A White Paper on Neural Network Quantization

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, Tijmen Blankevoort

Qualcomm AI Research

Date: 2025-11-24

Author

Markus Nagel

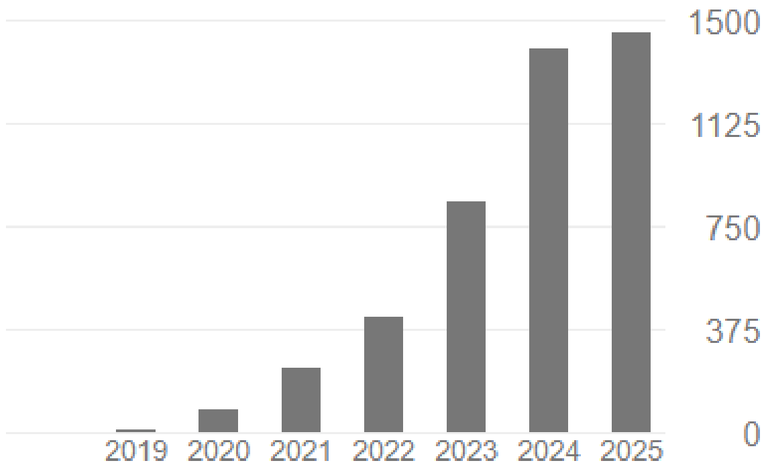
FOLLOW

Qualcomm AI Research
Verified email at qualcomm.com
Machine learning Deep Learning

TITLE	CITED BY	YEAR
A White Paper on Neural Network Quantization M Nagel, M Fournarakis, RA Amjad, Y Bondarenko, M van Baalen, ... arXiv preprint arXiv:2106.08295	937	2021
Up or Down? Adaptive Rounding for Post-Training Quantization M Nagel, RA Amjad, M van Baalen, C Louizos, T Blankevoort Proceedings of the 37th International Conference on Machine Learning	843	2020
Data-Free Quantization through Weight Equalization and Bias Correction M Nagel, M Baalen, T Blankevoort, M Welling Proceedings of the IEEE International Conference on Computer Vision, 1325-1334	789	2019
LSQ+: Improving low-bit quantization through learnable offsets and better initialization Y Bhalgat, J Lee, M Nagel, T Blankevoort, N Kwak Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern ...	347	2020
Understanding and Overcoming the Challenges of Efficient Transformer Quantization Y Bondarenko, M Nagel, T Blankevoort arXiv preprint arXiv:2109.12948	206	2021
Bayesian bits: Unifying quantization and pruning M Van Baalen, C Louizos, M Nagel, RA Amjad, Y Wang, T Blankevoort, ... Advances in neural information processing systems 33, 5741-5752	176	2020

Cited by

	All	Since 2020
Citations	4534	4473
h-index	22	22
i10-index	30	29



Background

What is **Quantization**?

- Imagine your model normally thinks using 32-bit float numbers (very precise, like using a microscope)
- Quantization forces it to think using 8-bit integers (rougher, like using reading glasses).

Why it is important?

- Models become **smaller** (4× smaller)
- Models run **faster**, especially on mobile/embedded/chips
- Power consumption becomes **much lower**

Quantization Fundamentals and Hardware

1. Hardware background

Describe a typical neural network inference accelerator: matrix-vector multiply (MAC) units, accumulators. for eg, processing elements, accumulators.

Hardware Operation: Neural network accelerators are optimized for Multiply-Accumulate (**MAC**) operations. Using low-bit fixed-point representations like INT8 reduces the size and energy consumption of MAC operations and data transfer.

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

To move from floating-point to the efficient fixed-point operations, we need a scheme for converting floating-point vectors to integers.

$$A_n = \mathbf{b}_n + \sum_m C_{n,m}$$

$$\widehat{\mathbf{x}} = s_{\mathbf{x}} \cdot \mathbf{x}_{\text{int}} \approx \mathbf{x}$$

