A Target-Centric Survey of Quantization-Aware Training

Anonymous ACL submission

Abstract

The rapid development of LLMs incurs prohibitive memory footprints and intensive computational demands. Quantization-Aware Training (QAT) techniques have emerged as a promising solution to address these challenges by explicitly *simulating quantization effects during model training*, yielding low-bit models that achieve accuracy comparable to their full-precision counterparts. In this work, we provide a target-centric survey of QAT, aimed at clarifying both its theoretical foundations and its evolving implementation landscape. We systematically review existing QAT methods based on a taxonomy organized by quantization targets. In-depth analysis of technical connections and distinctions among various QAT methods are presented, followed by a summary of QAT's evaluation paradigms. Lastly, we discuss the challenges in QAT evaluation, optimization, and deployment, and outline potential directions for future research.

1 Introduction

Recent years have witnessed rapid development progress of large-scale foundation models capable of communicating across different languages, understanding diverse data modalities, and generating multimedia assets. The rapidly growing scale of these models demands intensive memory and computational resources far exceeding the capacity of most current edge devices, sometimes even consumer-grade GPUs (Nagel et al., 2021; Wei et al., 2024; Yang et al., 2025; Liu et al., 2025a). To mitigate these resource bottlenecks, model quantization serves as a fundamental technique, compressing the size of these massive models by lowering the bit-width of their parameters (Jacob et al., 2018; Nagel et al., 2021; Wei et al., 2024).

Model quantization methods can be broadly divided into Post-training quantization (**PTQ**) and Quantization-aware training (**QAT**; Nagel et al. (2021); Wei et al. (2024); Yang et al. (2025)). PTQ applies low-precision quantization to a pre-trained model without further training to reduce model size and improve inference efficiency (Xiao et al., 2023; Liu et al., 2024b; Zhang and Shrivastava, 2024). However, directly quantizing a pre-trained model introduces significant performance degradations due to the limited representational range of low-bit weights. For instance, Bondarenko et al. (2021) identified that activations in residual connections contain structured outliers with high dynamic ranges, which are difficult to represent in low-bit formats. This has been repeatedly observed across numerous quantization methods; researchers are then cautious about directly applying PTQ.

In contrast, *QAT simulates quantization during training* using fake quantization modules, optimizing the model directly under quantized inference constraints, and often recovering the accuracy loss of PTQ, especially at ultra-low bit-widths such as INT4 (Jacob et al., 2018; Bengio et al., 2013). Recent QAT methods further combine quantization with parameter-efficient adaptation or distillation, making them increasingly applicable to large-scale LLMs (Xu et al., 2024; Chen et al., 2025).

There exist surveys covering QAT, however, they often discuss QAT as a component of the broader topic of neural network quantization, leaving the target-specific design principles of QAT underexplored. For instance, Wei et al. (2024) present a broad taxonomy of neural network quantization, but their discussion of QAT remains largely introductory and does not offer a detailed analysis of key challenges. Likewise, Nagel et al. (2021) focus on deployment-oriented industrial pipelines, with limited theoretical or analytical depth of QAT. More recently, Yang et al. (2025) organize quantization methods primarily by model architectures, but this coarse-grained categorization offers limited insight into module-specific QAT strategies.

In contrast, we primarily focus on QAT and or-

ganizes the literature accordingly. This perspective highlights how the numerical characteristics, lifetime, reuse frequency, and deployment constraints of different quantization targets shape the design of QAT schemes, and why targeting different modules in QAT results in diverse benefits and costs.

In this work, we firstly establish theoretical foundations of QAT, focusing on the mathematical formulation of uniform quantization, the fake quantization mechanism, and gradient approximation techniques such as the Straight-Through Estimator (§2). Next, as QAT is adaptable to diverse model architectures, we explore various quantization targets, emphasizing not only the common weights and activations, but also gradients and the emerging bottlenecks such as KV cache (§3). Thirdly, we detail the evaluation protocols of QAT via multiple perspectives, including task-level effectiveness, structural error diagnostics, reasoning-aware evaluation for LLMs, and deployment-grounded efficiency analysis. We also propose a minimal evaluation protocol, with detailed evaluation paradigms, metrics in Appendix C, and representative quantitative evidence provided in Appendix D (§4). Lastly, we discuss limitations of current QAT methods such as optimization instability and outline potential directions and opportunities for future research (§5).

2 Preliminary Knowledge

2.1 Uniform Affine Quantization (UAQ)

Quantization maps high-precision (e.g., FP32) real-valued tensors to a low-precision discrete set that is more amenable to efficient integer arithmetic and reduced memory usage. Different bit-widths b provide different trade-offs between efficiency and representational capacity. While FP32 offers high precision and a large dynamic range, lower-precision formats such as INT8 and INT4 significantly reduce memory footprint and computation cost, at the expense of potential information loss.

Let $x \in \mathbb{R}^{d_1 \times \cdots \times d_k}$ denote a real-valued tensor, e.g., a weight matrix. UAQ is parameterized by a positive **step size** $s > 0$, an integer **zero-point** $z \in \mathbb{Z}$, and an integer **representable range** $[q_{\min}, q_{\max}]$. For signed b-bit integers, one common choice is $q_{\min} = -2^{b-1}$ and $q_{\max} = 2^{b-1} - 1$. The quantization operator first maps x to an integer tensor $x_{\text{int}} \in \mathbb{Z}^{d_1 \times \cdots \times d_k}$:

$$x_{\text{int}} = \text{clamp}\left(\left\lfloor \frac{x}{s} \right\rceil + z,\ q_{\min},\ q_{\max} \right), \qquad (1)$$

where $\lfloor \cdot \rceil$ denotes deterministic rounding to the nearest integer, and $\text{clamp}(\cdot, q_{\min}, q_{\max})$ clips values to the integer range (Jacob et al., 2018).

During QAT, computation is usually still performed in floating point, while low-precision effects are emulated through a quantize–dequantize (i.e., fake-quantization) operator:

$$\hat{x} = s \cdot (x_{\text{int}} - z), \qquad (2)$$

where $\hat{x}$ is the **fake-quantized** proxy of x used in the forward pass. The quantization error can be written as $\epsilon = x - \hat{x}$.

UAQ is commonly instantiated in two forms. Symmetric quantization sets $z = 0$, while asymmetric quantization allows $z \neq 0$. We detail derivations of quantization parameters in Appendix A, and Appendix A.1 shows detailed derivations of (s, z) from a chosen clipping range.

Quantization Granularity. The scope over which (s, z) are shared defines the granularity. E.g., per-tensor quantization uses one pair for an entire tensor while per-channel quantization is more fine-grained, targeting specific dimensions of a weight tensor (Nagel et al., 2021).

2.2 QAT and Gradient Approximation

Quantization-aware training (QAT) *incorporates quantization effects during training* by inserting quantization operators into the training graph (Jacob et al., 2018; Nagel et al., 2021). This allows the model to effectively learn to adapt its parameters and intermediate representations to discretization effects encountered during low-precision inference.

Fake Quantization. In QAT, selected tensors are replaced by their fake-quantized counterparts during the forward pass. For a layer that computes $y = f(Wx + b)$, QAT instead computes

$$\begin{aligned} \hat{W} &= \mathcal{Q}(W), \qquad \hat{x} = \mathcal{Q}(x), \\ y &= f(\hat{W}\hat{x} + b), \end{aligned} \qquad (3)$$

where $\mathcal{Q}(\cdot)$ denotes the quantize–dequantize operator in Eqs. 1–2. In most QAT implementations, model parameters are not stored as integers during training. Instead, QAT maintains full-precision master weights W and uses their fake-quantized versions $\hat{W}$ only in the forward pass. This design preserves small gradient updates that would otherwise vanish after discretization, while exposing the model to quantization noise during training.

Straight-Through Estimator. Eq. 1 contains a non-differentiable rounding operator. The derivative of rounding is zero almost everywhere and

undefined at integer thresholds: $\frac{\partial \lfloor u \rfloor}{\partial u} = 0$, *a.e.* so naive backpropagation through the quantization path would produce vanishing or ill-defined gradients, preventing effective optimization of the underlying full-precision parameters (Nagel et al., 2022). QAT often relies on surrogate gradients; Straight-Through Estimator (STE) is the most widely used method, treating rounding as an identity operation in the backward pass (Bengio et al., 2013):

$$\frac{\partial \mathcal{L}}{\partial x} \approx \frac{\partial \mathcal{L}}{\partial \hat{x}} \cdot \mathbb{I}(x \in [x_{\min}, x_{\max}]) \,, \tag{4}$$

where $[x_{\min}, x_{\max}]$ is the clipping range and $\mathbb{I}(\cdot)$ is the indicator function. Intuitively, STE allows gradients to pass through the quantization operator as if $\hat{x} \approx x$ inside the valid range, while suppressing gradients for values that are saturated by clipping.

Though simple and efficient, STE is a coarse approximation. Later QAT methods therefore either improve the surrogate-gradient design, learn quantizer parameters, stabilize low-bit optimization, or replace hard rounding with differentiable relaxations (cf. Appendix B).

3 Taxonomy

Existing surveys often treat QAT as a component of general neural-network quantization or organize methods primarily by architecture (Nagel et al., 2021; Wei et al., 2024; Yang et al., 2025). In contrast, we organize QAT methods by quantization target, *because different targets induce different sources of error and require different mitigation strategies*: weights are affected by salient-weight loss and threshold oscillation (Nagel et al., 2022; Yuan et al., 2024); activations by range drift and outliers (Bondarenko et al., 2021; Xiao et al., 2023); KV cache by recurrent decoding error accumulation (Liu et al., 2024a; Hooper et al., 2024); and gradients by optimization-direction mismatch and convergence instability (Zhu et al., 2020; Xi et al., 2023). This target-centric taxonomy[1] is therefore more diagnostic than architecture-only or bit-width-only taxonomies.

We present an overview of our target-centric taxonomy of representative QAT frameworks in Figure 1. While many methods quantize multiple targets simultaneously, we group each work under the targets where its key ideas, algorithmic designs, and ablations are centered. Appendix D Table 8 shows the full method-level categorization.

[1]Throughout this survey, we use W*n*A*m* to denote n-bit weight and m-bit activation quantization (e.g., W4A4 refers to 4-bit weights and 4-bit activations).

3.1 Quantization Target: Model Weights

This section focuses on works applying QAT primarily to model weights.

Low-Bit Feasibility and Stability. Early weight-QAT methods established the feasibility of low-bit training but exposed trade-offs between compression and stability (Zhou et al., 2016). Binary quantization (Rastegari et al., 2016) offers maximal compression at the cost of larger accuracy loss, while ternary methods (Li et al., 2016) improve representational flexibility by adding a zero state. At very low bit-widths, the mismatch between continuous weight updates and discrete quantization levels can cause weights near thresholds to repeatedly flip between adjacent values, leading to oscillation and unstable convergence (Nagel et al., 2022). This problem is further amplified in low-bit ViTs by learnable scaling factors and coupled Query/Key weight quantization (Liu et al., 2023). Existing methods mitigate such instability through oscillation dampening and iterative weight freezing (Nagel et al., 2022), ViT-specific designs including statistical weight quantization, confidence-guided annealing, and Query-Key reparameterization (Liu et al., 2023), as well as stronger supervision or staged training for ternary and binary language models (Zhang et al., 2020; Bai et al., 2021). Weight QAT must also consider hardware feasibility, since low-precision deployment may suffer from accumulator overflow even when fake-quantized simulation is accurate. A2Q addresses this by incorporating accumulator-aware constraints (Colbert et al., 2023).

