

Sensitivity-aligned Mixed-precision Quantization and Low-rank Adaptation for LLMs

Syed Faizan Hussain Shah

Jeju National University

Gyeongmin Kim

gmkim@jejunu.ac.kr

Jeju National University

Article

Keywords: mixed-precision quantization, large language models, parameter-efficient fine-tuning, qLoRA, model compression

Posted Date: October 16th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11015135/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Sensitivity-aligned Mixed-precision Quantization and Low-rank Adaptation for LLMs

Syed Faizan Hussain Shah^{1,2} and Gyeongmin Kim^{1,2*}

¹Department of Artificial Intelligence, Jeju National University, Jeju-si, 63243, Jeju Special Self-Governing Province, Republic of Korea.

²NAVI Hyper-Tropicalization Research Institute, Jeju National University, Jeju-si, 690-756, Republic of Korea.

*Corresponding author(s). E-mail(s): gmkim@jejunu.ac.kr;
Contributing authors: faizanbukhari8815@gmail.com;

Abstract

We propose a sensitivity-guided block-wise mixed-precision quantization framework with adaptive-rank low-rank adaptation for efficient large language models. The proposed framework assigns 4/6/8-bit precision at block granularity and allocates low-rank capacity according to quantization sensitivity and compression pressure. Attention-derived sensitivity guides mixed-bit allocation under a global precision budget, while adaptive ranks provide greater adaptation capacity to layers requiring stronger recovery from quantization-induced distortion. We evaluate the method across model families ranging from 1.1B to 8B parameters using WikiText-2, downstream reasoning benchmarks, and system-level measurements. Experimental results show that the proposed approach preserves downstream performance and, in several cases, improves perplexity relative to corresponding FP16 baselines. For the TinyLlama-1.1B configuration, the proposed method achieves a held-out test perplexity of 7.884 with a decoding throughput of 73.11 tokens/s. The mixed-bit allocation provides a theoretical reduction in packed parameter payload under the deployed rewrite scope, while measured runtime GPU memory remains close to the FP16 reference because the current implementation uses dequantized-buffer execution. Layer-wise analysis further reveals that quantization-induced distribution shift varies substantially across transformer depth, while adaptive fine-tuning provides complementary recovery in compressed regions. Overall, the results demonstrate that sensitivity-guided mixed-bit allocation combined with adaptive low-rank adaptation offers an effective approach for balancing model quality, precision allocation, and deployment-oriented efficiency in large language models.

Keywords: mixed-precision quantization, large language models, parameter-efficient fine-tuning, qLoRA, model compression

1 Introduction

Large Language Models (LLMs) have recently demonstrated strong empirical performance across a wide range of natural language processing (NLP) tasks, including reasoning, knowledge-intensive question answering, and multi-turn dialogue generation. Scaling model size has been a primary driver of these advances, with modern open-source models reaching billions of parameters. However, this improved performance comes at a considerable cost: the large memory footprint and high inference-time bandwidth demands make these models difficult to deploy on resource-constrained platforms, including edge devices, single-GPU systems, and privacy-sensitive offline environments [1–3].

To overcome these challenges, model compression has become an essential tool for enabling the practical deployment of LLMs. Among the widely explored techniques are pruning, distillation, and quantization. Pruning reduces computation by removing parameters or even whole layers, but when pushed too far it often causes noticeable and unstable drops in performance. Quantization has therefore gained more attention as a practical alternative, since it can significantly reduce memory usage while keeping accuracy largely intact. Among these methods, post-training quantization (PTQ) is particularly attractive because it provides these benefits without requiring the full model to be retrained [4–9].

Recent quantization methods focus on uniform precision assignments, such as quantizing all model weights to 8-bit or 4-bit representations [10–14]. While effective to some extent, uniform quantization overlooks the fact that different layers within a transformer contribute unequally to model performance. Empirical studies have shown that quantizing sensitive layers can disproportionately harm accuracy, especially at low bit-widths. To address this, several works have proposed mixed-precision or layer-wise quantization strategies [15–17], where layers are assigned different bit-widths based on heuristics such as weight statistics or representation changes [18–22]. These approaches demonstrate that variable-precision quantization can outperform uniform schemes under the same memory budget. However, they typically operate at the level of entire layers, assigning a single bit-width per layer based on aggregated importance metrics. In this work, we refer to such strategies as *coarse-grained*, as they do not capture intra-layer variability. In contrast, *fine-grained* modeling refers to assigning precision at a sub-layer level (e.g., block level) using localized sensitivity signals, enabling more precise allocation within transformer modules. This distinction is based on the resolution at which both sensitivity estimation and bit allocation are performed, rather than on a fixed external benchmark.