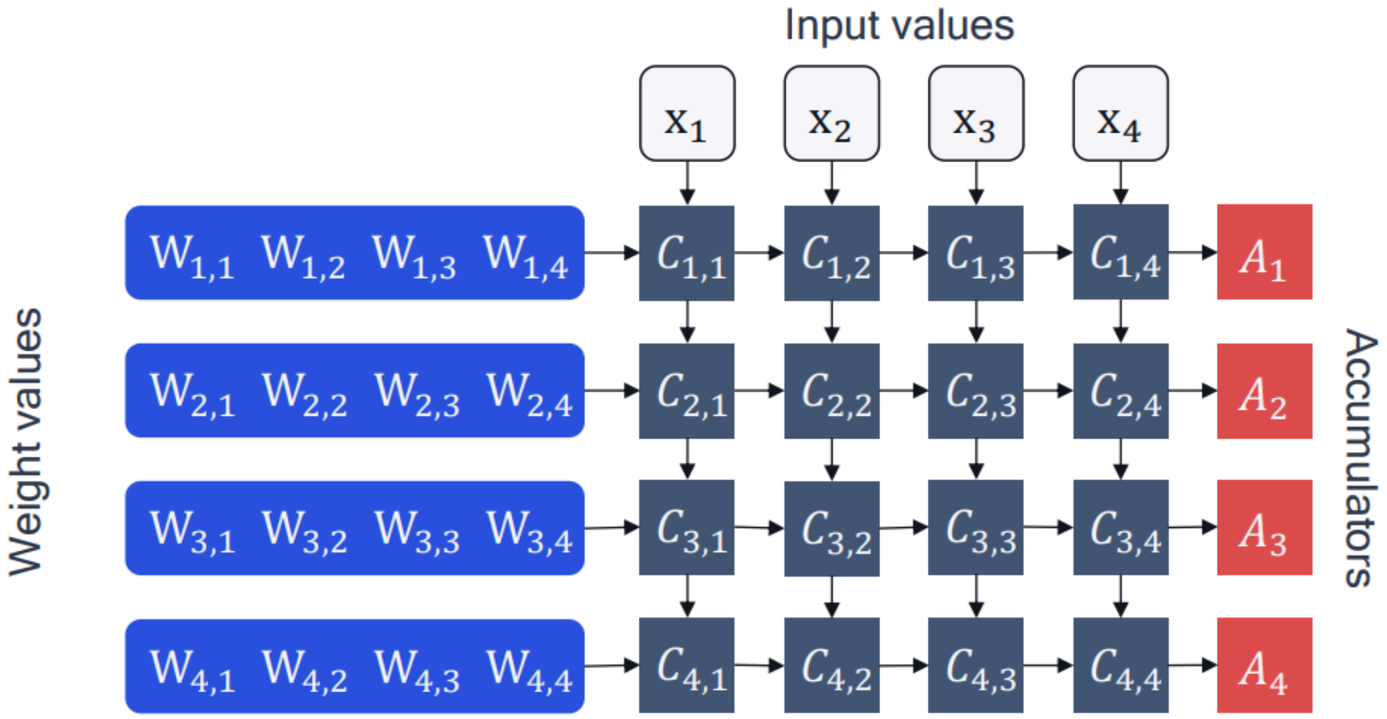


Figure 1: A schematic overview of matrix-multiply logic in neural network accelerator hardware.