Selective Weight Quantization. In ultra-low-bit weight QAT, (e.g., below 3 bits), the central challenge is to preserve model capacity under a highly constrained parameter space. Different low-bit regimes require different quantization functions and training strategies, rather than a uniform treatment of all weights(Liu et al., 2025b). Preserving a small subset of salient weights in higher precision (Yuan et al., 2024), using Hessian-based group-wise quantization, and adopting mixed-precision allocation (Shen et al., 2019) can all reduce the degradation caused by uniform low-bit quantization. The other line of work improves scalability by restricting optimization to quantization-related parameters, auxiliary parameters, or low-rank subspaces. EfficientQAT (Chen et al., 2025) expands the op-

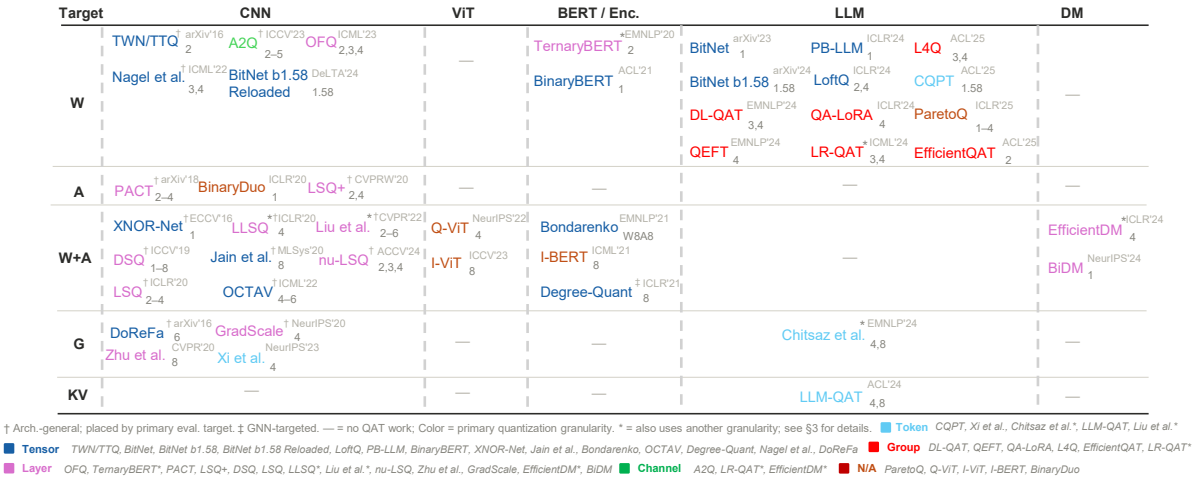

Figure 1: Taxonomy of QAT methods, organized by quantization target (rows) and model family (columns). Each entry shows the method name, bit-width as subscript, and publication venue as superscript. Method name color encodes primary quantization granularity: tensor-, layer-, channel-, row-, column-, group-, token-, or instance-level quantization. Methods with multiple granularities are colored by the one their ablations center on and marked with * in the legend. † denotes architecture-general methods placed by their primary evaluation target; ‡ denotes GNN-targeted; — indicates no identified QAT work in that cell. Appendix D Table 8 shows a table view.

timization space through staged training and then updates only quantization parameters, while DL-QAT (Ke et al., 2024) formulates QAT as low-rank adaptation in a decomposed quantization space. LoRA-style methods further reduce QAT cost by coupling quantization with low-rank adaptation: L4Q (Jeon et al., 2025) achieves LoRA-like training cost while producing a fully quantized final model, QA-LoRA (Xu et al., 2024) improves quantization flexibility through group-wise operators and supports lossless adapter merging, and LoftQ (Li et al., 2024) reduces quantization error through joint backbone quantization and low-rank initialization. QEFT (Lee et al., 2024) bridges these two directions by using sensitivity-aware mixed precision and hardware-friendly reordering to preserve important weight structures while improving the efficiency of quantized fine-tuning.

Pre-Training Integrated QAT. Recent work explores integrating QAT to pre-training: Low-bit weights are not only used to approximate a pre-trained full-precision model, but become part of the model's large-scale pre-training dynamics. Binary Transformer pre-training shows that discrete weight parameterization can be used from the beginning of training (Wang et al., 2023), while 1.58-bit ternary training provides a middle ground between binary compression and full-precision expressivity (Ma et al., 2024). The timing of quantization is also important: native 1.58-bit training remains possible at smaller scales but may need architectural or capacity adjustments (Nielsen and Schneider-Kamp, 2024), and continual quantization-aware pre-training treats the switch from high precision to 1.58-bit as a key design choice (Nielsen et al., 2025).

3.2 Quantization Target: Activations

Activation-centric QAT quantizes intermediates produced in the forward pass, e.g, hidden states, attention outputs, and residual streams. Activations are input-dependent and vary across data instances, layers, tokens and timesteps. Activation quantization is sensitive to distribution shift (Jacob et al., 2018; Liu et al., 2024a), and is central to integer-only inference and memory-efficient long-sequence processing. The Transformer architecture further challenges quantization by its structured activation outliers: a small number of tokens dominate the dynamic range, forcing most values into poorly resolved quantization intervals (Bondarenko et al., 2021; Xiao et al., 2023). Activation QAT for diffusion models is also challenging due to temporal non-stationarity, i.e., activation statistics vary along the denoising trajectory and errors accumulate along timesteps (Li et al., 2023b; He et al., 2024). At very low bit-widths, gradient mismatch between the training surrogate and the inference-time discrete operator becomes an additional bottleneck (Kim et al., 2020).

Existing methods address the challenges from two directions: (1) making the quantizer adaptive. Instead of fixed symmetric ranges (Eq. 1), learnable clipping thresholds, step sizes, offsets, or asymmetric ranges are optimized so that the quantization grid tracks shifting activation distributions (Choi et al., 2018; Esser et al., 2020; Bhalgat et al., 2020); (2) reducing the mismatch between smooth training surrogates and discrete inference operators via training–inference decoupling and smoother surrogate activations (Kim et al., 2020).

Effective activation QAT needs to properly handle outliers, non-stationary statistics, and training–inference mismatch. Transformer activation outliers, diffusion-model timestep shifts, and binary-network gradient mismatch each require target-specific solutions; no single quantizer design addresses all of them.

3.3 Quantization Target: Weight + Activation

Weights and activations (and other intermediates) can also be jointly quantized (**WA-QAT**) during training, making the simulated low-precision training graph closer to the inference graph during deployment (Jacob et al., 2018; Nagel et al., 2021). WA-QAT introduces coupled distortion: quantized weights reshape downstream activation distributions, activation clipping affects backward updates, and integer deployment imposes operator- and accumulator-aware constraints (Bondarenko et al., 2021; Colbert et al., 2023). Thus, WA-QAT is a co-design problem involving quantizer parameterization, graph-level integerization, and architecture-specific adaptation. We identify three primary challenges in WA-QAT and survey the corresponding solutions developed to address them.

Range Adaptation for Coupled Quantization Noise. In WA-QAT, upstream weight quantization continuously shifts downstream feature statistics, such that the activation ranges become unstable once weights and activations are discretized. Static calibration is therefore insufficient; models need to learn the range parameters adaptively. PACT learns activation clipping thresholds (Choi et al., 2018), while LSQ learns quantization step sizes through STE-style gradients (Esser et al., 2020). LSQ+ introduces learnable offsets for asymmetric activation distributions (Bhalgat et al., 2020). Recent work extends this to non-uniform grids for heavy-tailed or multi-modal distributions (Gongyo et al., 2024). On the other hand, hardware-aware methods restrict these ranges to symmetric or fixed-point thresholds to guarantee integer-only compatibility (Zhao et al., 2020; Jain et al., 2020; Sakr et al., 2022). Overall, adaptive range strategies must therefore balance the need to accurately model shifting distributions against the strict execution constraints of integer-only hardware.

Integer Transformers as Graph-Level Redesign. Quantizing only the linear projections in a Transformer is in-sufficient for integer-only deployment, because numerical errors and floating-point fallbacks can still arise from operations like Softmax, GELU, and LayerNorm (Kim et al., 2021; Li and Gu, 2023). Integer Transformer methods therefore reformulate these operators for integer-friendly arithmetic: I-BERT-style and I-ViT-style target integer-compatible nonlinear and normalization operations (Kim et al., 2021; Li and Gu, 2023). Q-ViT shows that attention and feed-forward activations are the primary degradation sources in fully quantized ViTs and introduces information rectification to mitigate this (Li et al., 2022). Together, these results show that effective Transformer WA-QAT should be understood as a computational-graph redesign, not just quantizing individual tensors.

Heterogeneity-Aware QAT Across Structures, Inputs, and Timesteps. Uniform WA-QAT assumes stable activation statistics, but this does not hold in several scenarios. In graph neural networks (GNNs), the degree of nodes can change aggregation statistics; degree-aware quantization avoids overly conservative global ranges (Tailor et al., 2021). For vision models, instance-conditioned bit allocation assigns lower precision to easier samples and higher precision to harder ones (Liu et al., 2022). In diffusion models, quantization error accumulates across denoising timesteps, motivating timestep-aware strategies to control temporal degradation (Zheng et al., 2024; He et al., 2024). Heterogeneity-aware methods address this by adapting precision or quantization behavior to the source of variation. Across these scenarios, adapting precision to the source of activation variability outperforms a single global configuration.

Overall, WA-QAT involves strong interaction between numerical approximation, optimization stability, and deployable integer execution. Existing solutions remain architecture-specific: methods stabilizing ViTs are not directly transferable to GNNs LLMs, or diffusion models.

3.4 Quantization Target: KV Cache

As context windows and batch sizes grow, the KV cache becomes a major memory bottleneck in autoregressive decoding. Cached keys and values grow with sequence length and are repeatedly reused by future attention steps, making KV-cache quantization complementary to weight and/or activation quantization (Kwon et al., 2023; Liu et al., 2024c; Hooper et al., 2024). For KV-cache QAT, errors in quantized KV states may accumulate across decoding steps as cached values are reused.

The direct relevant method involving KV cache is LLM-QAT (Liu et al., 2024a), which performs QAT by generating synthetic training data from a full-precision teacher and jointly simulating low-bit weights, activations, and KV cache during training. KV-cache QAT is still an emerging direction and the methodsremain limited. Promising directions may include: distribution-aware approaches exploiting the varying statistics of KV, RoPE, outliers (Liu et al., 2024c; Hooper et al., 2024; Liu et al., 2024b); adaptive and vector-quantized approaches compressing KV states via token-wise precision allocation or learned codebooks (Shutova et al., 2025; Boroujeni et al., 2026; Li et al., 2025); retrieval-oriented approaches preserving only attention-relevant tokens (He et al., 2025).