At the same time, advances in parameter-efficient fine-tuning (PEFT) methods, such as LoRA and qLoRA, have shown that low-rank adaptation can effectively specialize large models with minimal additional parameters [23]. These methods are

particularly attractive in quantized settings, as they enable task adaptation without modifying the full-precision backbone. However, most existing approaches apply uniform adapter configurations across layers [24], implicitly assuming that all layers benefit equally from additional adaptation capacity [16, 25, 26]. This assumption becomes particularly relevant in mixed-bit settings, where quantization severity varies across the network. Although prior work has studied nonuniform precision assignment and adaptive low-rank adaptation, the use of an explicit mixed-bit precision plan together with its calibration signal to configure subsequent layer-wise adapter ranks remains less explored.

In this work, we focus on using a planned mixed-bit allocation to configure layer-wise low-rank adaptation. A precision plan specifies which weight blocks receive particular bit-widths, but it does not uniquely determine the adapter ranks used during fine-tuning. The assigned precision and the calibration sensitivity scores used to construct the plan provide two distinct inputs to this decision: the former describes the intended numerical representation, whereas the latter records the priorities used during precision allocation. Neither quantity is, by itself, a direct measurement of the adaptation capacity required by a layer. We therefore examine an explicit heuristic that uses both quantities to configure adapter ranks and evaluate the resulting adapted model relative to applying the same final mixed-bit policy without adaptation.

To investigate this question, we propose a sequential framework that couples attention-aware adaptive block-wise quantization (A²BQ) with qLoRA⁺, a plan-guided low-rank adaptation strategy. A²BQ constructs block-level allocation scores from the variance of post-softmax attention probabilities collected on a calibration corpus and assigns 4-, 6-, or 8-bit precision under an average-bit constraint over the allocation map. The resulting sensitivity scores and assigned bit-widths are aggregated at the transformer-layer level and combined through a bounded heuristic to determine adapter ranks. qLoRA⁺ uses these ranks to train low-rank adapters on a frozen 4-bit NF4 backbone. After fine-tuning, the learned adapter updates are merged with the pretrained weights, and the planned mixed-bit policy is applied to eligible projections. The NF4 representation used during adaptation is therefore distinct from the final mixed-bit representation. The framework connects the two stages through the allocation plan rather than jointly optimizing bit-widths and ranks within a single training objective.

We evaluate the framework across five model families ranging from 1.1B to 8B parameters, using WikiText-2 perplexity as the primary measure of language-modeling quality. The main comparison examines mixed-bit rewriting without adaptation and the combined A²BQ+qLoRA⁺ configuration under the same specified final allocation policy and rewrite scope. Across the evaluated model families, the combined configuration achieves lower test perplexity than mixed-bit rewriting alone. These results characterize the effect of the complete adaptation pipeline under the evaluated precision policies; they do not isolate the independent contribution of adaptive rank from the additional adaptation capacity and other effects of fine-tuning. We additionally compare the final models with their corresponding FP16 references on downstream benchmarks. The task-dependent gains and losses in these comparisons show that lower WikiText-2 perplexity does not imply uniform improvement across downstream tasks.

The contribution of our study is an explicit connection between block-wise precision planning and layer-wise adaptation configuration, together with an empirical evaluation of the resulting models. Specifically, we make the following contributions:

- We propose A²BQ, an attention-derived block-wise mixed-precision allocation framework that uses post-softmax attention variance as a calibration signal to rank hidden-row blocks and assigns 4-, 6-, or 8-bit precision under an average-bit constraint.
- We introduce qLoRA⁺, a plan-guided adaptation strategy that conditions layer-wise adapter ranks on both the sensitivity information used by the precision planner and the resulting allocated precision, providing an explicit connection between mixed-bit precision planning and adaptation configuration.
- We develop a sequential deployment pipeline in which adapters are trained using a frozen NF4 backbone, merged with the pretrained weights, and subsequently subjected to A²BQ mixed-bit rewriting with explicit module-eligibility and boundary-protection constraints.
- We provide a layer-wise KL-divergence analysis to characterize how the mixed-bit and adapted mixed-bit configurations alter model distributions across transformer depth. The observed effects are architecture dependent rather than uniformly beneficial across all evaluated models.
- We evaluate our approach across multiple model architectures and downstream benchmarks while distinguishing theoretical packed-weight payload from the runtime memory behavior of the reference implementation, which reconstructs quantized weights into floating-point buffers during inference.

