Introduction to Shrinking Models with Quantization-Aware Training and Post-Training Quantization

Robert Cimpeanu

Machine Learning Software Engineer

NXP Semiconductors

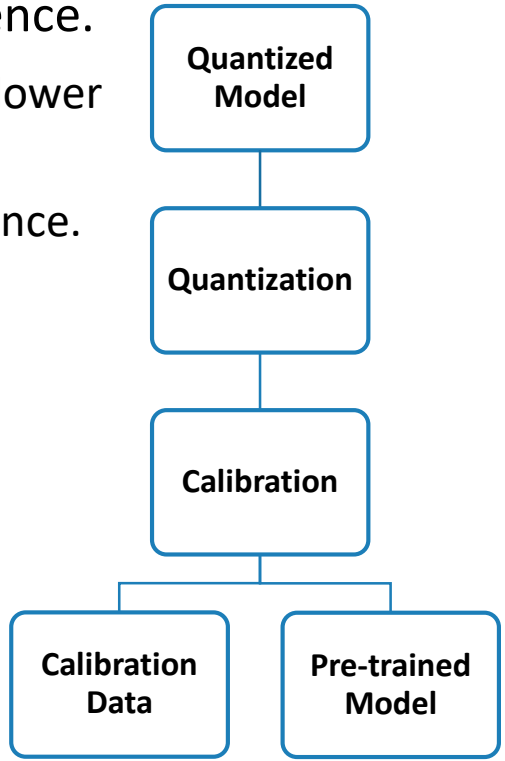
Agenda

- Introduction
- Post-training quantization
- Quantization-aware training
- Architectures and quantization impact
- Quantization workflow
- Conclusions
- Resources