3.5 Quantization Target: Gradients

Gradient QAT[2] quantizes backward pass information such as activation gradients, layer-wise errors, and parameter gradients. Gradient QAT directly affects optimization directions, effective step sizes, and convergence stability.

Gradients can be quantized together with weights and activations, with stochastic quantization, careful scaling (Banner et al., 2018), or partial high-precision computation (Wu et al., 2018) to ensure convergence. Others treat gradient QAT as an optimization-alignment problem. Unified INT8 training (Zhu et al., 2020) introduces direction-sensitive clipping. Distribution-adaptive INT8 training (Zhao et al., 2021) exploits channel-wise gradient statistics and magnitude-aware clipping to reduce range mismatch. Uniform low-bit gradients are often unstable for LLMs. Recent work proposes selective mechanisms like bit splitting, sampling (Xi et al., 2023), per-block scaling (Xi et al., 2024), stochastic rounding, or specialized low-bit formats (Chmiel et al., 2025) to stabilize training.

Overall, gradient QAT is an optimization-stability problem requiring range control, direction preservation, and structure-aware selectivity. Future work may borrow from adjacent fields like low-rank gradient training (Zhao et al., 2024), stochastic rounding, or low-bit floating-point formats.

4 Evaluation

QAT evaluation is also challenging. First, the same nominal bit-width could produce diverse quantized computation graphs. E.g., a W4 evaluated with fake quantization is not directly comparable to a W4 model executed with integer kernels, quantized activations, KV-cache compression, and integer nonlinear operators. Next, task accuracy or perplexity alone could lead to omissions about activation outliers, gradient mismatch, weight oscillation, accumulator overflow, reasoning degradation, or latency regressions. Lastly, reported compression does not necessarily translate into deployment efficiency. Real-world speedups depend on kernel availability, compiler support, and whether nonlinear operations fall back to the floating point (Jacob et al., 2018; Nagel et al., 2021; Colbert et al., 2023).

We organize QAT evaluation around three dimensions: QAT performance, structural diagnostics, and deployment evidence. Detailed protocols and quantitative results are in Appendix C.

4.1 Evaluation Setup and Comparability

A QAT result is interpretable only when its quantization setting is fully specified. We identify five reporting dimensions: (1) quantization target (weights, activations, gradients, KV cache, optimizer states, and nonlinear operators); (2) quantization granularity (e.g., per-tensor, per-channel, etc); (3) numerical format (symmetric/asymmetric ranges, uniform/non-uniform grids, rounding and clipping rules, accumulator bit-width); (4) training stage (from-scratch, post-pretraining, fine-tuning, Q-PEFT, continual quantization-aware pretraining); (5) hardware evidence (fake quantization, simulated integer arithmetic, real integer kernels, measured latency/throughput).

QAT methods differ in factors like training budgets, quantization targets, and deployment backends. We synthesize evidence within comparable models and derive generalizable evaluation lessons than constructing a unified leaderboard. Evaluation

[2] In this work, we focus on methods that quantize gradients within the computation graph, rather than gradients synchronized in the distributed training system.

priorities also differ by target: weight QAT suffers from salient-weight loss, activation QAT from outliers and range drift, KV-cache QAT from accumulated attention errors, gradient QAT from optimization instability, low-bit pre-training from representation drift, and diffusion QAT from timestep-wise error accumulation. Task scores alone are insufficient; they should be paired with structural diagnostics and deployment evidence.

4.2 Empirical Findings

Finding 1: INT8 QAT is largely mature, but deployment evidence still matters. For CNNs and Transformer encoders, INT8 QAT is near-lossless: Unified INT8 training reports only a 0.26 Top-1 drop on ResNet-50 (Zhu et al., 2020), and I-BERT achieves essentially lossless W8A8 inference for RoBERTa (Kim et al., 2021). Similar trends appear in I-ViT, where W8A8 DeiT-S slightly exceeds the full-precision baseline (Li and Gu, 2023). The remaining challenge is deployment fidelity: whether the fake-quantized training graph matches the real integer execution graph (Jacob et al., 2018; Kim et al., 2021; Li and Gu, 2023; Colbert et al., 2023).

Finding 2: W4 QAT is practical, but failure modes are architecture-dependent. On residual CNNs, W4A4 QAT can match or slightly exceed full precision shown by LLSQ and LSQ-style methods (Zhao et al., 2020; Esser et al., 2020; Bhalgat et al., 2020). Group-wise W4 methods like LR-QAT, EfficientQAT, LoftQ, and L4Q slightly affect LLM perplexity (Bondarenko et al., 2024; Chen et al., 2025; Li et al., 2024; Jeon et al., 2025). However, failure modes vary: compact or depthwise architectures like MobileNetV2 are sensitive to activation clipping (Sakr et al., 2022), ViTs degrade through attention and nonlinear operators (Li et al., 2022; Kim et al., 2021; Li and Gu, 2023), and for LLMs, weight-only W4 is far easier than joint WA-QAT or KV-cache QAT (Liu et al., 2024a).

Finding 3: W2 and binary QAT are qualitatively different from W4. Degradations at sub-3-bit are sharp across model families: Q-ViT sees -7.80 Top-1 on DeiT-S at W2A2 (Li et al., 2022), EfficientQAT W2g64 reports -4.72 zero-shot ability and +1.39 perplexity, on LLaMA-2-7B (Chen et al., 2025). These results reflect a new regime dominated by STE-induced gradient mismatch, weight oscillation, salient-weight loss, and error accumulation (Yin et al., 2019; Nagel et al., 2022; Liu et al., 2023; Zheng et al., 2024).

Finding 4: Perplexity is insufficient for LLM QAT. Perplexity measures local next-token fidelity but does not capture long-horizon reasoning. W4 methods may report small perplexity changes while reasoning benchmarks like GSM8K degrade substantially (Yuan et al., 2024), and this gap widens at W2/NF2 (Li et al., 2024). Conversely, Q-PEFT methods may inflate MMLU or commonsense scores via adaptation gains unrelated to quantization quality (Xu et al., 2024; Ke et al., 2024; Lee et al., 2024; Jeon et al., 2025).

Finding 5: Compression does not imply acceleration. Real-world speedup depends on kernel availability, compiler support, and whether nonlinear operators fall back to floating point. Integer-only methods like I-ViT report 3.72–4.11× latency reduction by reformulating nonlinear operators for integer execution, while partial integer pipelines yield smaller gains (Kim et al., 2021; Li and Gu, 2023). QAT evaluations should therefore report measured latency or throughput, not only nominal compression.

4.3 Structural, Reasoning, and Deployment Diagnostics

Beyond task scores, structural diagnostics are needed to locate where quantization error enters and how it propagates. Basic structural diagnostic metrics include normalized MSE, cosine similarity, clipping ratio, saturation ratio, and layer-wise error profiles (Choi et al., 2018; Esser et al., 2020; Bhalgat et al., 2020; Jain et al., 2020; Zhao et al., 2020). For Transformers, these should be paired with block-output drift and KL divergence between full-precision and quantized logits, since local perturbations can be amplified by attention, normalization and other nonlinear operators (Hinton et al., 2015; Kim et al., 2021; Liu et al., 2024a).

For LLM QAT, reasoning should be evaluated separately from perplexity. Quantization noise can accumulate across layers and decoding steps, leading to reasoning degradation while perplexity changes are small. A minimal protocol should cover mathematical reasoning (GSM8K/MATH; Cobbe et al. (2021); Hendrycks et al. (2020)), multi-step knowledge reasoning (BBH/MMLU; Suzgun et al. (2023); Hendrycks et al. (2021)) and code generation (HumanEval/MBPP; Chen et al. (2021); Austin et al. (2021)). Fixed prompting and decoding configurations should be reported to ensure comparability.

Efficiency evaluation should distinguish model footprint, training cost, and realized inference

speed. Model footprint may include quantized weights, scales, zero-points, preserved full-precision tensors, adapters, and KV-cache storage when applicable (Li et al., 2024; Chen et al., 2025). Training and inference efficiency should be reported separately, as QAT often maintains full-precision master weights, optimizer states and other information during training. For LLMs, prefill latency and decoding throughput should also be distinguished (Liu et al., 2024a)

4.4 Minimal Evaluation Protocol

QAT evaluation should identify where quantization works, why quantization fails, and whether compression translates into deployable efficiency. We propose a minimal evaluation protocol for future QAT-related work: Firstly, authors should specify the full quantization configuration (§4.1) and evaluate at least two bit-widths with matched full-precision baselines and absolute deltas. Secondly, results should be accompanied by at least one structural diagnostic such as layer-wise distortion, gradient mismatch or overflow rate (Nagel et al., 2022), and at least one training- or deployment-efficiency measurement (Bondarenko et al., 2024). Lastly, for LLMs, perplexity alone is insufficient; at least one reasoning-intensive benchmark should be included (Li et al., 2024; Yuan et al., 2024).

This protocol improves reproducibility while capturing the central lesson of existing evidence: QAT performance depends on the interaction among bit-width, quantization target, architecture, and deployment backend.

5 Challenges and Future Directions

Optimization Instability. STE-based surrogate gradients do not reflect the true derivatives of the quantized objective, introducing gradient bias and mismatch (Sakr et al., 2022), degrading optimization stability and convergence. Near quantization thresholds, small weight updates cause abrupt value changes, leading to oscillation (Liu et al., 2023). In ultra-low-bit regimes, coarse quantization can trap weights within a single quantization bin (Lee et al., 2021), preventing effective optimization. Better alignment between backward gradients and the discrete objective remains an open problem.

Activation Outliers. Strong input-dependent outliers in attention modules and residual streams can skew quantization ranges and degrade accuracy, especially for Transformers and LLMs (Bondarenko et al., 2021). Under aggressive joint WA-QAT, this noise accumulates across layers and interacts unfavorably with LayerNorm and Softmax. Current methods like knowledge distillation and rotation-based transformations add significant overhead (Zhang et al., 2020; Liu et al., 2024b). Future work may explore quantization-native architectures or training objectives suppressing outliers, reducing reliance on post-hoc corrections.

Simulation-to-Deployment Gap. Simulated fake quantization in FP32 does not capture low-level hardware constraints such as accumulator overflow, rounding modes, and operator fusion (Jacob et al., 2018; Colbert et al., 2023). Models converged under QAT simulation may degrade or fail to execute efficiently on integer-only accelerators (Kim et al., 2021; Li and Gu, 2023). Future work should aim to close this gap by incorporating hardware-aware constraints during QAT training.

High Training Overhead. QAT requires full-precision master weights, optimizer states, and often distillation teachers, making training memory and compute costs prohibitive for billion-parameter models (Liu et al., 2024a; Chen et al., 2025). This creates an inherent tension between compression objectives and the cost of the compression process itself (Dremov et al., 2025; Huang et al., 2024). Reducing QAT overhead without sacrificing quantization quality could be a promising direction.

Homogenization. Uniform quantizers, fixed bit-width allocation, and layer-agnostic design choice can constrain expressive capacity, causing architecturally diverse models to converge to similar weight distributions and activation patterns after quantization (Nagel et al., 2021). This effect intensifies with increased model-scale, suppressing fine-grained features and limiting downstream adaptability to various tasks (Yu et al., 2024). Balancing compression efficiency and representational diversity remains an open research challenge.

