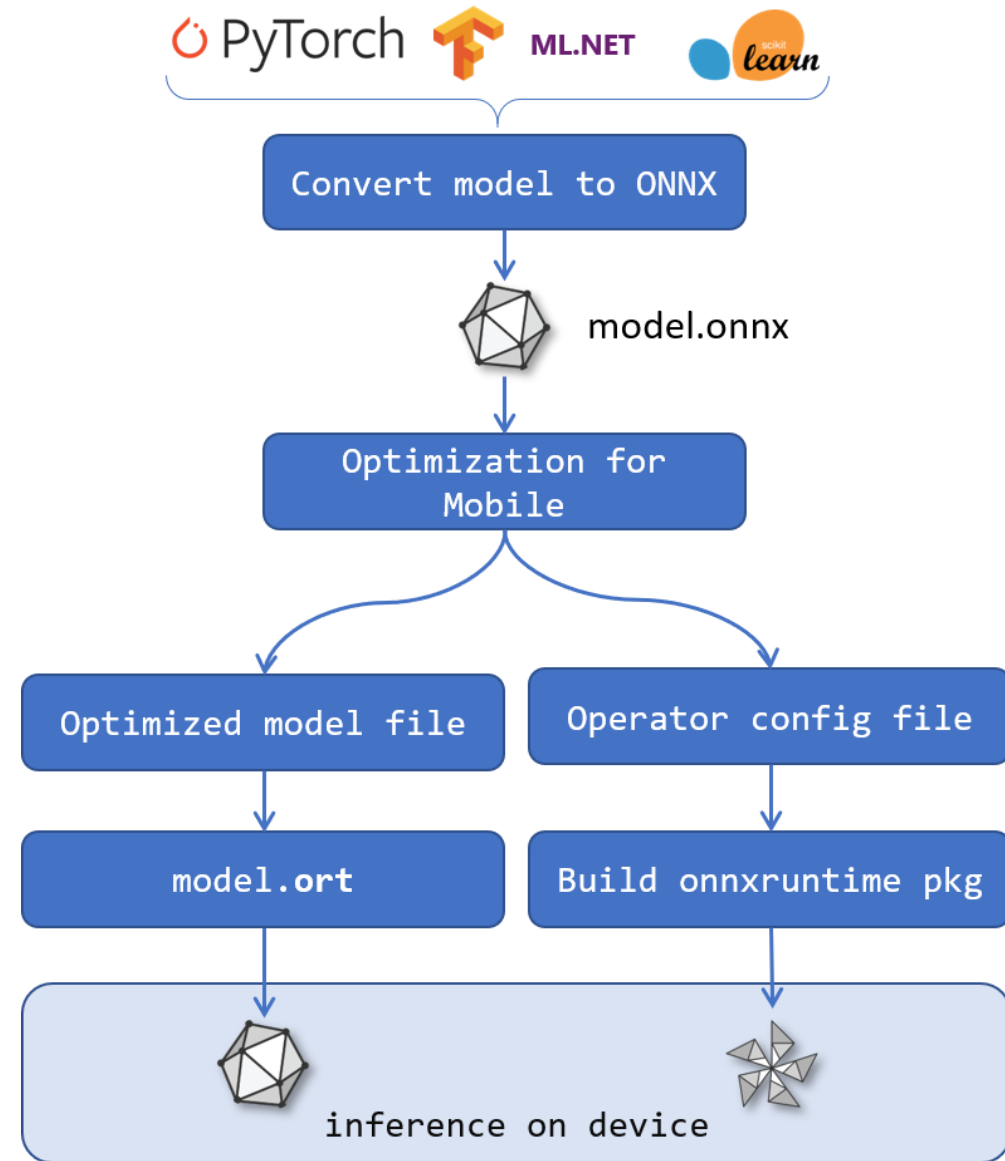
- ONNX Runtime for Win32 and
- ONNX Runtime Mobile for Android and iOS