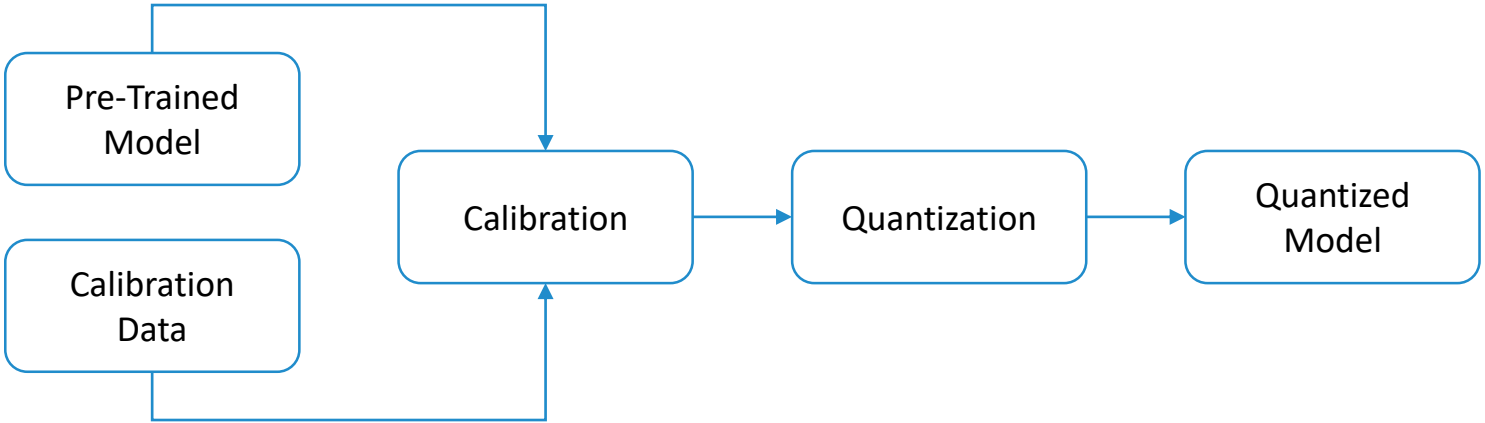
Post-Training Quantization

Post-Training Quantization

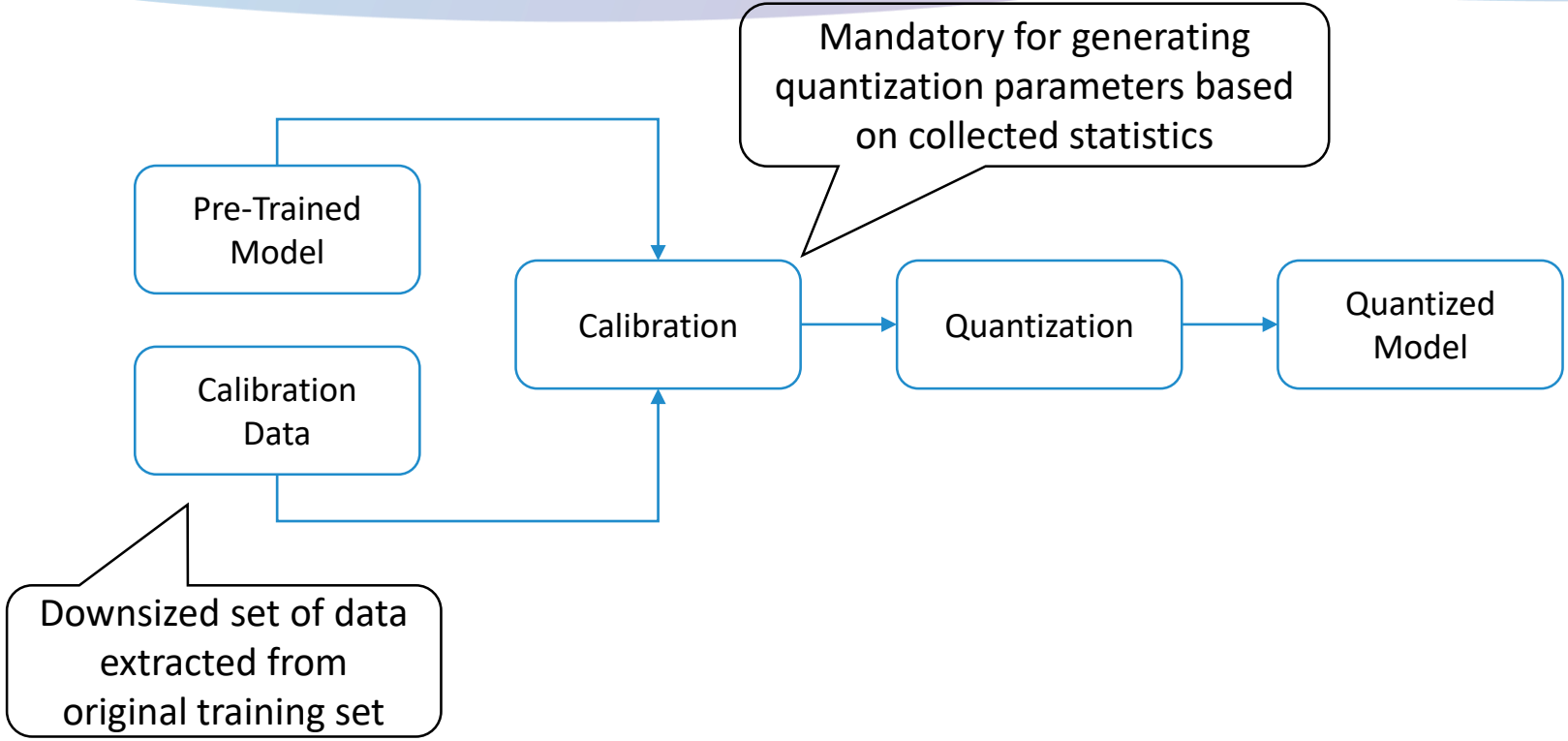
- Method used to reduce computational resources for inference.
  - It maps the data from a high-precision floating-point (FP) to a lower precision, e.g., FP32 –> INT8
  - This process is done after training the model and before inference.
  - Substantial enhancement to model's size and inference speed.
  - It aims to preserve the accuracy of the model.
- Multiple format choices of quantization:
  - Symmetric vs asymmetric
  - Per-channel vs per-tensor
  - Dynamic vs static quantization



Post-Training Quantization



Post-Training Quantization



Post-Training Quantization

- **Calibration dataset** – helps determine the dynamic range of activations in each layer of the neural network.
  - Quantization parameters are computed based on the extracted statistics.
- Key aspects:
  - Representativeness - calibration dataset is representative of the data the model will encounter during deployment.
  - Size – small and unlabeled. Typically, 1-5% of the dataset. This value may vary depending on the complexity of the model or the PTQ technique used.
  - Batch size – the same or at least close to the size that will be used in deployment.

