

Quantization Techniques Technical Overview and Improvement Expectations

R3AL.AI

July 2026

1 Introduction

This document provides an overview of quantization techniques for computer vision models - these can be based on the CNN architecture or on the transformer architecture.

2 General terminology

- Weights, activations.
- Scaling factor, zero-point. These concepts are relevant for INT8 or INT4 quantization. With INT8 quantization, original “full-precision” values are approximated by one of 255 candidates that uniformly span a range determined by a scaling factor s and a zero-point z . More specifically, INT8 integers are one of 255 integer values in $\{-127, -126, \dots, 127\}$ ¹. Suppose we want to approximate some arbitrary set of full-precision values $\{-0.012, 0.3, \dots, 2.108\}$. With scaling factor $s = \frac{254}{2.108 - (-0.012)} = 119.81$ and zero-point $z = -126$ the INT8 integers correspond to the candidates $\{\frac{-127-z}{s}, \frac{-126-z}{s}, \dots, \frac{127-z}{s}\} = \{-0.008, 0, 0.008, \dots, 2.11\}$. The zero-point chooses one of the integers (namely z) to correspond to a candidate 0. The scaling factor shrinks or broadens the integer difference of an INT8 integer with the zero-point to get a suitable range of candidate values. For faster computation, the zero-point is often chosen to be 0, so that the INT8 integer 0 actually represents the candidate 0. The candidates are called “quantization levels”.
- Block format²/granularity. The block format specifies the blocks of weights/activations that share a scaling factor. Per-tensor or per-channel are two common options. There is a speed versus accuracy trade-off here.
- Calibration.
- PTQ versus QAT

¹For symmetry, the value 0 is chosen to be encoded in INT8 with 10000000 as well as 00000000. Hence just 255, not 256, integers are representable in INT8.

²NVIDIA terminology

- Static quantization, dynamic quantization. Static and dynamic quantization differ only in the way they calculate the scales to quantize the activations. Static quantization uses a couple of representative calibration batches to determine sensible scale values for the activations and keeps those fixed during inference. Dynamic quantization determines the scales for the activations at runtime. This causes some computational overhead (more latency, this paper³ reported latency increase with a factor 1.3 - 7 x), but makes the quantization of the activations more precise. For this reason, when activations tend to vary a lot (transformer architectures) dynamic quantization might be used.

Using the above, quantization decreases memory requirements of running/storing the model substantially (/2). Typical speedups for INT8 quantization are 1.5 - 2 x, with a minimal decrease in accuracy, compared to running/storing the model in FP16.

3 Advanced techniques

3.1 Calibration techniques

Percentile or max calibration are the standard calibration techniques. Whereas in max calibration, the maximum absolute value in an activation tensor is used as clipping threshold α , in percentile calibration, the 99.99th or 99.999th percentile of the absolute values in an activation tensor is used as clipping threshold α . The activation values during inference are assumed to lie within $[-\alpha, \alpha]$. If they happen to lie outside this interval, they are clipped, i.e. an activation value a is set to $\hat{a} = \min(\alpha, \max(-\alpha, a))$. A scaling factor s is then chosen that corresponds to scaling $\pm\alpha$ to ± 127 (INT8). All calibration techniques need calibration samples (about 256 - 512 diverse samples). All ‘basic’ calibration techniques, that just select a good clipping threshold, in order of complexity:

1. Max calibration
2. Percentile calibration: 99.99%, 99.999%
3. Entropy calibration

3.1.1 Entropy calibration

Choose a scaling factor s so that the KL divergence⁴ between the distribution of an activation tensor and the distribution of the quantized values is as small as possible. In the NVIDIA paper, entropy calibration tends to perform well (about as well as a well-chosen percentile: 99.99% or 99.999%).

Practical implementation (TensorRT): first divide the original range $[0, \beta]$ of absolute activation values into 2048 histogram bins and let $\text{BIN}_0, \dots, \text{BIN}_{2047}$ count the values in each bin. Then choose a scaling factor that corresponds to a shorter range by iterating over $i \in \{128, \dots, 2048\}$, putting the “outlier” values of BIN_j into bin BIN_{i-1} , $j \geq i$. Call the resulting distribution P_i . Each time use an interpolation scheme to merge the bins $\text{BIN}_0, \dots, \text{BIN}_{i-1}$ (ignoring

³<https://arxiv.org/pdf/2303.05016>

⁴notation $KL(P||Q)$, a measure that stems from information theory

the outliers) to 128 bins and then expand back to i bins, this time preserving bins with no values in P_i . Call the resulting distribution Q_i . Then compare the resulting distributions with $KL(P_i||Q_i)$. The $i_{\min}$ that minimizes this is used to determine the clipping threshold α by $\alpha := (i_{\min} + 0.5) \cdot \text{BIN_WIDTH}$. Formula for the Kullback-Leibler divergence:

$$KL(P||Q) = \sum_j P(j) \log \left(\frac{P(j)}{Q(j)} \right).$$

Example of merging and expanding:
Let's start from 6 bins that contain

$$1, 2, 0, 3, 2, 1$$

samples. Merging them to three bins gives three bins that contain

$$1 + 2, 0 + 3, 2 + 1$$

samples. Expanding back to six bins, preserving bins with no samples, gives six bins with samples:

$$3/2, 3/2, 0, 3, 3/2, 3/2.$$

Merging 5 to 3 bins goes similarly, first expand to $5 \times 3 = 15$ bins, then merge into 3 bins. Or, directly,

$$s_1, \dots, s_5$$

is merged into

$$s_1 + \frac{2}{3}s_2, \frac{1}{3}s_2 + s_3 + \frac{1}{3}s_4, \frac{2}{3}s_4 + s_5.$$