6 Conclusion

We review quantization-aware training methods via a target-centric taxonomy covering weights, activations, KV cache, and gradients. We also review QAT evaluations and propose a minimal evaluation protocol for future research. We identify persistent challenges like optimization instability, activation outliers, simulation-to-deployment gap, high training overhead, model homogenization, and outline directions for future research.

Limitations

This paper focuses primarily on quantization-aware training, emphasizing quantization methods, evaluations, outlining challenges and future directions. Some deployment-oriented systems and hardware-specific optimization techniques may be underrepresented. Since existing studies differ in setup, the quantitative evidence in this survey should be interpreted as general trends rather than strict head-to-head comparisons. Although we cover representative works, newly released methods, benchmarks, and deployment backends may not be fully reflected in this survey. We present supplementary mathematical details of quantization parameters in Appendix A, surrogate-gradient variants in Appendix B, quantitative evaluations in Appendix C.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online. Association for Computational Linguistics.

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. 2018. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. 2020. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. 2024. Low-rank quantization-aware training for llms. *arXiv preprint arXiv:2406.06385*.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. 2021. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7947–7969, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sayed Pedram Haeri Boroujeni, Niloufar Mehrabi, Patrick Woods, Gabriel Hillesheim, and Abolfazl Razi. 2026. Don't waste bits! adaptive kv-cache quantization for lightweight on-device llms. *arXiv preprint arXiv:2604.04722*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. 2025. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. 2025. Fp4 all the way: Fully quantized training of large language models. In *Advances in Neural Information Processing Systems*, volume 38, pages 91130–91151. Curran Associates, Inc.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Ian Colbert, Alessandro Pappalardo, and Jakoba Petri-Koenig. 2023. A2q: Accumulator-aware quantization with guaranteed overflow avoidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16989–16998.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Aleksandr Dremov, David Grangier, Angelos Katharopoulos, and Awni Hannun. 2025. Compute-optimal quantization-aware training. *arXiv preprint arXiv:2509.22935*.

Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. 2020. Learned step size quantization. In

8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks . In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4851–4860, Los Alamitos, CA, USA. IEEE Computer Society.

Shinya Gongyo, Jinrong Liang, Mitsuru Ambai, Rei Kawakami, and Ikuro Sato. 2024. Learning non-uniform step sizes for neural network quantization. In *Proceedings of the Asian Conference on Computer Vision*, pages 4385–4402.

Junhui He, Junna Xing, Nan Wang, Rui Xu, Shangyu Wu, Peng Zhou, Qiang Liu, Chun Jason Xue, and Qingan Li. 2025. A2ats: Retrieval-based kv cache reduction via windowed rotary position embedding and query-aware vector quantization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12451–12463.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2024. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: towards 10 million context length llm inference with kv cache quantization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. 2024. Robust and efficient quantization-aware training via coreset selection. *Transactions on Machine Learning Research*.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. *Advances in neural information processing systems*, 29.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Sambhav Jain, Albert Gural, Michael Wu, and Chris Dick. 2020. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *Proceedings of Machine Learning and Systems*, 2:112–128.

Hyesung Jeon, Yulhwa Kim, and Jae-Joon Kim. 2025. L4q: parameter efficient quantization-aware fine-tuning on large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2002–2024.

Wenjing Ke, Zhe Li, Dong Li, Lu Tian, and Emad Barsoum. 2024. Dl-qat: Weight-decomposed low-rank quantization-aware training for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 113–119.

Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. 2020. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. *ArXiv*, abs/2002.06517.

Minsoo Kim, Sihwa Lee, Suk-Jin Hong, Du-Seong Chang, and Jungwook Choi. 2022. Understanding and improving knowledge distillation for quantization aware training of large transformer encoders. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6713–6725, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Changhun Lee, Jun-gyu Jin, YoungHyun Cho, and Eunhyeok Park. 2024. QEFT: Quantization for efficient fine-tuning of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13823–13837, Miami, Florida, USA. Association for Computational Linguistics.

Junghyup Lee, Dohyung Kim, and Bumsub Ham. 2021. Network quantization with element-wise gradient scaling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6448–6457.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. 2016. Ternary weight networks. *arXiv preprint arXiv:1605.04711*.

Junyan Li, Yang Zhang, Muhammad Yusuf Hassan, Talha Chafekar, Tianle Cai, Zhile Ren, Pengsheng Guo, Foroozan Karimzadeh, Chong Wang, and Chuang Gan. 2025. Commvq: Commutative vector quantization for kv cache compression. *arXiv preprint arXiv:2506.18879*.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. 2023a. Q-diffusion: Quantizing diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17535–17545.

Yanjing Li, Sheng Xu, Xianbin Cao, Xiao Sun, and Baochang Zhang. 2023b. Q-dm: an efficient low-bit quantized diffusion model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. 2022. Q-vit: Accurate and fully quantized low-bit vision transformer. *Advances in neural information processing systems*, 35:34451–34463.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2024. Loftq: Lora-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhikai Li and Qingyi Gu. 2023. I-vit: Integer-only quantization for efficient vision transformer inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17065–17075.

Kai Liu, Qian Zheng, Kaiwen Tao, Zhiteng Li, Haotong Qin, Wenbo Li, Yong Guo, Xianglong Liu, Linghe Kong, Guihai Chen, Yulun Zhang, and Xiaokang Yang. 2025a. Low bit model quantization for deep neural networks: A survey. *arXiv preprint arXiv:2505.05530*.

Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. 2023. Oscillation-free quantization for low-bit vision transformers. In *International conference on machine learning*, pages 21813–21824. PMLR.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2024a. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484.

Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. 2020. Reactnet: Towards precise binary neural network with generalized activation functions. In *European conference on computer vision*, pages 143–159. Springer.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024b. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. 2025b. Paretoq: Improving scaling laws in extremely low-bit llm quantization.

Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. 2022. Instance-aware dynamic neural network quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12434–12443.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024c. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *ArXiv*, abs/2106.08295.

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. 2022. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pages 16318–16330. PMLR.

Jacob Nielsen and Peter Schneider-Kamp. 2024. Bitnet b1. 58 reloaded: State-of-the-art performance also on smaller networks. In *International Conference*

on Deep Learning Theory and Applications, pages 301–315. Springer.

Jacob Nielsen, Peter Schneider-Kamp, and Lukas Galke. 2025. Continual quantization-aware pre-training: When to transition from 16-bit to 1.58-bit pre-training for bitnet language models? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13483–13493.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer.

Charbel Sakr, Steve Dai, Rangharajan Venkatesan, Brian Zimmer, William J. Dally, and Brucek Khailany. 2022. Optimal clipping and magnitude-aware differentiation for improved quantization-aware training. In *International Conference on Machine Learning*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2019. Q-bert: Hessian based ultra low precision quantization of bert. In *AAAI Conference on Artificial Intelligence*.

Alina Shutova, Vladimir Malinovskii, Vage Egiazarian, Denis Kuznedelev, Denis Mazur, Surkov Nikita, Ivan Ermakov, and Dan Alistarh. 2025. Cache me if you must: Adaptive key-value quantization for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55451–55473. PMLR.

Xiao Sun, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi Viji Srinivasan, and Kailash Gopalakrishnan. 2020. Ultra-low precision 4-bit training of deep neural networks. *Advances in Neural Information Processing Systems*, 33:1796–1807.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Shyam A Tailor, Javier Fernandez-Marques, and Nicholas D Lane. 2021. Degree-quant: Quantization-aware training for graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Lu Wei, Zhong Ma, Chaojie Yang, and Qin Yao. 2024. Advances in the neural network quantization: A comprehensive review. *Applied Sciences*, 14(17).

Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. 2018. Training and inference with integers in deep neural networks. *arXiv preprint arXiv:1802.04680*.

Haocheng Xi, Yuxiang Chen, Kang Zhao, KAI JUN TEH, Jianfei Chen, and Jun Zhu. 2024. Jetfire: efficient and accurate transformer pretraining with int8 data flow and per-block quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Haocheng Xi, ChangHao Li, Jianfei Chen, and Jun Zhu. 2023. Training transformers with 4-bit integers. In *Advances in Neural Information Processing Systems*, volume 36, pages 49146–49168. Curran Associates, Inc.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2024. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiawei Yang, Zhongbo Li, Zeqin Feng, and Yongqiang Xie. 2025. A survey on neural network quantization. In *Proceedings of the 2025 6th International Conference on Computer Information and Big Data Applications*, CIBDA '25, page 384–394, New York, NY, USA. Association for Computing Machinery.

Penghang Yin, J. Lyu, Shuai Zhang, S. Osher, Yingyong Qi, and Jack Xin. 2019. Understanding straight-through estimator in training activation quantized neural nets. *ArXiv*, abs/1903.05662.

Jeffrey Yu, Kartik Prabhu, Yonatan Urman, Robert M. Radway, Eric Han, and Priyanka Raina. 2024. 8-bit transformer inference and fine-tuning for edge accelerators. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. 2024. PB-LLM: partially binarized large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Tianyi Zhang and Anshumali Shrivastava. 2024. Leanquant: Accurate and scalable large language model quantization with loss-error-aware grid. *arXiv preprint arXiv:2407.10032*.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. TernaryBERT: Distillation-aware ultra-low bit BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 509–521, Online. Association for Computational Linguistics.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: memory-efficient llm training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Kang Zhao, Sida Huang, Pan Pan, Yinghan Li, Yingya Zhang, Zhenyu Gu, and Yinghui Xu. 2021. Distribution adaptive int8 quantization for training cnns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3483–3491.

Xiandong Zhao, Ying Wang, Xuyi Cai, Cheng Liu, and Lei Zhang. 2020. Linear symmetric quantization of neural networks for low-precision integer hardware. In *International Conference on Learning Representations*.

Xingyu Zheng, Xianglong Liu, Yichen Bian, Xudong Ma, Yulun Zhang, Jiakai Wang, Jinyang Guo, and Haotong Qin. 2024. Bidm: Pushing the limit of quantization for diffusion models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. 2020. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1969–1979.

A Mathematical Details of Quantization Parameters

This appendix provides the mathematical details for deriving uniform quantization parameters and for computing surrogate gradients of learnable step sizes. The main text introduces these concepts at a high level in Section 2; here we provide the corresponding derivations.

A.1 Deriving Uniform Quantization Parameters

A common choice in uniform affine quantization is to derive the step size s and zero-point z from a chosen clipping range $[x_{\min}, x_{\max}]$. For asymmetric quantization, the real-valued interval $[x_{\min}, x_{\max}]$ is mapped to the integer interval $[q_{\min}, q_{\max}]$. The step size is given by

$$s = \frac{x_{\max} - x_{\min}}{q_{\max} - q_{\min}}. \tag{5}$$

The zero-point is chosen so that the real value zero is represented as accurately as possible in the integer domain:

$$z = \left\lfloor q_{\min} - \frac{x_{\min}}{s} \right\rceil. \tag{6}$$

In practice, z is further clipped to the valid integer range $[q_{\min}, q_{\max}]$ if necessary. The clipping range $[x_{\min}, x_{\max}]$ can be estimated from calibration statistics, optimized by minimizing local reconstruction error, or learned jointly with model parameters.