Post-Training Quantization

Advantages

- Easy to implement.
- Reduced memory footprint, especially relevant for edge devices.
- Faster inference.

Disadvantages

- Accuracy loss.
- Sensitive to data distribution.
- Less effective on complex models (e.g., ViT)

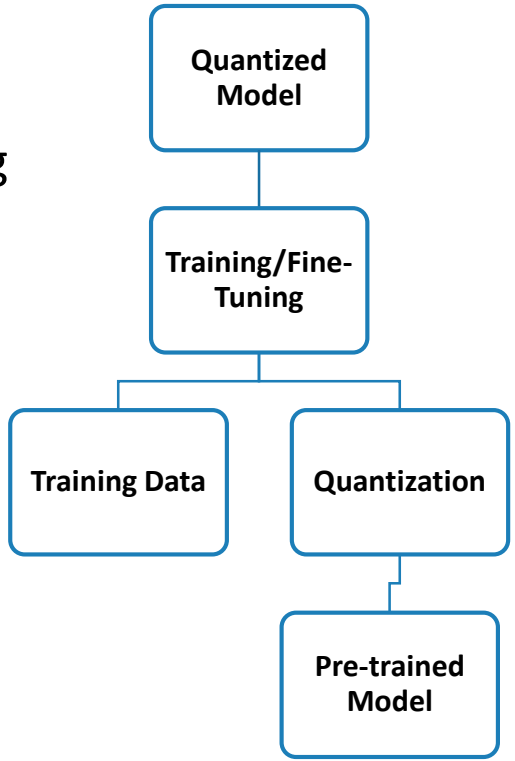
Limitations

- Cannot fine-tune weights post-quantization.
- Cannot handle outliers adaptively without special techniques.