2 Related Work

PTQ is one of the most widely adopted approaches for reducing the memory footprint and inference cost of LLMs, as it avoids expensive retraining while providing substantial compression benefits. Early and widely used PTQ methods typically apply a uniform bit-width across all layers, focusing on weight-only quantization or joint weight-activation quantization to reduce memory traffic and storage requirements [4, 11, 27]. Despite their effectiveness at moderate precision, uniform low-bit quantization often leads to significant accuracy degradation as bit-width decreases, and this degradation is not evenly distributed across model components [8, 28, 29]. Empirical analyses have shown that attention projections and feed-forward layers are particularly sensitive to quantization noise, motivating more structured calibration and error-mitigation strategies [30–32].

Several works have addressed these limitations by improving PTQ robustness through calibration techniques and system-aware design [33]. ABQ-LLM, for example, targets extreme low-bit inference by introducing block-wise distribution correction and bit-balancing strategies, together with a GPU execution engine that supports arbitrary bit-width computation [12, 34–37]. Related efforts such as Cross-Block Quantization further explore cross-block correction mechanisms to reduce quantization-induced distribution shift [38]. While these approaches substantially improve the robustness and efficiency of low-bit inference, they primarily treat quantization error as a calibration

or systems problem [39], and do not explicitly consider how learning-based adaptation can compensate for mixed-precision distortion [22].

These observations have motivated mixed-precision quantization strategies, where different layers are assigned different bit-widths under a fixed memory or latency budget [18–20]. The central premise of this line of work is that LLMs exhibit strong heterogeneity in layer sensitivity, and that preserving higher precision in critical layers can significantly improve accuracy retention compared to uniform quantization. Importance-guided allocation methods typically rely on sensitivity estimates derived from data-driven signals, such as representation perturbations [40], or from data-free statistics based on weight distributions and outlier behavior [17, 41]. While these methods consistently outperform naive layer-wise quantization, they remain largely layer-granular and therefore do not capture fine-grained variation within layers, such as differences across attention heads or feed-forward blocks.

In parallel, PEFT has emerged as a dominant paradigm for adapting large models under memory constraints. Low-rank adaptation methods such as LoRA and qLoRA enable effective fine-tuning with a small number of additional parameters, and are particularly attractive in quantized settings where updating full-precision weights is impractical [26, 42]. However, most existing PEFT pipelines assume a fixed adapter rank across layers, implicitly treating all layers as equally capable of absorbing adaptation. This assumption becomes increasingly mismatched in mixed-precision models, where quantization severity and representational capacity vary substantially across depth and across submodules.

Several recent works have begun to explore tighter coupling between quantization and adaptation. In particular, LoFTQ explicitly models the interaction between low-rank adaptation and quantization error, demonstrating that adaptation capacity should be concentrated in regions most affected by precision loss [16]. This observation suggests that uniform adapter placement is suboptimal in heterogeneous precision settings, motivating joint allocation of precision and adaptation capacity to better recover model performance.

Beyond algorithmic compression, system-oriented studies examine quantization in the context of deployment constraints. Works such as QServe co-design quantization formats, KV-cache precision, and GPU execution kernels to optimize throughput during LLM serving [43, 44], while other studies integrate quantization decisions into edge scheduling and resource allocation frameworks [1, 3]. These approaches highlight the importance of system-level orchestration, but typically assume fixed quantization policies and focus on scheduling efficiency rather than model robustness under heterogeneous precision.

While prior quantization methods such as GPTQ and AWQ have demonstrated competitive performance, they fundamentally differ from our approach in both granularity and optimization strategy. GPTQ performs sequential weight quantization using approximate second-order information, whereas AWQ applies activation-aware scaling to preserve salient weight channels. However, these methods typically operate at layer- or channel-level granularity and treat quantization as a standalone process [45, 46].