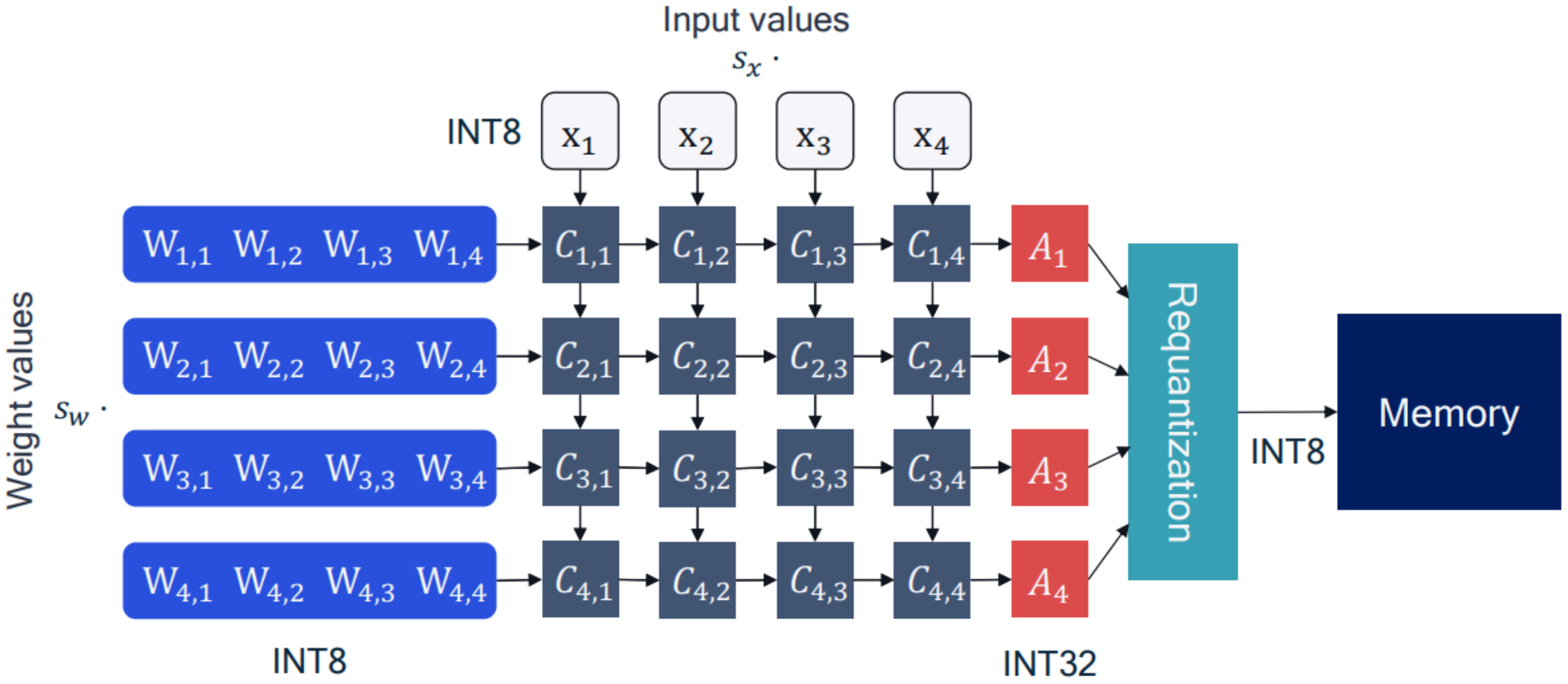


Figure 2: A schematic of matrix-multiply logic in an neural network accelerator for quantized inference.

Quantization Fundamentals and Hardware

2. Uniform Affine Quantization: Uniform affine quantization, also known as asymmetric quantization.

- It is defined by the scale factor (*s*), zero-point (*z*), bit-width (*b*)
- Compare symmetric quantization (zero-point = 0) vs asymmetric.
- The quantized value is defined by the function:

$$\widehat{\mathbf{x}} = q(\mathbf{x}; s, z, b) = s\left[\text{clamp}\left(\left\lfloor \frac{\mathbf{x}}{s} \right\rceil + z; 0, 2^b - 1\right) - z\right]$$

Quantization granularity: Per-tensor vs per-channel vs per-group.
Pros/cons: Per-channel can improve accuracy but increases hardware complexity

Experiments

There are two types of quantization techniques.

- Post-Training Quantization
- Quantization-aware training

1.Post-Training Quantization

- Take a pretrained FP32 model and quantize weights/activations with no or minimal retraining.
- Requires little or no calibration data.
- Advantage: lightweight, low engineering effort, fast deployment.
- Limitation: Works well typically for 8-bit, but for lower bit widths (e.g., 4-bit) the accuracy gap to FP32 may increase.
- Pipeline overview: range-setting → optional corrections (equalization, bias correction) → final quantized model.

Post-Training Quantization Pipeline

- Quantization range setting
- Cross-Layer Equalization
- Bias correction
- AdaRound
- Debugging