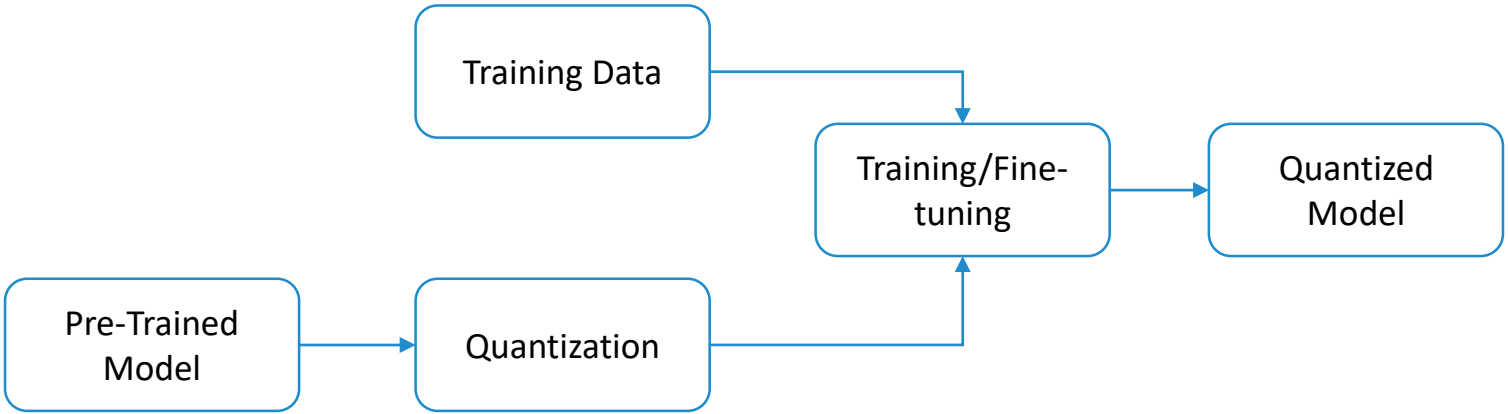
Quantization-Aware Training

Quantization-Aware Training

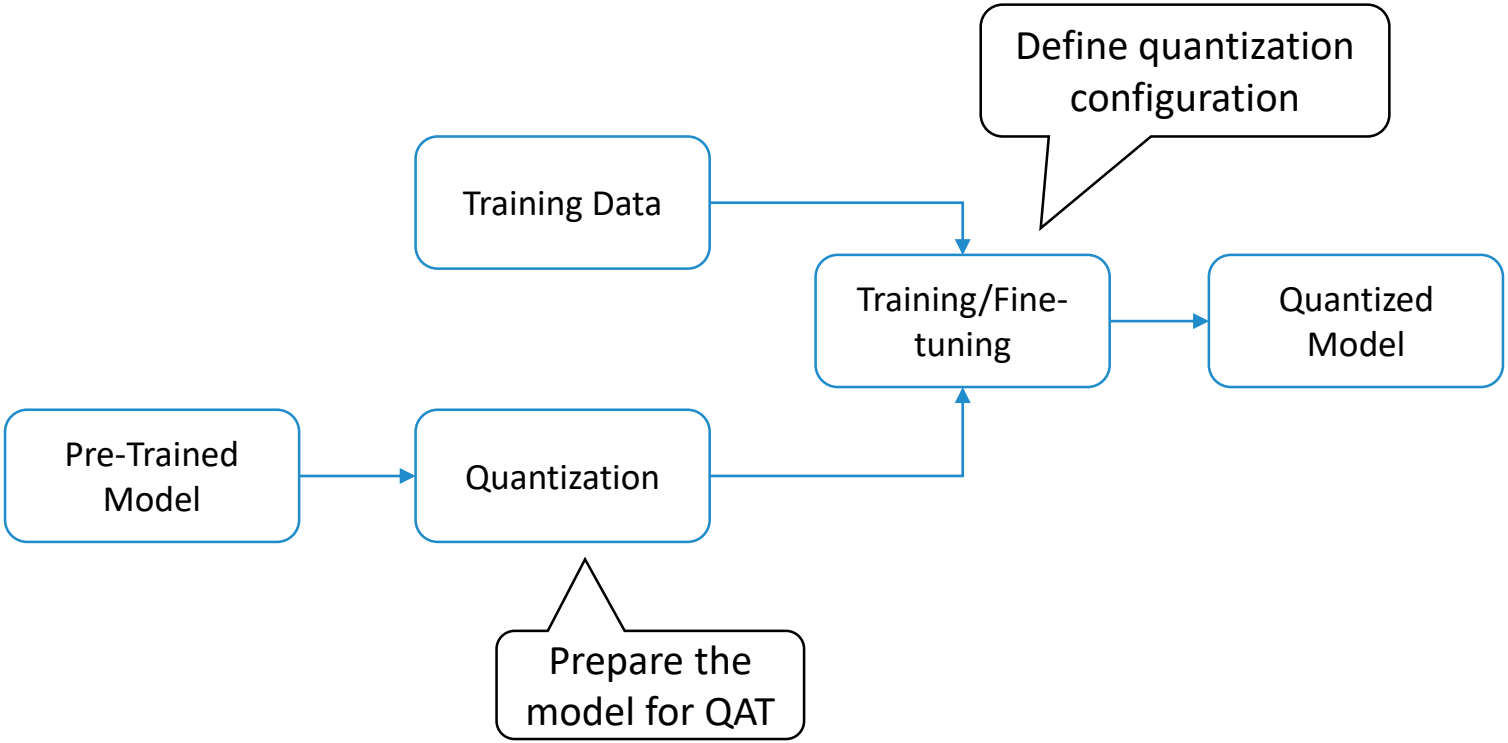
- A more complex method than PTQ, but often better for model's accuracy.
- It involves retraining the model based on the same training dataset.
- As a quantization method its features are:
  - Maps the data from high-precision to a lower-precision representation of data, e.g.: FP32 –> INT8
  - Multiple format choices of quantization:
    - Symmetric vs asymmetric
    - Per-channel vs per-tensor
    - Dynamic vs static quantization



Quantization-Aware Training



Quantization-Aware Training



Quantization-Aware Training

- **Model preparation** – essential steps taken to have a model ready for QAT:
  - Layer fusion – fusing patterns of adjacent layers into a single layer:
    - Convolution, batch normalization, activation -> Convolution
    - Convolution, batch normalization -> Convolution
  - Calibration – running a representative dataset through the model:
    - Helps to collect statistics on activations.
    - Helps to set appropriate quantization parameters.

Quantization-Aware Training

Advantages

- Higher accuracy after quantization.

Disadvantages

- Time consuming process.
- Hardware resources required to re-train the model.
- Requires access to labeled data.