In contrast, A²BQ introduces block-wise sensitivity-aware quantization, enabling fine-grained precision allocation within transformer layers. Furthermore, our framework jointly integrates mixed-bit quantization with adaptive-rank qLoRA⁺ fine-tuning, where adaptation capacity is explicitly aligned with quantization severity. This joint design improves recovery from precision loss in highly compressed regions, which is not addressed in existing PTQ or mixed-precision methods. The following section details the proposed A²BQ framework, including block-wise sensitivity estimation, mixed-bit allocation, adaptive qLoRA⁺ optimization, and deployment-oriented model rewriting.

3 Methodology

We propose an end-to-end framework for building a memory-efficient LLM through adaptive block-wise mixed-bit quantization and quantization-aware parameter-efficient adaptation. The key idea is to avoid uniform treatment of the network: blocks that are likely to be sensitive to quantization are preserved at higher precision, while less sensitive blocks are aggressively compressed. To compensate for capacity loss in low-bit regions, the LoRA capacity is allocated non-uniformly, prioritizing blocks that are both more aggressively quantized and empirically more sensitive. The final evaluation model is obtained by merging the learned adaptive low-rank updates into the base weights and applying the mixed-bit quantization policy to eligible linear projections. In the reference implementation used in this study, quantized weights are subsequently reconstructed into floating-point dequantized buffers for inference. Accordingly, the learned precision map characterizes the numerical mixed-precision policy and its potential packed storage requirement, whereas runtime measurements correspond to the dequantized-buffer backend.

As shown in Fig. 1, the framework consists of three main stages: (i) block-wise sensitivity estimation, (ii) constrained mixed-bit allocation under a global bit budget, and (iii) quantization-aware adaptive fine-tuning followed by mixed-bit rewriting for deployment. Let $\mathbf{W}_i \in \mathbb{R}^{m \times n}$ be the weight matrix of the i -th linear transformation. We partition $\mathbf{W}_i$ into contiguous *row* blocks of fixed size b by *stacking*:

$$\mathbf{W}_i = [\mathbf{B}_{i,1}; \mathbf{B}_{i,2}; \dots; \mathbf{B}_{i,J_i}], \quad (1)$$

where

$$\mathbf{B}_{i,j} \in \mathbb{R}^{r_{i,j} \times n}, \quad r_{i,j} = \begin{cases} b, & j < J_i, \\ m - (J_i - 1)b, & j = J_i, \end{cases} \quad J_i = \left\lceil \frac{m}{b} \right\rceil. \quad (2)$$

Each block $\mathbf{B}_{i,j}$ is the atomic unit for sensitivity estimation and bit-width assignment. Block-level statistics are subsequently aggregated at the layer level to determine adaptive low-rank capacity. We fix b across layers ($b = 128$ in the main experiments) to provide consistent allocation granularity across architectures. The resulting mixed-bit plan assigns a bit-width $b_{i,j}$ to each block. These assignments determine the block-wise quantization operation and define the theoretical packed parameter payload associated with the allocation.