For symmetric quantization, the zero-point is typically fixed to $z = 0$, and the clipping range is chosen to be symmetric around zero. Let

$$\alpha = \max(|x_{\min}|, |x_{\max}|). \tag{7}$$

A common symmetric step size is

$$s = \frac{\alpha}{q_{\max}}. \tag{8}$$

The quantize–dequantize operation then simplifies to

$$\hat{x} = s \cdot \mathrm{clamp}\left(\left\lfloor \frac{x}{s} \right\rceil,\ q_{\min},\ q_{\max}\right). \tag{9}$$

Symmetric quantization is commonly used for weight tensors because it simplifies integer arithmetic, while asymmetric quantization is often useful for activation tensors with shifted or non-negative distributions.

A.2 Gradient Derivation for Learnable Step Size

Methods such as LSQ treat the quantization step size s as a learnable parameter (Esser et al., 2020). Consider the symmetric quantization case with $z = 0$. The fake-quantized output can be written as

$$\hat{x} = s \cdot \mathrm{clamp}\left(\left\lfloor \frac{x}{s} \right\rceil,\ q_{\min},\ q_{\max}\right). \tag{10}$$

Let $v = x/s$. Under the STE approximation, the derivative of $\hat{x}$ with respect to the step size s is approximated piecewise as

$$\frac{\partial \hat{x}}{\partial s} = \begin{cases} q_{\min}, & v \leqslant q_{\min}, \\ \lfloor v \rceil - v, & q_{\min} < v < q_{\max}, \\ q_{\max}, & v \geqslant q_{\max}. \end{cases} \tag{11}$$

Equivalently, substituting $v = x/s$ gives

$$\frac{\partial \hat{x}}{\partial s} = \begin{cases} q_{\min}, & \frac{x}{s} \leqslant q_{\min}, \\ \left\lfloor \frac{x}{s} \right\rceil - \frac{x}{s}, & q_{\min} < \frac{x}{s} < q_{\max}, \\ q_{\max}, & \frac{x}{s} \geqslant q_{\max}. \end{cases} \tag{12}$$

This gradient allows the quantization grid to be adjusted according to the task loss. Intuitively, the learnable step size controls the trade-off between clipping-induced distortion and rounding-induced error. LSQ further applies gradient scaling to stabilize the magnitude of step-size updates relative to weight updates (Esser et al., 2020).

B Surrogate-Gradient Variants and Stabilization Techniques

Section 2 introduces STE as the basic mechanism that enables backpropagation through non-differentiable quantization operators. This appendix provides a more detailed discussion of common strategies for improving or bypassing vanilla STE, as well as complementary stabilization techniques used in QAT.

B.1 Improving and Bypassing STE

Surrogate-gradient design is a core component of QAT because it determines whether the quantized objective can be optimized stably. Existing methods improve or bypass vanilla STE along several directions, including clipped gradients, learnable quantization parameters, gradient scaling, oscillation-aware optimization, generalized activation functions, and differentiable soft quantization.

Clipped and masked STE. A basic refinement is to make the backward approximation more consistent with the clipping behavior in the forward pass. The clipped STE in Eq. 4 masks gradients outside the valid quantization range, reducing the mismatch between forward saturation and backward propagation. This form is widely used in practical QAT pipelines (Jacob et al., 2018; Esser et al., 2020). However, it also introduces dead zones: once weights or activations are saturated, their gradients may vanish, making it difficult for them to re-enter the quantization range. This motivates methods that learn or adapt clipping ranges during training.

Learnable quantization parameters. Another important direction extends surrogate gradients from quantized tensors to quantizer parameters themselves. Instead of using fixed scales, offsets, or clipping thresholds, these methods optimize the quantization grid jointly with model parameters. PACT learns activation clipping thresholds to balance clipping-induced distortion and rounding-induced error (Choi et al., 2018). LSQ treats the quantization step size as a learnable parameter and updates it through an STE-style surrogate gradient with proper gradient scaling (Esser et al., 2020). LSQ+ further introduces learnable offsets to better fit asymmetric activation distributions (Bhalgat et al., 2020). Similarly, TQT optimizes quantization thresholds for fixed-point inference (Jain et al., 2020). These methods improve STE-based QAT not by changing the rounding operator itself, but by making the quantization range and resolution task-adaptive. The detailed piecewise gradient derivation for learnable step sizes in LSQ-style methods is provided in Appendix A.2.

Gradient scaling and magnitude-aware differentiation. Since STE can distort not only the existence of gradients but also their scale, several methods modify the surrogate update to better preserve optimization dynamics. LSQ introduces gradient

scaling for the learnable step size so that its update magnitude is comparable to that of model weights (Esser et al., 2020). OCTAV further studies optimal clipping and magnitude-aware differentiation, showing that better gradient treatment for quantization parameters can improve low-bit QAT stability (Sakr et al., 2022). In low-precision training, related techniques such as GradScale use dynamic scaling and overflow feedback to stabilize quantized gradients (Sun et al., 2020). These methods suggest that effective STE design should account for gradient magnitude, saturation behavior, and the interaction between quantized forward computation and backward updates.

Oscillation-aware optimization. A further limitation of vanilla STE is threshold oscillation. Because the forward quantizer is piecewise constant, small full-precision updates around a quantization threshold can cause the quantized value to flip repeatedly across training steps. Such oscillation can prevent convergence and is especially harmful in low-bit regimes. Oscillation-aware QAT methods explicitly analyze this phenomenon and modify the update or quantization procedure to reduce unstable threshold crossings (Nagel et al., 2022; Liu et al., 2023). These methods do not necessarily replace STE, but they make STE-based optimization more stable by accounting for the discrete structure of the quantized parameter space.

Generalized activation functions. In ultra-low-bit regimes, especially binarization, the extreme nonlinearity of the sign function makes STE gradients highly unreliable (Hubara et al., 2016). Instead of only modifying the backward pass, some methods change the forward parameterization so that the network becomes easier to optimize under STE. ReActNet introduces learnable activation components such as RSign and RPReLU to reshape activation distributions and improve the compatibility between binary forward computation and surrogate-gradient training (Liu et al., 2020). Such generalized activation functions reduce gradient mismatch by increasing the expressive flexibility of binarized networks, thereby narrowing the performance gap with full-precision models.

Differentiable soft quantization. A more radical strategy is to replace hard rounding with differentiable relaxations during training. Differentiable Soft Quantization (DSQ) formulates quantization as a smooth parametric function and uses an annealing process to gradually approach the discrete target grid (Gong et al., 2019). This provides well-defined gradients during most of training and alleviates the non-differentiability of rounding. However, soft quantization introduces additional hyperparameters and may create a relaxation-to-deployment gap if the smooth training objective does not perfectly match the final hard quantizer. This trade-off is particularly relevant for noise-sensitive models such as diffusion models, where smooth optimization can help prevent error accumulation across denoising steps (Li et al., 2023a).

Practical trade-offs. Overall, STE improvement in QAT can be understood as a spectrum. Clipped or masked STE is simple and efficient, making it suitable for moderate bit-widths such as INT8 or INT4. Learnable scales, offsets, and clipping thresholds improve adaptability to non-stationary tensor distributions, especially for activation quantization. Gradient scaling, magnitude-aware differentiation, and oscillation-aware optimization improve stability in more aggressive low-bit regimes. Generalized activation functions are particularly important for binarized networks, where the sign function makes vanilla STE unreliable. Differentiable soft quantization reduces reliance on hard STE by providing smoother gradients, but it increases algorithmic complexity and may introduce a mismatch between relaxed training and discrete inference. Therefore, the choice of STE variant or alternative should depend on bit-width, quantization target, model architecture, and deployment constraints.

B.2 Additional Stabilization Techniques

Beyond surrogate-gradient design, QAT often incorporates additional stabilization strategies. Knowledge distillation can provide smoother supervision from a full-precision teacher, which is especially useful when aggressive quantization causes large prediction shifts (Hinton et al., 2015; Kim et al., 2022). Regularization terms can also be used to constrain quantized representations or reduce sensitivity to rounding noise. These techniques are complementary to STE improvements: surrogate gradients make optimization possible, while distillation and regularization help stabilize the training trajectory under severe low-precision constraints.

C Detailed Evaluation Protocols

This appendix provides detailed benchmark protocols and extended diagnostic recommendations. These details support the synthesis in Section 4, but are kept in the appendix because protocols are comparable mainly within model families rather than across the entire QAT literature.

C.1 Benchmarks and Metrics

Table 1 summarizes the common evaluation protocols used in QAT studies. The protocol maturity varies substantially across model families. CNN and BERT-style encoder quantization have relatively standardized benchmarks, while ViT, diffusion, and LLM QAT require more diagnostic evaluation because quantization errors are often localized in activation outliers (Bondarenko et al., 2021), attention maps (Li et al., 2022), nonlinear operators (Kim et al., 2021; Li and Gu, 2023), KV cache (Liu et al., 2024a; Bondarenko et al., 2024), or denoising timesteps (He et al., 2024; Zheng et al., 2024).

Tables 2–7 provide a more detailed view of benchmark and evaluation-dimension coverage across representative QAT methods. Specifically, Table 2 summarizes the coverage of language modeling, commonsense, knowledge, reasoning, and code benchmarks for LLM QAT methods; Tables 3, 4, and 5 report evaluation coverage for CNNs, ViTs, and BERT-style encoders; and Tables 6 and 7 summarize the corresponding coverage for diffusion models and GNNs. For these tables, ✓indicates that the benchmark or evaluation dimension is reported in the main evaluation setting of the corresponding method, △ indicates auxiliary, partial, simulated, theoretical, or otherwise less directly comparable evidence, and "–" indicates not reported. Coverage rate counts both ✓and △ as coverage, while △ flags comparability limitations. These coverage rates should be interpreted as descriptive summaries rather than leaderboard-style comparisons, because papers differ in backbones, training budgets, benchmark choices, kernels, accelerators, deployment assumptions, and reporting protocols. These tables show that QAT evaluation remains uneven across model families: mature settings such as CNNs and encoder models have relatively standardized benchmarks, whereas LLMs, diffusion models, and GNNs still exhibit substantial gaps in reasoning, deployment, temporal, or structure-specific evaluation.

C.2 Extended Structural Evaluation

Structural evaluation is needed to identify where quantization error enters and how it propagates. Tensor distortion can be measured with normalized mean squared error:

$$\mathrm{nMSE}(X) = \frac{\|X - Q(X)\|_F^2}{\|X\|_F^2}.$$

This metric is often complemented by cosine similarity, clipping ratio, saturation ratio, and layer-wise error profiles (Choi et al., 2018; Esser et al., 2020; Bhalgat et al., 2020; Jain et al., 2020; Zhao et al., 2020). Reporting per-layer curves is preferable to a single aggregate number because quantization error is usually concentrated in a few sensitive modules (Bondarenko et al., 2021; Li et al., 2022; Chen et al., 2025).