Limitations

- More complex to implement than PTQ.
- The features of this technique may vary depending on the framework used.

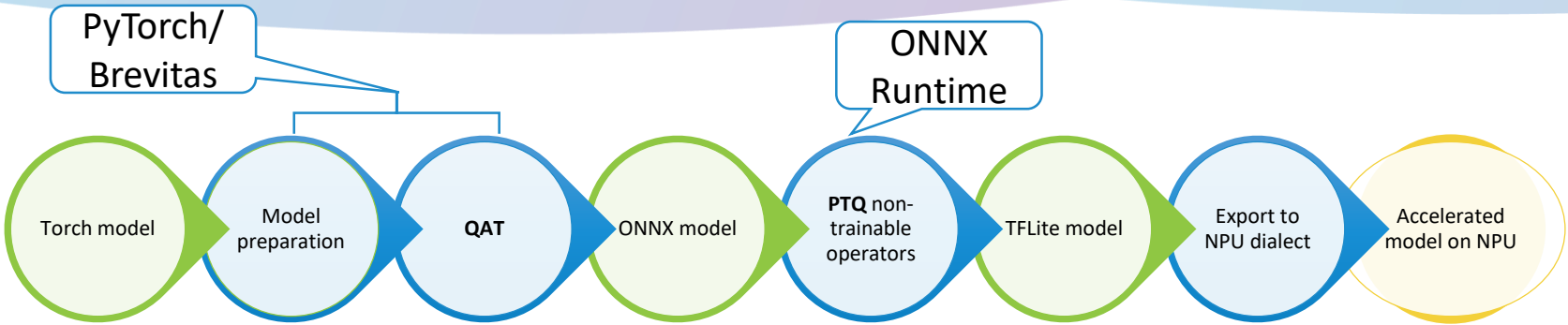
Architectures and Quantization Impact

Architectures and Quantization Impact

- Architectures that may suffer a more significant drop in accuracy after PTQ, and where QAT might be a better approach are:
  - Mobile and embedded models - MobileNet, EfficientNet.
    - Already optimized for efficiency, which is why further quantization can impact accuracy.
    - Especially on blocks: bottleneck/inverted residuals, squeeze-and-excite, Swish/Hard Swish activations, depthwise separable convolutions.
  - Transformer based models - LLM, ViT.
    - Complex architectures that are sensitive to activation range variations.
    - Most sensitive blocks: layer normalization, attention block, GELU activation.

Quantization Workflow

Quantization Workflow



- State of the model where it is evaluated
- Processing steps
- Model's final format

- QAT - using Brevitas for trainable operators such as convolutions, activations and linear layers.
- PTQ - using ONNX Runtime for the rest of the operators that don't have weights to fine-tune.

Quantization Workflow

Model	Format	Top1 Acc.%	Top5 Acc. %
EfficientViT-B1 On Imagenet-1k	PyTorch(FP32)	**79.386**	**94.344**
	PyTorch quantized(mixed precision)	0.1	0.5
	PyTorch after QAT(mixed precision)	72.826	91.126
	ONNX after PTQ(INT8)	72.758	91.080
	TFLite(INT8)	72.332	90.816
EfficientViT-B0 On Imagenet-1k	PyTorch(FP32)	**77.648**	**93.572**
	PyTorch quantized(mixed precision)	0.132	0.506
	PyTorch after QAT(mixed precision)	77.335	93.703
	ONNX after PTQ(INT8)	63.356	84.616
	TFLite(INT8)	61.014	83.130

Conclusions

Conclusions

- Some models are more prone to accuracy loss after applying PTQ.
- QAT is in general the best solution to maintain close-to-original accuracy, but this solution is computationally expensive, because of the need to re-train the model.
- A representative calibration dataset is mandatory for both PTQ and QAT.
- Applying graph optimization techniques, such as layer fusion, is highly recommended for effective QAT.
- Combining both QAT and PTQ can lead to a fully quantized model with a smaller loss of accuracy.

Resources

Resources

Tools used in the workflow

Brevitas – QAT

https://github.com/Xilinx/brevitas

ONNX Runtime – PTQ

https://github.com/microsoft/onnxruntime

“Scaling i.MX Applications Processors’ Native Edge AI with Discrete AI Accelerators”

Ali Osman Ors, Thursday, May 22, 2:05 pm, Enabling Technologies