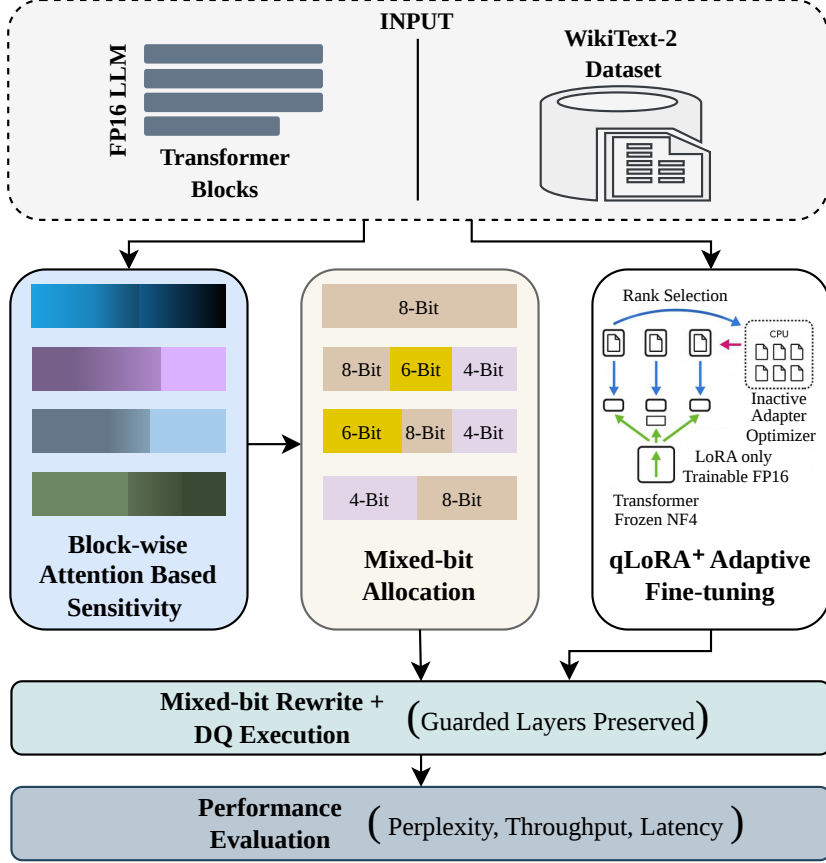


Fig. 1: End-to-end pipeline of the proposed sensitivity-aligned mixed-precision framework. Attention-derived block sensitivity guides 4/6/8-bit allocation under a global precision budget, followed by adaptive-rank fine-tuning, adapter merging, and mixed-bit rewriting using the dequantized-buffer reference backend.

3.1 Attention-Derived Block-Wise Sensitivity

We estimate per-row-block sensitivity from *attention probabilities* produced by the FP16 base model on a small calibration set. Attention probabilities are computed using the model’s standard forward-pass attention mechanism with causal masking, and sensitivity is derived from the post-softmax attention distributions.

For transformer layer i and attention head h , let $\mathbf{A}^{(i,h)}(x) \in [0, 1]^{T \times T}$ denote the post-softmax attention matrix returned for calibration chunk x . For each chunk, we measure the dispersion of the attention probabilities using the variance over the entries

of the returned attention matrix:

$$S_{\text{var}}^{(i,h)}(x) = \text{Var}_{t,t'} \left(A_{t,t'}^{(i,h)}(x) \right). \quad (3)$$

This statistic captures how strongly the post-softmax attention probabilities deviate from a uniform distribution within a head. In contrast, the mean over all entries of a row-normalized post-softmax attention matrix is determined by the sequence length and therefore does not provide a discriminative layer-head sensitivity signal.

Let $\mathcal{C}$ denote the set of valid token chunks generated from the calibration documents. We first average the variance statistic across the calibration chunks:

$$\bar{S}_{\text{var}}^{(i,h)} = \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} S_{\text{var}}^{(i,h)}(x). \quad (4)$$

The resulting layer-head statistics are then standardized globally over all layer-head pairs:

$$\tilde{S}_{\text{var}}^{(i,h)} = \frac{\bar{S}_{\text{var}}^{(i,h)} - \mu_{\text{var}}}{\sigma_{\text{var}} + \epsilon}, \quad (5)$$

where μ_{var} and σ_{var} denote the global mean and standard deviation of the chunk-averaged attention-variance statistics, respectively, and ϵ is a small constant used for numerical stability.

Let the hidden size be H with N_h attention heads, giving $d_h = H/N_h$ hidden coordinates per head. The hidden dimension is partitioned into contiguous row blocks,

$$\mathbf{W}_i = [B_{i,1}; B_{i,2}; \dots; B_{i,J_i}], \quad (6)$$

where each block $B_{i,j}$ spans a contiguous row range $[r_0, r_1]$ of fixed size $r_1 - r_0 + 1 = 128$ in the main experiment while the block-size ablation additionally evaluated $b \in \{64, 128, 256\}$.

The set of attention heads whose corresponding hidden-coordinate ranges intersect the row range of block $B_{i,j}$ is

$$h_0 = \left\lfloor \frac{r_0}{d_h} \right\rfloor, \quad h_1 = \left\lfloor \frac{r_1}{d_h} \right\rfloor, \quad \mathcal{H}(B_{i,j}) = \{h \mid h_0 \leq h \leq h_1\}. \quad (7)$$

The final block-level sensitivity score is computed as the unweighted mean of the standardized attention-variance scores of the heads covered by the block:

$$S(B_{i,j}) = \frac{1}{|\mathcal{H}(B_{i,j})|} \sum_{h \in \mathcal{H}(B_{i,j})} \tilde{S}_{\text{var}}^{(i,h)}. \quad (8)$$

The resulting $S(B_{i,j})$ is used as the block-level allocation score, with larger values indicating blocks for which the attention distributions exhibit greater variation and therefore receive more conservative precision assignments during the mixed-bit allocation stage. Fig. 2 summarizes the sensitivity-estimation and allocation workflow.