For Transformers, tensor-level distortion should be paired with block-output drift and logit divergence, since small perturbations can be amplified by attention, residual connections, normalization, and nonlinear functions (Bondarenko et al., 2021; Li et al., 2022; Kim et al., 2021; Li and Gu, 2023). Useful diagnostics include block-output MSE, hidden-state cosine similarity, KL divergence between full-precision and quantized logits, and the change in cross-entropy or perplexity on the same evaluation batches (Hinton et al., 2015; Kim et al., 2022; Liu et al., 2024a). These metrics connect local quantization noise to functional degradation.

Training stability is a QAT-specific structural criterion. Since rounding is non-differentiable, most QAT methods rely on STE-style surrogate gradients (Bengio et al., 2013; Jacob et al., 2018; Esser et al., 2020). We therefore recommend reporting gradient cosine similarity, sign agreement, gradient-norm drift, and the fraction of weights whose quantized values repeatedly flip across adjacent updates (Zhu et al., 2020; Sun et al., 2020; Nagel et al., 2022; Liu et al., 2023). These diagnostics are especially important below 3 bits, where gradient mismatch and threshold oscillation often dominate final performance (Yin et al., 2019; Nagel et al., 2022; Liu et al., 2023).

Hardware feasibility should also be evaluated structurally. For integer matrix multiplication, the accumulator range depends on both operand ranges and the reduction dimension (Jacob et al., 2018; Colbert et al., 2023). A conservative bound is

$$\left|\sum_{i=1}^{n} q_i^w q_i^a\right| \leqslant n \cdot q_{\max}^w \cdot q_{\max}^a.$$

Family	Representative works	Benchmarks	Metrics	Bits	Evaluation purpose
CNNs	DoReFa-Net, PACT, LSQ, LLSQ, A2Q, Unified INT8	ImageNet, CIFAR	Top-1/Top-5, overflow, latency	W8A8, W4A4, W2A2, W1A1	Evaluate low-bit convolution, activation clipping, and integer-training feasibility.
ViT	Q-ViT, I-ViT, OFQ, INT4 training	ImageNet / ImageNet-1K	Top-1, attention distortion, speedup	W8A8, W4A4, W2A2	Measure sensitivity of attention, Softmax, GELU, and LayerNorm.
BERT	TernaryBERT, BinaryBERT, I-BERT	GLUE, SQuAD v1.1/v2.0	Accuracy, F1, EM, compression	W8A8, W2, W1	Test whether encoder representations survive ultra-low-bit compression.
LLMs	LLM-QAT, LR-QAT, QA-LoRA, LoftQ, EfficientQAT, PB-LLM	WikiText-2, C4, MMLU, ARC, GSM8K, HumanEval	PPL, zero-shot accuracy, reasoning, pass@1	W4, W3, W2, NF4, NF2	Assess language modeling, knowledge, instruction following, and reasoning.
Native low-bit LMs	BitNet, BitNet b1.58, CQPT, ParetoQ	Pretraining corpora, zero-shot suites	Loss, PPL, downstream accuracy	W1, W1.58, W2–W4	Study whether quantization should be integrated into pretraining.
Diffusion models	EfficientDM, BiDM	ImageNet, LSUN, COCO-style generation	FID, IS, CLIP score, sampling cost	W4A8, W4A4, W2A8, W1A8	Evaluate error accumulation along denoising trajectories.
GNNs	Degree-Quant	Citation and inductive graph datasets	Accuracy, generalization to unseen graphs, CPU speedup	INT8, INT4	Evaluate degree-dependent quantization error and robustness to graph-structure heterogeneity.

Table 1: Common evaluation protocols for QAT. Benchmarks include ImageNet (Deng et al., 2009), GLUE (Wang et al., 2018), SQuAD (Rajpurkar et al., 2016), WikiText-2 (Merity et al., 2016), (Raffel et al., 2020), MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and FID (Heusel et al., 2017).

Thus, the accumulator bit-width may be much larger than the weight or activation bit-width. Hardware-aware QAT should report analytical safety margins and empirical overflow or saturation rates under representative inputs, as exemplified by accumulator-aware QAT (Colbert et al., 2023).

C.3 Extended Reasoning Evaluation

For LLM QAT, reasoning should be evaluated separately from language modeling. Perplexity is useful for measuring average next-token fidelity, but reasoning requires maintaining intermediate states across long generated sequences, as emphasized by chain-of-thought prompting studies (Wei et al., 2022). Quantization noise in weights, activations, or KV cache can accumulate across layers and decoding steps (Liu et al., 2024a; Bondarenko et al., 2024), making reasoning degradation visible even when perplexity changes are small.

A minimal reasoning protocol should include both answer-only and rationale-generating prompts. GSM8K or MATH-style benchmarks measure mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2020); BBH or MMLU-style benchmarks test multi-step knowledge reasoning (Suzgun et al., 2023; Hendrycks et al., 2021); and HumanEval or MBPP-style benchmarks evaluate code generation (Chen et al., 2021; Austin et al., 2021). The prompt template, number of shots, decoding temperature, maximum generation length, answer-extraction rule, and invalid-output handling should be fixed and reported. We recommend reporting final-answer accuracy, invalid-output rate, and generation length.

When possible, LLM QAT papers should compare W-only, W+A, W+A+KV, and KV-only settings under the same prompting and decoding configuration (Liu et al., 2024a; Bondarenko et al.,

Method	Backbone	Lang. Modeling		Commonsense		Higher-order Reasoning		
		WT2	C4	Zero-shot	MMLU	GSM8K	MATH/BBH	HumanEval
Base model compression (W-only / W+A, no fine-tuning)								
LLM-QAT	LLaMA-1 7&13&30B	✓	✓	✓	✓	–	–	–
LR-QAT	LLaMA-1/2/3, Mistral	✓	–	✓	–	–	–	–
EfficientQAT	LLaMA-2/3 7&13&30B	✓	✓	✓	△	–	–	–
DL-QAT	LLaMA-1/2 7&13B	✓	–	✓	△	–	–	–
PB-LLM	LLaMA-1 7B, OPT 1.3B	✓	✓	✓	–	–	–	–
ParetoQ	LLaMA-3, MobileLLM	✓	–	✓	–	–	–	–
Q-PEFT (quantization-aware parameter-efficient fine-tuning)								
QA-LoRA	LLaMA-1/2 7&13B	–	–	✓	✓	–	–	–
LoftQ	LLaMA-2 7&13B	✓	–	–	–	✓	–	–
L4Q	LLaMA, OpenLLaMA, Mistral	–	–	✓	✓	–	–	–
QEFT	LLaMA-2 7&13&70B	△	–	✓	✓	✓	–	–
Native low-bit pre-training								
BitNet b1.58	BitNet-LM (⩾2B)	✓	✓	✓	–	–	–	–
CQPT	BitNet-LM	–	–	✓	–	–	–	–
Coverage rate	**(out of 12)**	**9/12**	**4/12**	**11/12**	**6/12**	**2/12**	**0/12**	**0/12**

Table 2: Benchmark coverage of representative LLM QAT methods. Zero-shot task sets differ across papers, so aggregate averages are not strictly comparable. LoftQ evaluates GSM8K in a fine-tuning / natural-language-generation setting. BitNet b1.58 results refer to the 2B4T technical report.

Method	Backbone	Image Classification		Architecture Coverage		Efficiency / Deployment Evidence		
		ImageNet	CIFAR/SVHN	ResNet/AlexNet	Compact CNNs	Low-bit Grad.	Latency/Speedup	Hardware-aware
Early low-bit CNN QAT								
DoReFa-Net	AlexNet / CNN	✓	✓	✓	–	✓	△	△
PACT	AlexNet / ResNet	✓	✓	✓	–	–	△	△
DSQ	ResNet / general CNN	✓	✓	✓	✓	–	✓	✓
LSQ	ResNet	✓	–	✓	△	–	–	–
LLSQ	ResNet / MobileNetV2	✓	✓	✓	✓	–	△	△
Learnable thresholds, scaling, and integer-oriented CNN QAT								
TQT	VGG/ Inception/ ResNet/ MobileNet/ DarkNet	✓	–	✓	✓	–	△	△
LSQ+	EfficientNet / MixNet	✓	–	–	✓	–	–	–
Unified INT8	ResNet / MobileNetV2	✓	✓	✓	✓	✓	✓	✓
GradScale	ResNet	✓	✓	✓	✓	✓	△	△
OCTAV	ResNet / MobileNet / BERT	✓	–	✓	✓	△	–	△
A2Q	General CNN	✓	–	✓	✓	–	–	✓
Instance-aware DQ	ResNet / MobileNetV2	✓	✓	✓	✓	–	△	△
Coverage rate	**(out of 12)**	**12/12**	**5/12**	**11/12**	**10/12**	**4/12**	**7/12**	**10/12**

Table 3: Benchmark coverage of representative CNN QAT methods. Compact CNNs refer to MobileNet-, EfficientNet-, MixNet-, or similarly resource-constrained convolutional backbones. Low-bit Grad. denotes explicit low-bit gradient or low-bit training-flow evaluation. Latency/Speedup and Hardware-aware evidence are not strictly comparable because papers differ in accelerators, kernels, fixed-point assumptions, and whether real deployment is measured.

2024; Chen et al., 2025). This isolates whether reasoning loss is caused mainly by weight discretization, activation outliers, KV-cache accumulation, or their interaction. Such target-specific evaluation is particularly important for long-context generation, where KV-cache errors may remain small at each step but accumulate over the whole decoding trajectory (Liu et al., 2024a; Bondarenko et al., 2024).

C.4 Extended Efficiency and Deployment Evaluation

Efficiency evaluation should distinguish model compression, training feasibility, and realized inference speed (Nagel et al., 2021; Jacob et al., 2018). Model footprint should include quantized weights,

Method	Backbone	Image Classification		ViT Architecture Coverage		Transformer-specific Diagnostics / Deployment		
		ImageNet-1K	Top-1 Δ	DeiT	Swin/ViT-B	Attention Dist.	Integer-only Ops	Latency/Speedup
Fully quantized low-bit ViTs								
Q-ViT	DeiT / Swin	✓	✓	✓	✓	✓	–	△
OFQ	DeiT / Swin	✓	✓	✓	✓	✓	–	–
Integer-only and low-bit Transformer training								
I-ViT	DeiT / Swin / ViT-B	✓	✓	✓	✓	✓	✓	✓
Xi et al.	ViT-L/16 / DeiT-S	✓	✓	✓	✓	✓	△	△
Coverage rate	**(out of 4)**	**4/4**	**4/4**	**4/4**	**4/4**	**4/4**	**2/4**	**3/4**

Table 4: Benchmark coverage of representative Vision Transformer QAT methods. Attention Dist. denotes attention-map, self-attention, query-key, or block-level distortion analysis. Integer-only Ops denotes whether nonlinear Transformer operators such as Softmax, GELU, and LayerNorm are made integer-compatible. Latency/Speedup evidence is not directly comparable because Q-ViT reports theoretical acceleration, I-ViT reports deployment-oriented speedup, and low-bit training works may focus on training data flow rather than inference kernels.