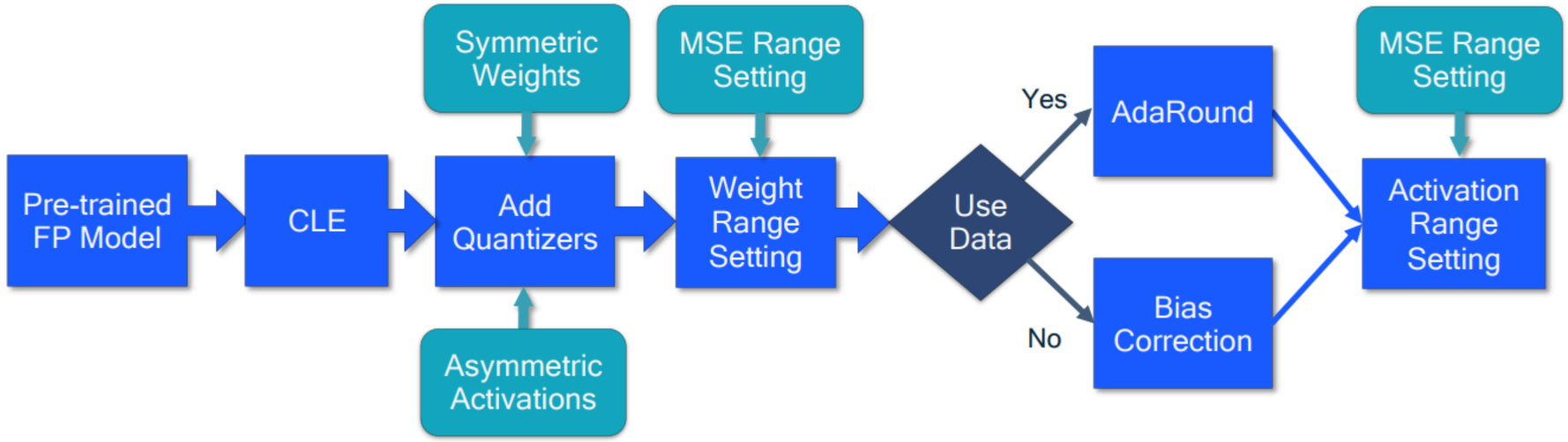


Figure 3: Standard PTQ pipeline

2. Quantization-aware training

- During training (or fine-tuning) simulate quantization in the forward pass so the network adapts to quantization noise.
- Requires labelled data, more computing effort than PTQ.
- Advantage: Enables lower bit widths (e.g., 4-bit or less) with competitive accuracy.
- Pipeline overview: Insert quantizer blocks in training graph (after weights/activations) → train/fine-tune to compensate for quantization effects → deploy quantized model.

Quantization-Aware Training Pipeline

- Quantizer Simulation
- Batch normalization
- Initialization

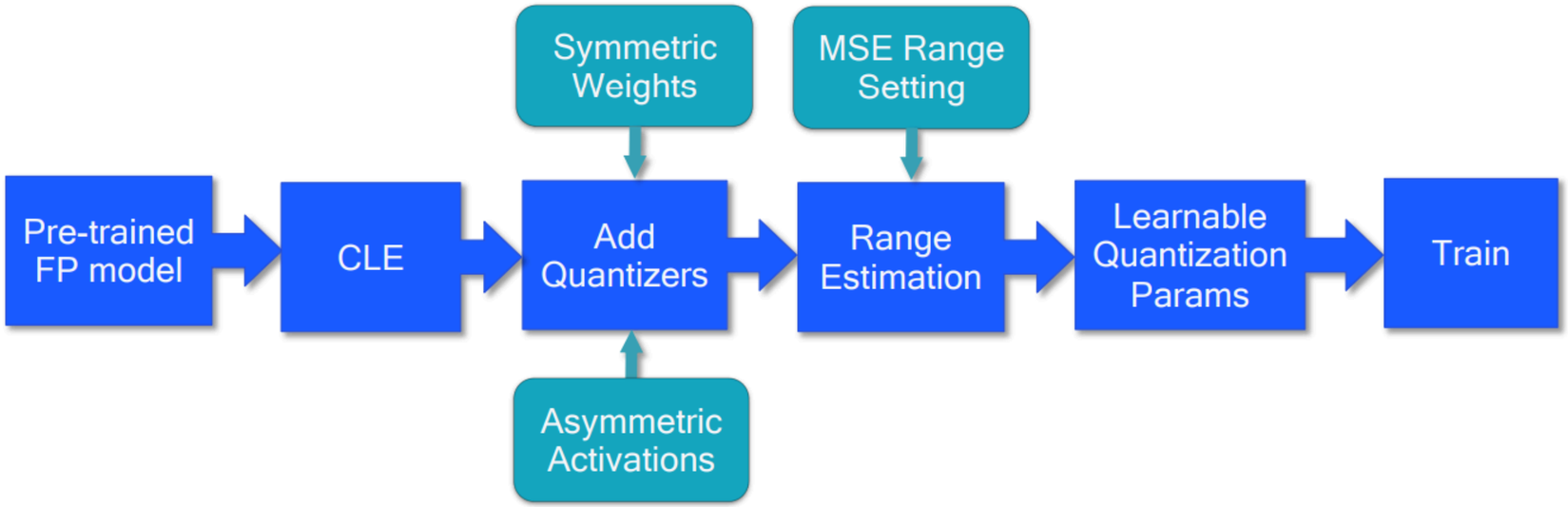


Figure 4: Standard QAT pipeline

3. Simulating Quantization on Floating-Point Hardware

1. In real deployment, neural networks run on **fixed-point hardware**, such as microcontrollers or low-power accelerators.
2. But during training, we use GPUs/CPUs which operate in **floating-point**.
3. To evaluate how well the model would run on a quantized device, it **simulate quantized behavior** on floating-point hardware.
4. **Quantization Simulation** is a key part of **Quantization-Aware Training (QAT)**.

