Quantization for Inference

Precision Reduction Techniques for Efficient LLM Deployment

February 2026 • Technical Report

Table of Contents

1. Introduction

Quantization is the process of reducing the numerical precision of a model's weights, activations, and KV cache from high-precision floating-point formats (FP32 or FP16/BF16) to lower-bit representations such as INT8, INT4, FP8, or FP4. For large language models, quantization has become one of the most impactful optimization techniques, enabling models that would otherwise require multiple high-end GPUs to run on a single card, reducing inference latency by decreasing memory bandwidth requirements, and lowering serving costs by increasing throughput per dollar.

This report provides a comprehensive overview of quantization techniques as applied to LLM inference, covering the theoretical foundations, practical methods, calibration strategies, and the accuracy trade-offs involved at each precision level.

2. Why Quantization Matters for LLMs

2.1 The Memory Bandwidth Bottleneck

LLM inference, especially the autoregressive decode phase, is fundamentally memory-bandwidth bound. Each token generation step requires reading the full model weights from GPU memory while performing relatively little computation per byte loaded. Reducing weight precision from 16 bits to 8 bits halves the data that must be read, directly translating to higher effective memory bandwidth and faster token generation. At 4-bit precision, the improvement is 4x. This makes quantization one of the few optimizations that directly addresses the core bottleneck.

2.2 Memory Capacity and Accessibility

A 70-billion-parameter model in FP16 requires approximately 140 GB of memory, exceeding the capacity of any single consumer or workstation GPU. INT4 quantization reduces this to roughly 35 GB, fitting comfortably on a single 48 GB GPU. This democratization effect has been transformative for the open-source LLM community, enabling researchers and hobbyists to run frontier-class models on affordable hardware.

3. Quantization Fundamentals

3.1 Uniform and Non-Uniform Quantization

Uniform quantization maps floating-point values to a fixed grid of evenly spaced integer levels. The mapping is defined by a scale factor and a zero-point, computed from the range of the tensor being quantized. Non-uniform quantization uses unevenly spaced levels, often based on the statistical distribution of values, to better represent frequently occurring values with higher precision. While non-uniform schemes can achieve better accuracy, uniform quantization is far more hardware-friendly and dominates practical LLM deployment.

3.2 Weight-Only vs. Weight-and-Activation Quantization

Weight-only quantization quantizes only the model weights while keeping activations in their original precision. Since weights are static after model loading, they can be carefully quantized offline with sophisticated calibration. This approach is simple, has minimal impact on output quality, and is the most widely used method for LLM inference. **Weight-and-activation quantization** (W8A8, W4A8, etc.) additionally quantizes the dynamic activation tensors during inference. This enables the use of integer matrix multiplication hardware (such as INT8 tensor cores), providing additional speedups but requiring more careful calibration to handle the wider dynamic range and outlier values common in LLM activations.

3.3 Per-Tensor, Per-Channel, and Per-Group Quantization

The granularity at which quantization parameters are computed significantly affects accuracy. Per-tensor quantization uses a single scale and zero-point for an entire weight matrix, which is simple but can lose precision when value ranges vary significantly across channels. Per-channel quantization computes separate parameters for each output channel, better capturing the range of each channel. Per-group quantization divides each channel into groups (commonly 32 or 128 elements) with independent parameters, offering the finest granularity and best accuracy at the cost of slightly more complex dequantization logic.

4. Post-Training Quantization Methods

4.1 GPTQ

GPTQ is a one-shot weight quantization method based on approximate second-order information. It processes the weight matrix column by column, quantizing each column and then adjusting the remaining unquantized columns to compensate for the introduced error, using an approximate inverse Hessian computed from a small calibration dataset. GPTQ can quantize a large model in a matter of hours on a single GPU and produces INT4 or INT3 models with remarkably low perplexity degradation. It has become one of the standard quantization methods for the open-source community.

4.2 AWQ (Activation-Aware Weight Quantization)

AWQ observes that not all weights are equally important for model accuracy. It identifies salient weight channels by examining activation magnitudes from a calibration set, then applies per-channel scaling to protect important channels before quantization. By concentrating quantization error on less important channels, AWQ achieves better accuracy than naive round-to-nearest quantization at the same bit width, with lower computational cost than GPTQ.

4.3 SqueezeLLM and SpQR

These methods take a mixed-precision approach, identifying outlier weights that are particularly sensitive to quantization and storing them at higher precision (e.g., FP16) while quantizing the majority of weights to very low precision (3-bit or 2-bit). SqueezeLLM uses sensitivity-based non-uniform quantization with a lookup table, while SpQR isolates outlier weights using second-order sensitivity analysis. Both achieve competitive accuracy at extremely low average bit widths.