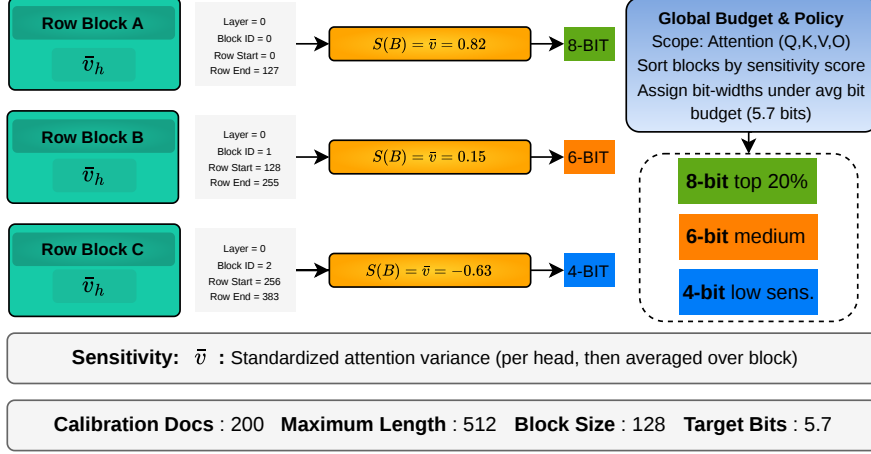


Fig. 2: Block-wise sensitivity estimation and mixed-bit assignment within a transformer layer.

3.2 Mixed-Bit Allocation with Block Granularity

Given the map $S(B_{i,j})$ from Eq. (8), we assign each block a bit-width $b_{i,j} \in \mathcal{W}$ (e.g., $\mathcal{W} = \{4, 6, 8\}$) to meet a global average-bit target while minimizing a sensitivity-weighted penalty. The fused block sensitivities are normalized across all blocks using z -score normalization:

$$\tilde{S}(B_{i,j}) = \text{zscore}(S(B_{i,j})). \quad (9)$$

Let $r_{i,j}$ be the number of rows in block $B_{i,j}$ (typically 128, smaller for the tail block). We formulate the mixed-bit allocation objective as

$$\min_{\{b_{i,j} \in \mathcal{W}\}} \sum_{i,j} \tilde{S}(B_{i,j}) \Delta(b_{i,j}) \quad (10)$$

subject to a row-weighted average-bit budget. We impose a global average bit-width constraint $\bar{b}$, which defines the target average precision across all blocks and is chosen to satisfy a desired compression or memory budget:

$$\frac{1}{\sum_{i,j} r_{i,j}} \sum_{i,j} r_{i,j} b_{i,j} \leq \bar{b}. \quad (11)$$

Here $\Delta(\cdot)$ is a monotone penalty encoding the relative risk of low precision (e.g., $\Delta(4) > \Delta(6) > \Delta(8)$). This formulation corresponds to the allocation stage shown in Fig. 2, where sensitivity estimates are converted into a constrained precision assignment across blocks.

The target $\bar{b}$ is imposed over the complete allocation map. When boundary-layer guards are subsequently applied during model rewriting, those protected layers are

excluded from low-bit execution. Consequently, the global allocation average and the average precision within the actively rewritten region are distinct quantities. We report both quantities in the block-size ablation 6.2.

We implement a greedy cost-risk procedure: all blocks start at a conservative precision, and we apply downgrades $b \rightarrow b'$ ($b' < b$) in order of the smallest *risk per bit saved*. A candidate for block $B_{i,j}$ has

$$\text{risk}(B_{i,j}, b \rightarrow b') = \tilde{S}(B_{i,j}) [\Delta(b') - \Delta(b)], \quad \text{savings}(B_{i,j}, b \rightarrow b') = r_{i,j} (b - b'), \quad (12)$$

and we rank by

$$\frac{\text{risk}(B_{i,j}, b \rightarrow b')}{\text{savings}(B_{i,j}, b \rightarrow b')}. \quad (13)$$

We apply the lowest ratios until the constraint in Eq. (11) is met, while protecting the top 20% most sensitive allocation blocks at 8-bit precision. The resulting mixed-bit map determines the block-wise quantization policy and provides the precision information used to construct the adaptive-rank configuration for the subsequent low-rank adaptation stage.