Runtime Size

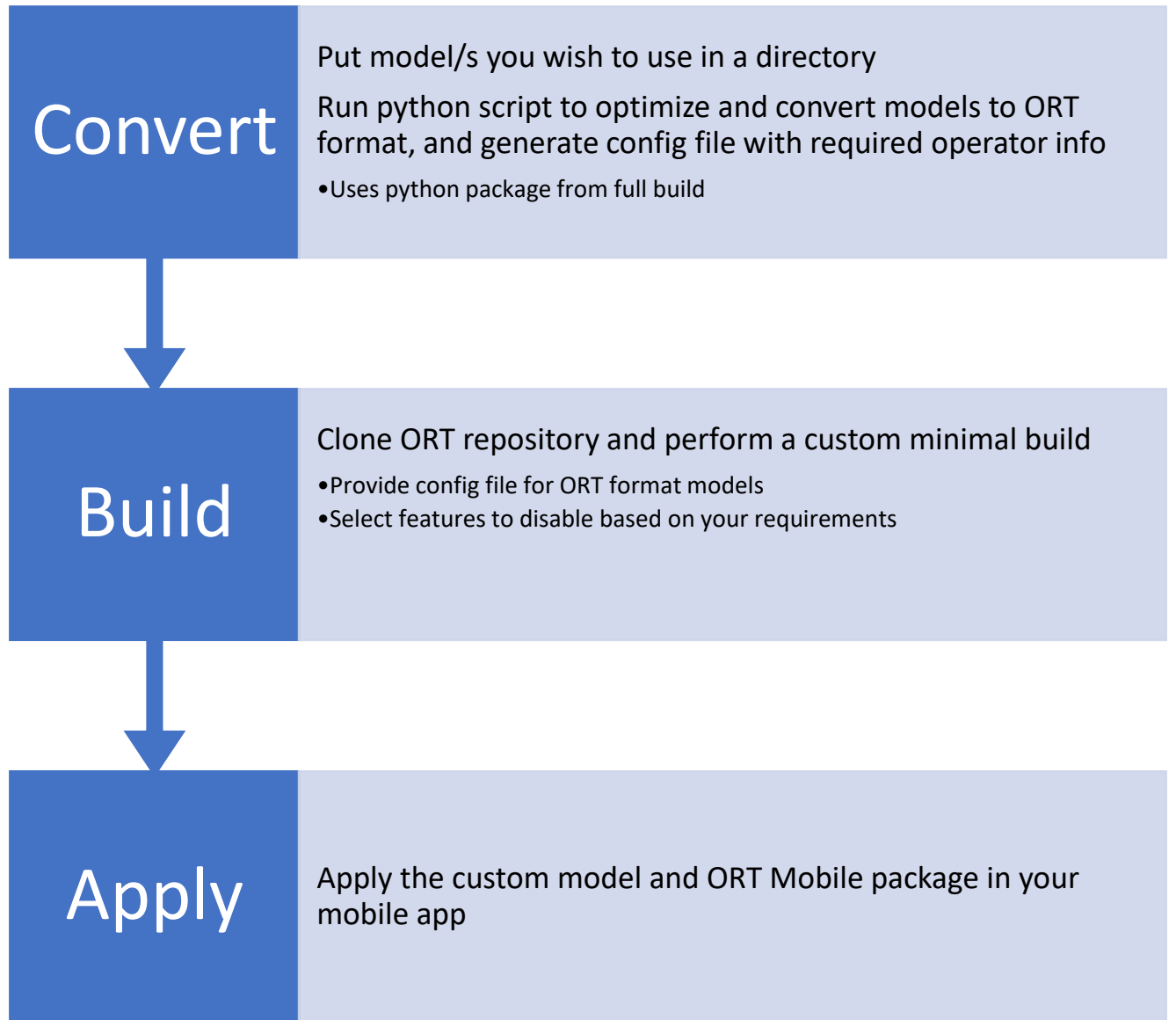
- ONNX Runtime Mobile + TextPrediction Ops: 1.3MB (487KB compressed)



Use ONNX Runtime Mobile



Use ONNX Runtime Mobile





Demo

Executing quantized ONNX Model on Android
using ONNX Runtime Mobile

Demo flow

Real-time image classification mobile APP with Mobilenet



ONNX Runtime Quantization

Export Pytorch Mobilenet to ONNX

Quantize ONNX Mobilenet to INT8



ONNX Runtime mobile

Generate mobile format model with quantized Mobilenet

Build mobile package with quantized Mobilenet

Deploy to mobile device



ONNX Runtime

<https://github.com/microsoft/onnxruntime>

Thank you!