Method	Backbone	NLU Benchmarks		Compression Setting		Deployment / Analysis		
		GLUE	SQuAD	Ultra-low W/E	W+A / INT8	Model Size	Latency/Speedup	Distill./Diagnostics
Ultra-low-bit BERT compression								
TernaryBERT	BERT-base	✓	✓	✓	✓	✓	–	✓
BinaryBERT	BERT-base	✓	✓	✓	✓	✓	–	✓
Integer-only and activation-aware Transformer encoder quantization								
Bondarenko et al.	BERT / Transformer Enc.	✓	–	✓	✓	✓	–	✓
I-BERT	RoBERTa-base / large	✓	–	–	✓	△	✓	✓
OCTAV	BERT-base / large	–	✓	△	✓	–	–	✓
Coverage rate	**(out of 5)**	**4/5**	**3/5**	**4/5**	**5/5**	**4/5**	**1/5**	**5/5**

Table 5: Benchmark coverage of representative BERT-family and Transformer-encoder QAT methods. Ultra-low W/E denotes binary or ternary weight and embedding quantization. W+A / INT8 denotes joint weight–activation quantization or integer-only INT8 inference. Distill./Diagnostics covers knowledge distillation, layer-wise analysis, activation-outlier analysis, or quantization-error diagnostics. Latency/Speedup remains a coverage gap because most BERT-family QAT studies emphasize accuracy and compression rather than end-to-end measured deployment.

Method	Backbone	Generation Benchmarks			Generation Metrics		Temporal / Efficiency Evidence	
		CIFAR-10	LSUN	ImageNet	FID/sFID	IS/CLIP	Timestep Analysis	Sampling Cost
Low-bit diffusion-model QAT / QAT-like fine-tuning								
Q-DM	DDPM / DDIM	✓	–	✓	✓	✓	✓	✓
EfficientDM	DDIM / LDM	✓	✓	✓	✓	✓	✓	✓
Extreme binarization for diffusion models								
BiDM	DDIM / LDM	✓	✓	–	✓	✓	✓	✓
Coverage rate	**(out of 3)**	**3/3**	**2/3**	**2/3**	**3/3**	**3/3**	**3/3**	**3/3**

Table 6: Benchmark coverage of representative diffusion-model QAT methods. FID/sFID is the dominant evaluation family for quantized diffusion models, while IS/CLIP-style metrics are much less consistently reported in these representative QAT works. Timestep Analysis denotes timestep-aware quantization, timestep-wise learned step sizes, cross-timestep structure, or explicit discussion of quantization error accumulation along denoising trajectories. Sampling Cost denotes reported denoising-step, quantization-speed, storage, OPs, or efficiency evidence.

scales, zero-points, preserved full-precision tensors, adapters, and KV-cache storage when applicable. This is especially important for group-wise QAT and Q-PEFT, where scale tensors or adapter pa-

Method	Backbone	Graph Benchmarks		Quantization Setting		Graph-specific Evaluation	
		Citation Graphs	Inductive / Unseen Graphs	INT8	INT4	Degree Analysis	Unseen Graph Generalization
GNN-targeted quantization-aware training							
Degree-Quant	GCN / GAT / GIN	✓	✓	✓	△	✓	✓
Coverage rate	**(out of 1)**	**1/1**	**1/1**	**1/1**	△	**1/1**	**1/1**

Table 7: Benchmark coverage of representative GNN QAT methods. Degree-Quant is the primary GNN-targeted QAT method in this taxonomy. Citation Graphs denotes transductive citation-network evaluation, while Inductive / Unseen Graphs denotes graph-classification, graph-regression, or unseen-graph evaluation. INT4 is marked with △ because it is reported only for GIN on REDDIT-BINARY, rather than across all evaluated backbones and datasets. Degree Analysis captures the central motivation of Degree-Quant: quantization error depends on graph degree and aggregation statistics. The coverage rate is shown out of one representative method, rather than used to imply that GNN QAT is a mature or densely covered area.

rameters may occupy non-negligible memory (Hu et al., 2022; Xu et al., 2024; Li et al., 2024; Bondarenko et al., 2024; Chen et al., 2025). A reported compression ratio should therefore reflect actual storage rather than nominal weight bit-width alone.

Training efficiency should be reported independently of inference efficiency. QAT often maintains full-precision master weights, optimizer states, activation checkpoints, and sometimes full-precision teacher models for distillation (Jacob et al., 2018; Nagel et al., 2021; Hinton et al., 2015; Liu et al., 2024a). Hence, practical feasibility depends on peak memory, wall-clock time, number of steps or tokens, batch size, sequence length, optimizer precision, gradient checkpointing, and hardware type (Liu et al., 2024a; Yuan et al., 2024; Bondarenko et al., 2024; Chen et al., 2025). For LLM QAT, whether a method fits on a single commodity GPU can be more informative than its nominal FLOP count (Bondarenko et al., 2024; Chen et al., 2025; Dremov et al., 2025).

Inference efficiency should be measured on the intended deployment path rather than inferred from bit-width (Jacob et al., 2018; Kim et al., 2021; Li and Gu, 2023). For autoregressive LLMs, prefill latency and decoding throughput should be reported separately, since weight quantization and KV-cache quantization affect these stages differently (Liu et al., 2024a; Bondarenko et al., 2024). Reports should specify prompt length, generated length, context length, batch size, KV-cache precision, kernel backend, compiler stack, and whether any operators fall back to floating point. For vision and encoder models, latency should specify input resolution, batch size, and whether Softmax, GELU, and LayerNorm are executed in integer arithmetic (Kim et al., 2021; Li and Gu, 2023).

We distinguish four deployment levels. Level 1 is fake quantization, where low-bit effects are simulated in floating point (Jacob et al., 2018; Nagel et al., 2021). Level 2 is simulated integer arithmetic, where integer ranges and rounding are modeled but kernels are not deployed (Colbert et al., 2023). Level 3 is partial integer deployment, where matrix multiplications use integer kernels but nonlinear functions or normalization remain in floating point (Kim et al., 2021; Li and Gu, 2023). Level 4 is end-to-end integer deployment, where linear layers, nonlinear functions, normalization, and accumulators are compatible with the target integer backend (Kim et al., 2021; Li and Gu, 2023). QAT papers should state which level their evaluation supports.

D Supplementary Tables

The supplementary tables provide a compact reference for the methods and quantitative results discussed in the main text. They complement the target-centric taxonomy in Section 3 and the evaluation discussion in Section 4, rather than serving as a unified leaderboard. Since QAT methods differ in quantization targets, numerical formats, training stages, evaluation protocols, and deployment assumptions, the reported results should be interpreted primarily within the same model family or experimental setting.

Unless otherwise stated, W, A, G, KV, E, and Opt denote weights, activations, gradients, key-value cache, embeddings, and optimizer states, respectively. For example, W4A8KV4 indicates 4-

bit weights, 8-bit activations, and 4-bit KV cache. The notation g64/g128 denotes group size, and NF2/NF4 denotes 2-/4-bit NormalFloat formats. Gran. denotes quantization granularity, such as tensor-, layer-, channel-, row-, column-, group-, token-, or instance-level quantization.

For all quantitative result tables, values in parentheses denote the difference between the quantized model and the corresponding full-precision baseline reported in the original paper. The baseline is typically FP32 for CNN, ViT, BERT-family, and diffusion-model results, and FP16/BF16 for LLM results. Therefore, these deltas should be interpreted as within-paper comparisons rather than cross-paper normalized scores. "–" indicates that the corresponding value or baseline difference is not reported in the original paper.

The remaining tables summarize representative task-level results by model family. We report the metrics most commonly used in each area, including ImageNet Top-1 accuracy for CNNs and Vision Transformers, MNLI/SQuAD results for BERT-family encoders, perplexity and zero-shot accuracy for base LLMs, MMLU and commonsense averages for instruction-tuned LLMs, and FID for diffusion models. These results are not strictly comparable across all papers, because training budgets, quantization targets, benchmark choices, prompting or decoding settings, and deployment backends may differ. Therefore, the tables are intended to show representative evidence rather than to rank methods globally.

For LLM tables, Avg. zero-shot and Avg. commonsense scores are not strictly comparable across papers. Different works may average over different task sets, use different prompt templates, numbers of shots, decoding settings, or scoring metrics such as `acc` and `acc_norm`. In addition, some instruction-tuned or Q-PEFT results may reflect both quantization robustness and adaptation gains. We therefore report these averages as paper-specific summary statistics rather than directly comparable leaderboard scores.

For diffusion models, FID is a lower-is-better metric. The reported ΔFID values are computed relative to the corresponding full-precision baseline from the same paper. Because sampling schedules and denoising steps may differ across papers, diffusion-model results should be interpreted within each original evaluation setting.

Approach	Venue	Target	Bits	Gran.	Model
TWN/TTQ	2016 arXiv	W	W2	Tensor	CNN / General-CV
TernaryBERT	2020 EMNLP	W	W2E2A8	Layer, Row	BERT
BinaryBERT	2021 ACL	W	W1E1A8, W1E1A4	Layer, Tensor	BERT
Nagel et al.	2022 ICML	W(+A)	W3A3, W4A4	Tensor	CNN / General
BitNet	2023 arXiv	W	W1A8	Tensor	LLM
A2Q	2023 ICCV	W	W2, W3, W4, W5	Channel	General / CNN
OFQ	2023 ICML	W(+A)	W2A2, W3A3, W4A4	Row	ViT
BitNet b1.58 (Ma et al.)	2024 arXiv	W	W1.58A8	Tensor	LLM
BitNet b1.58 Reloaded	2024 DeLTA	W	W1.58	Tensor	Small LM / Vision
DL-QAT	2024 EMNLP Industry	W	W3, W4	Group	LLM
QEFT	2024 EMNLP Findings	W	W3, W4	Group	LLM
PB-LLM	2024 ICLR	W	partial W1 + salient FP	Column / Salient weights	LLM
QA-LoRA	2024 ICLR	W	W2, W3, W4	Group	LLM
LoftQ	2024 ICLR	W	W2, W4; NF2, NF4	Group / Block	LLM
LR-QAT	2024 arXiv	W	W3, W4	Channel, Group	LLM
EfficientQAT	2025 ACL	W	W2g64, W2g128, W3g128, W4g128	Group	LLM
L4Q	2025 ACL	W	W3, W4	Group	LLM
CQPT	2025 ACL Findings	W	FP16 → W1.58	Layer, Token	BitNet LM
ParetoQ	2025 NeurIPS	W	W1, W1.58, W2, W3, W4	/	LLM
PACT	2018 arXiv	A	W2A2, W3A3, W4A4	Layer	CNN / General-CV
LSQ+	2020 CVPRW	A	W2A2, W3A3, W4A4	Layer	Efficient CNNs
BinaryDuo	2020 ICLR	A	W1A1; coupled A1 proxy	/	BNN
XNOR-Net	2016 ECCV	W + A	BWN: W1; XNOR: W1A1	Tensor	CNN / BNN
DSQ	2019 ICCV	W + A	W1A1, W2A2, W3A3, W4A4	Layer	CNN / General-CV
LSQ	2020 ICLR	W + A	W2A2, W3A3, W4A4	Layer	CNN / General-CNN
LLSQ	2020 ICLR	W + A	W3A3, W4A4	Channel, Layer	CNN
Jain et al.	2020 MLSys	W + A	W8A8, W4A4	Tensor	CNN
Bondarenko et al.	2021 EMNLP	W + A	W8A8; low-bit W/E ablations	Tensor / Embedding-group	BERT / Transformer encoder
I-BERT	2021 ICML	W + A	W8A8	/	BERT / RoBERTa
Degree-Quant	2021 ICLR	W + A	INT8, INT4	Tensor	GNN
Liu et al.	2022 CVPR	W + A	dynamic W/A 2–6; W4A4	Layer, Instance	CNN / Image classification
OCTAV	2022 ICML	W + A	W4A4, W5A5, W6A6	Tensor / Vector	CNN + BERT
Q-ViT	2022 NeurIPS	W + A	W2A2, W3A3, W4A4	/	ViT
I-ViT	2023 ICCV	W + A	W8A8	/	ViT
nu-LSQ	2024 ACCV	W + A	W2A2, W3A3, W4A4	Layer / Non-uniform steps	CNN / ViT
EfficientDM	2024 ICLR Spotlight	W + A	W4A8, W4A4	Channel, Layer	Diffusion Model
BiDM	2024 NeurIPS	W + A	W1A1; W1A8, W2A8	Layer	Diffusion Model
DoReFa-Net	2016 arXiv	W + A + G	W1A2G6	Tensor	CNN
Zhu et al.	2020 CVPR	W + A + G	W8A8G8	Layer	CNN
GradScale	2020 NeurIPS	W + A + G	W4A4G4	Layer	General
Xi et al.	2023 NeurIPS	W + A + G	W4A4G4 / INT4 training	Token / Structure-aware	Transformer
Chitsaz et al.	2024 EMNLP Findings	W + A + G + Opt	4, 8	Tensor, Channel, Token	Transformer LM
LLM-QAT	2024 ACL Findings	W + A + KV	W4A8KV4, W4A8KV8	Token	LLM