3.3 Quantization-Aware Adaptive Low-Rank Adaptation

After constructing the mixed-bit allocation, we perform parameter-efficient fine-tuning using low-rank adapters while keeping the base model frozen. Unlike conventional qLoRA configurations that use a fixed adapter rank across all target layers, the proposed qLoRA⁺ strategy assigns the nonuniform adaptation capacity according to both the estimated quantization sensitivity and the allocated precision of each transformer layer, as illustrated in Fig. 3.

To determine the adaptive rank, the block-wise sensitivity scores and mixed-bit assignments are first aggregated at the layer level using row-weighted averages. For transformer layer i , the average sensitivity and allocated precision are defined as

$$\bar{S}_i = \frac{\sum_j r_{i,j} S(B_{i,j})}{\sum_j r_{i,j}}, \quad \bar{b}_i = \frac{\sum_j r_{i,j} b_{i,j}}{\sum_j r_{i,j}}, \quad (14)$$

where $r_{i,j}$ denotes the number of rows in block $B_{i,j}$, $S(B_{i,j})$ is its fused sensitivity score, and $b_{i,j}$ is its assigned bit-width.

The layer-level sensitivity is then standardized across transformer layers:

$$\hat{S}_i = \frac{\bar{S}_i - \mu_S}{\sigma_S + \epsilon}, \quad (15)$$

where μ_S and σ_S denote the mean and standard deviation of the layer-level sensitivity values, respectively, and ϵ is a small constant for numerical stability.

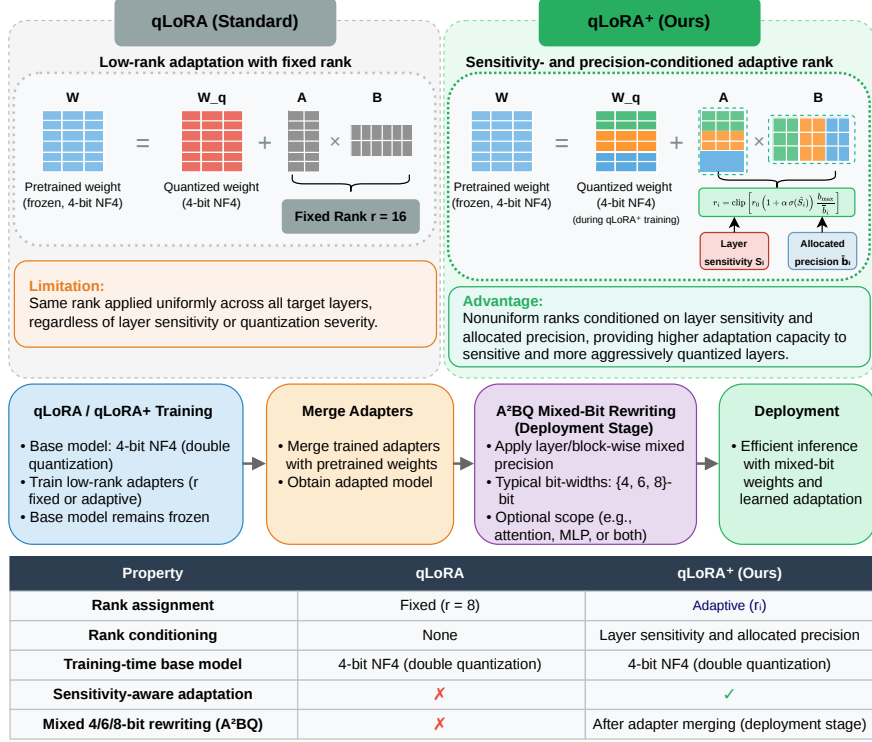


Fig. 3: Comparison of fixed-rank qLoRA and the proposed qLoRA⁺. qLoRA⁺ assigns nonuniform layer-wise adapter ranks conditioned on attention-derived sensitivity and allocated precision, followed by adapter merging and A²BQ mixed-bit rewriting for deployment.