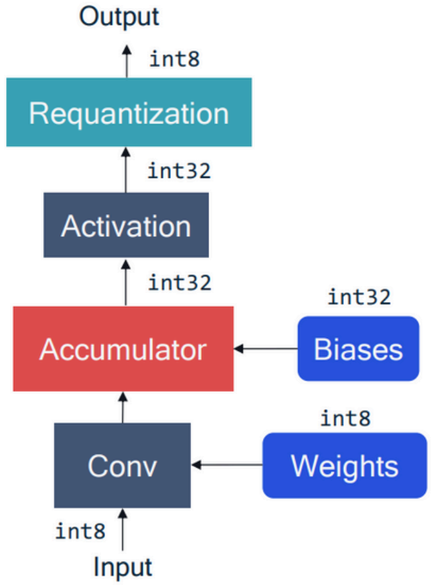


Fig 5(a) Diagram for quantized on-device inference with fixed-point operations.

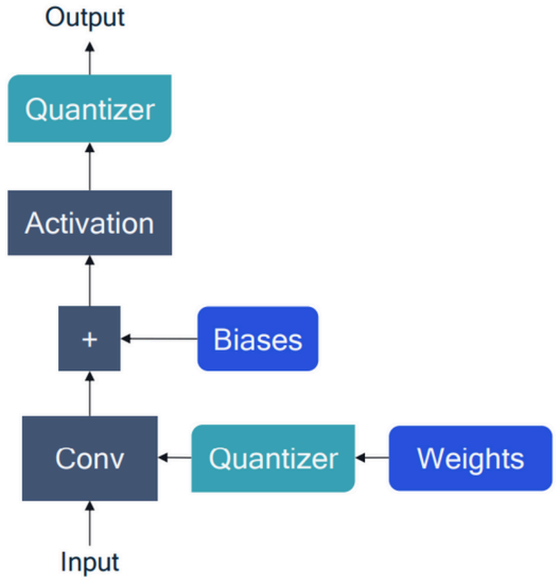


Fig 5(b) Simulated quantization using floating-point operations

Experiments & Key Results for PTQ

Table 1. Performance of our standard PTQ pipeline for various models and tasks.

Models	FP32	Per-tensor		Per-channel	
		W8A8	W4A8	W8A8	W4A8
ResNet18	69.68	69.60	68.62	69.56	68.91
ResNet50	76.07	75.87	75.15	75.88	75.43
MobileNetV2	71.72	70.99	69.21	71.16	69.79
InceptionV3	77.40	77.68	76.48	77.71	76.82
EfficientNet lite	75.42	75.25	71.24	75.39	74.01
DeeplabV3	72.94	72.44	70.80	72.27	71.67
EfficientDet-D1	40.08	38.29	0.31	38.67	35.08
BERT-base†	83.06	82.43	81.76	82.77	82.02

Experiments & Key Results for QAT

Table 2. Performance of our standard QAT pipeline for various models and tasks.

Models	FP32	Per-tensor			Per-channel		
		W8A8	W4A8	W4A4	W8A8	W4A8	W4A4
ResNet18	69.68	70.38	69.76	68.32	70.43	70.01	68.83
ResNet50	76.07	76.21	75.89	75.10	76.58	76.52	75.53
InceptionV3	77.40	78.33	77.84	77.49	78.45	78.12	77.74
MobileNetV2	71.72	71.76	70.17	66.43	71.82	70.48	66.89
EfficientNet lite	75.42	75.17	71.55	70.22	74.75	73.92	71.55
DeeplabV3	72.94	73.99	70.90	66.78	72.87	73.01	68.90
EfficientDet-D1	40.08	38.94	35.34	24.70	38.97	36.75	28.68
BERT-base	83.06	83.26	82.64	78.83	82.44	82.39	77.63

Conclusion

- Deep learning is widely deployed in edge devices, increasing the need for fast and power-efficient inference.
- Quantization reduces floating-point models to fixed-point formats, enabling lower latency and energy consumption.
- Two main quantization approaches: Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT).
- PTQ offers a lightweight pipeline achieving near-FP32 accuracy for 8-bit and even 4-bit weights.
- QAT simulates quantization during training, achieving more aggressive activation/weight quantization with minimal accuracy loss.

SEOULTECH
SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

Thank You!!

Any question?

BrAIn Lab.
Brain and Artificial Intelligence Lab.