Table 8: Taxonomy of quantization-aware training (QAT) frameworks.

Method	Model	Bits	Top-1 (%) (Δ)
Zhu et al.	ResNet-18	W8A8G8	69.67 (−0.63)
Zhu et al.	ResNet-50	W8A8G8	76.34 (−0.26)
Zhu et al.	MobileNetV2	W8A8G8	71.20 (−1.19)
LLSQ	ResNet-18	W4A4	69.84 (+0.08)
LLSQ	ResNet-18	W3A3	68.08 (−1.68)
LLSQ	MobileNetV2	W4A4	67.37 (−4.43)
TQT	MobileNet-V2	W8A8	71.80 (−0.10)
GradScale	ResNet-18	W4A4G4	69.60 (−0.70)
GradScale	ResNet-50	W4A4G4	75.78 (−0.62)
Nagel et al.	ResNet-18	W4A4	68.27 (−1.13)
Nagel et al.	ResNet-50	W4A4	74.01 (−2.47)
Nagel et al.	AlexNet	W4A4	56.38 (−1.18)
OCTAV	ResNet-50	W4A4	76.21 (+0.14)
OCTAV	ResNet-18	W4A4	69.90 (−0.22)
OCTAV	MobileNet-V2	W4A4	71.23 (−0.48)
LSQ+	ResNet-18	W2A2	66.80 (−3.30)
LSQ+	ResNet-18	W3A3	69.30 (−0.80)
LSQ+	ResNet-18	W4A4	70.80 (+0.70)
nu-LSQ-A	ResNet-18	W2A2	64.89 (−4.87)
PACT	ResNet-18	W2A2	64.40 (−6.00)
PACT	ResNet-18	W4A4	69.20 (−1.20)
DSQ	ResNet-18	W2A2	65.17 (−4.73)
DSQ	ResNet-18	W4A4	69.56 (−0.34)
LSQ	ResNet-18	W2A2	67.60 (−2.90)
LSQ	ResNet-18	W4A4	71.10 (+0.60)
DoReFa-Net	AlexNet	W1A2G6	46.10 (−9.80)
DoReFa-Net	AlexNet	W1A4G6	48.20 (−7.70)
Liu et al. 2022	ResNet-50	∼4MP/∼4MP	76.94 (−0.02)
Liu et al. 2022	MobileNetV2	∼4MP/∼4MP	72.05 (–)

Table 9: ImageNet Top-1 Accuracy (%) for Representative CNN Quantization-Aware Training Methods. For Liu et al. 2022, ∼4MP/∼4MP denotes the approximately 4-bit mixed/dynamic precision setting.

Method	Backbone	Bits	Top-1 %
Q-ViT	DeiT-S	W4A4	80.90 (+1.00)
Q-ViT	DeiT-S	W2A2	72.10 (−7.80)
Q-ViT	Swin-T	W4A4	82.50 (+1.30)
Q-ViT	Swin-T	W2A2	74.70 (−6.50)
I-ViT	DeiT-S	W8A8	80.12 (+0.27)
I-ViT	Swin-T	W8A8	81.50 (+0.15)
I-ViT	ViT-B	W8A8	84.76 (+0.23)
OFQ	DeiT-S	W2A2	75.72 (−4.18)
OFQ	Swin-T	W2A2	78.52 (−2.68)
OFQ	DeiT-S	W4A4	81.10 (+1.20)
OFQ	Swin-T	W4A4	81.88 (+0.68)
Xi et al.	ViT-L/16	W4A4	82.61 (−1.94)
Xi et al.	DeiT-S	W4A4	69.18 (−3.92)

Table 10: ImageNet Top-1 Accuracy (%) — Vision Transformer Architectures. I-ViT INT8 results are written as W8A8 for consistency, and Xi et al. report 4-bit fully quantized training rather than standard QAT-based inference quantization.

Method	Backbone	Bits	MNLI-m (%)	SQuAD F1 (%)
TernaryBERT	BERT-base	W2E2A8	83.3(−1.2)	87.4(−1.3)
BinaryBERT	BERT-base	W1E1A8	84.2(−0.4)	88.3(−0.2)
BinaryBERT	BERT-base	W1E1A4	83.9(−0.7)	87.2(−1.3)
I-BERT	RoBERTa-Base	W8A8	87.5(−0.3)	–
I-BERT	RoBERTa-Large	W8A8	90.4(+0.4)	–
OCTAV Dynamic	BERT-Base	W4	–	84.51(−3.73)
OCTAV Dynamic	BERT-Base	W8	–	88.34(+0.10)
OCTAV Dynamic	BERT-Large	W4	–	87.09(−3.91)
OCTAV Dynamic	BERT-Large	W8	–	90.78(−0.22)

Table 11: GLUE MNLI-m and SQuAD v1.1 Results for BERT-family Models. I-BERT reports INT8 integer-only inference results. Bits of OCTAV denotes the uniform quantization bit-width used in its BERT fine-tuning experiments.

Method	Base Model	Bits	PPL ↓	Avg. zero-shot
LLM-QAT	LLaMA-1-7B	W4A8KV8	11.2(+0.8)	64.8(−1.4)
LLM-QAT	LLaMA-1-7B	W4A8KV4	11.6(+1.2)	63.0(−3.2)
LLM-QAT	LLaMA-1-13B	W4A8KV8	10.0(+0.3)	66.8(−1.2)
LLM-QAT	LLaMA-1-13B	W4A8KV4	10.2(+0.5)	63.4(−4.6)
LR-QAT	LLaMA-1-7B	W4 g128	5.75(+0.07)	69.15(−0.53)
LR-QAT	LLaMA-2-7B	W4 g128	5.59(+0.12)	69.88(−0.59)
LR-QAT	LLaMA-2-13B	W4 g128	4.97(+0.09)	72.91(−0.27)
EfficientQAT	LLaMA-2-7B	W3g128	5.81(+0.34)	64.02(−0.84)
EfficientQAT	LLaMA-2-7B	W2g64	6.86(+1.39)	60.14(−4.72)
EfficientQAT	LLaMA-2-13B	W3g128	5.12(+0.24)	67.28(−0.53)
EfficientQAT	LLaMA-2-13B	W2g64	5.96(+1.08)	64.48(−3.33)
DL-QAT	LLaMA-1-7B	W4A8KV8	6.7(+1.02)	69.4(−1.1)
DL-QAT	LLaMA-2-7B	W4A16KV16	6.3(+0.83)	69.3(−1.2)
PB-LLM	LLaMA-1-7B	W1+30%W8	–	66.9(−1.8)
PB-LLM	LLaMA-1-7B	W1+10%W8	–	60.6(−8.1)
QA-LoRA	LLaMA-1-7B	W4 g32	–	61.8(+3.5)
QA-LoRA	LLaMA-1-7B	W2 g32	–	53.7(−4.6)
LoftQ	LLaMA-2-7B	W4 NF4	5.24(+0.16)	–
LoftQ	LLaMA-2-7B	W2 NF2	7.85(+2.77)	–
LoftQ	LLaMA-2-13B	W4 NF4	5.16(+0.04)	–
LoftQ	LLaMA-2-13B	W2 NF2	7.69(+2.57)	–
L4Q	LLaMA-1-7B	W4 g128	–	62.7(+1.0)
L4Q	LLaMA-1-13B	W4 g128	–	64.5(+0.7)
L4Q	LLaMA-2-7B	W4 g128	–	63.6(+1.7)
L4Q	LLaMA-2-13B	W4 g128	–	65.8(+0.8)

Table 12: WikiText-2 perplexity and zero-shot accuracy for base LLMs. The baseline of LoftQ is the 16-bit LoRA model.

Method	**Base Model**	**Bits**	**MMLU ↑**	**Avg. ↑**
QA-LoRA	LLaMA-1-7B	W4	39.4(+4.8)	61.8(+3.5)
QA-LoRA	LLaMA-1-7B	W3	37.4(+2.8)	60.1(+1.8)
QA-LoRA	LLaMA-1-7B	W2	27.5(−7.1)	53.7(−4.6)
DL-QAT	LLaMA-1-7B	W4	39.9(+5.3)	63.4(+5.1)
DL-QAT	LLaMA-1-7B	W3	33.9(−0.7)	61.1(+2.8)
L4Q	LLaMA-1-7B	W4	35.7(+0.6)	62.7(+1.0)
L4Q	LLaMA-1-7B	W3	31.8(−3.3)	61.2(−0.5)

Table 13: Instruction-tuned Quantized LLMs on MMLU and Commonsense Benchmarks. MMLU denotes 5-shot MMLU accuracy. Avg. denotes the reported zero-shot commonsense QA average

Method	Base Model	Benchmark	Bits	FID ↓
EfficientDM	DDIM	CIFAR-10	W4A4	10.48 (+6.34)
EfficientDM	LDM-4	LSUN-Bedrooms	W4A4	10.60 (+7.17)
EfficientDM	LDM-8	LSUN-Churches	W4A4	14.34 (+10.26)
BiDM	DDIM	CIFAR-10	W1A1	81.65 (+76.11)
BiDM	LDM-4	LSUN-Bedrooms	W1A1	22.74 (+19.75)
BiDM	LDM-8	LSUN-Churches	W1A1	29.70 (+25.34)

Table 14: FID Results for Quantized Diffusion Models.