4.4 FP8 Quantization

FP8 (8-bit floating point) has emerged as a hardware-native quantization format supported by NVIDIA Hopper and Blackwell GPUs. FP8 comes in two variants: E4M3 (4 exponent bits, 3 mantissa bits) for weights and E5M2 (5 exponent bits, 2 mantissa bits) for activations and gradients. The floating-point format naturally handles the wide dynamic ranges of LLM tensors better than INT8 for many distributions, and hardware FP8 tensor cores deliver the same throughput as INT8. FP8 has become the preferred 8-bit format for inference on modern NVIDIA hardware.

5. Quantization-Aware Training and Fine-Tuning

Quantization-aware training (QAT) incorporates quantization simulation into the training or fine-tuning process, allowing the model to learn to compensate for quantization error during optimization. While QAT generally produces better accuracy than post-training quantization, it requires access to training data and significant computational resources. For LLMs, QAT is most commonly applied during fine-tuning rather than full pre-training. Methods like QLoRA combine parameter-efficient fine-tuning (LoRA) with 4-bit quantized base weights, enabling fine-tuning of large models on consumer hardware while maintaining quality that approaches full-precision fine-tuning.

6. Accuracy Trade-Offs Across Precision Levels

The following table summarizes typical accuracy and efficiency characteristics at different precision levels for LLM inference.

Precision	Memory Savings	Perplexity Impact	Hardware Support	Common Use
FP16/BF16	Baseline	None (reference)	Universal	Training, reference inference
FP8 (E4M3)	2x	Negligible	Hopper/Blackwell GPUs	Production serving
INT8 (W8A8)	2x	Negligible–Minor	Most modern GPUs	Production serving
INT4 (weight-	4x	Minor (1–3%	With dequant	Edge, cost-sensitive

only)		ppl increase)	kernels	serving
INT4 (W4A8)	~3x effective	Minor–Moderate	Emerging hardware	Experimental serving
INT3/INT2	5–8x	Moderate–Significant	Custom kernels	Research, extreme edge

7. Practical Deployment Considerations

Choosing a quantization strategy involves balancing multiple factors. For latency-sensitive API serving on high-end GPUs, FP8 quantization offers excellent throughput with essentially no quality degradation and native hardware support. For cost-optimized serving or edge deployment, INT4 weight-only quantization with GPTQ or AWQ provides a strong accuracy-to-efficiency trade-off. For resource-constrained environments like laptops or mobile devices, aggressive 3-bit or 2-bit quantization with mixed-precision outlier handling may be necessary despite more noticeable quality impacts.

Calibration dataset selection is an underappreciated factor in quantization quality. The calibration data should be representative of the model's deployment distribution. Using general-purpose web text to calibrate a model that will primarily process code, for example, can lead to suboptimal quantization decisions. Most methods require only 128 to 512 calibration samples, making it practical to curate domain-specific calibration sets.

Runtime kernel support is also critical. The theoretical memory savings of quantization only translate to actual speedups if the inference engine has optimized dequantization and matrix multiplication kernels for the chosen format. Libraries like Marlin (for GPTQ), CUTLASS (for FP8), and the bitsandbytes library (for NF4) provide these optimized kernels, and choosing a quantization format without corresponding kernel support can negate much of the potential benefit.

8. Future Directions

The field of LLM quantization continues to advance rapidly. Sub-4-bit quantization methods are improving through better outlier handling and learned codebook approaches. Hardware vendors are adding native support for lower-precision formats, with dedicated INT4 and even INT2 tensor cores on the roadmap. Dynamic quantization schemes that adapt precision based on the difficulty of the current generation step are an active research area. Additionally, the interplay between quantization and other optimizations such as pruning, distillation, and sparsity is being explored to achieve compounding efficiency gains.

9. Conclusion

Quantization has become an indispensable tool in the LLM inference toolkit, enabling dramatic reductions in memory footprint and latency with remarkably modest impacts on output quality. From the hardware-native FP8 format on modern GPUs to aggressive sub-4-bit methods for edge devices, the field offers a spectrum of options suited to different deployment constraints. As hardware support for low-precision formats expands and quantization algorithms continue to improve, the gap between quantized and full-precision model quality will continue to narrow, making efficient LLM inference accessible to an ever-broader set of applications and hardware platforms.