Using the normalized sensitivity and average allocated precision, the adaptive LoRA rank for layer i is computed as

$$r_i = \text{clip}_{[r_{\min}, r_{\max}]} \left(\text{round} \left[r_0 \left(1 + \alpha \sigma(\hat{S}_i) \right) \frac{b_{\max}}{b_i} \right] \right), \quad (16)$$

where r_0 denotes the base rank, $\sigma(\cdot)$ is the logistic sigmoid function, α controls the contribution of the normalized sensitivity signal, and $b_{\max}$ is the maximum bit-width considered by the mixed-bit allocator. The rank is rounded to the nearest integer and bounded within $[r_{\min}, r_{\max}]$. In our experiments, we use $r_0 = 8$, $r_{\min} = 4$, $r_{\max} = 32$, $\alpha = 0.75$, and $b_{\max} = 8$. The sensitivity scaling coefficient $\alpha = 0.75$ is kept fixed across all experiments to provide a moderate sensitivity-dependent adjustment of the adapter rank and is not optimized separately for individual models or evaluation datasets.

The formulation is designed as a simple monotonic heuristic rather than as an analytically optimal rank-allocation rule. It is designed such that the assigned rank

increases with layer sensitivity and with quantization severity. The sigmoid $\sigma(\hat{S}_i)$ provides a smooth bounded transformation of the normalized sensitivity score, while the precision factor $b_{\max}/\bar{b}_i$ increases as the average allocated bit-width decreases. Their multiplicative combination therefore assigns greater adaptation capacity to layers that are simultaneously more sensitive and more aggressively quantized, while clipping to $[r_{\min}, r_{\max}]$ limits excessive parameter growth.

Importantly, Eq. (16) does not impose a fixed total adapter-parameter budget. With the hyperparameters used in this work, the resulting ranks can exceed the base rank r_0 ; consequently, qLoRA⁺ should be interpreted as a sensitivity- and precision-conditioned nonuniform adaptation-capacity rule rather than as a redistribution of a fixed rank budget. For the TinyLlama-1.1B configuration reported in Table 1, the saved adaptive rank map contains 154 module-specific assignments with ranks ranging from 11 to 19. This configuration contains 12,257,280 trainable adapter parameters, compared with 6,307,840 parameters for the fixed $r = 8$ qLoRA baseline. Therefore, the comparison between these configurations evaluates the complete adaptive qLoRA⁺ design and should not be interpreted as a parameter-budget-matched comparison of fixed versus adaptive rank allocation.

The resulting rank r_i is assigned to all targeted projection modules within transformer layer i . Specifically, the adaptive fine-tuning stage targets the attention projections (`q_proj`, `k_proj`, `v_proj`, and `o_proj`) and the MLP projections (`gate_proj`, `up_proj`, and `down_proj`) when present in the corresponding architecture. Thus, the implementation uses a per-module rank map, but the assigned rank is determined at the layer level; target projections within the same transformer layer receive the same rank. These qLoRA⁺ adapter targets should be distinguished from the subsequent mixed-bit deployment rewrite scope. The adapter stage permits adaptation of the attention and MLP projection modules listed above, whereas the final A²BQ rewriting stage operates on a restricted subset of eligible projection matrices selected by the deployment rewriting procedure in Section 3.4. Consequently, a module may receive a learned low-rank update during qLoRA⁺ fine-tuning without itself being rewritten using mixed-bit quantization during deployment.

During qLoRA⁺ fine-tuning, the frozen base model is represented using 4-bit NF4 quantization with double quantization enabled, with floating-point computation used for the quantized linear operations. Trainer mixed precision uses BF16 when supported and otherwise falls back to FP16. Only the low-rank adapter parameters are updated. After training, the original pretrained model is reloaded, the learned adapters are attached and merged into the corresponding pretrained weights, and the resulting merged model is subjected to the final mixed-bit rewriting procedure described in Section 3.4. Thus, NF4 serves as the memory-efficient training representation and is not the source representation from which the final mixed-bit weights are constructed. The complete qLoRA⁺ optimization and data-processing configuration is reported in Supplementary Section S5 for reproducibility.

3.4 Adapter Merging and Mixed-Bit Model Rewriting

After qLoRA⁺ fine-tuning, the learned adapter parameters are not merged into the temporary NF4 representation used during adapter training. Instead, the original pre-trained checkpoint is loaded again in the floating-point execution datatype, the trained qLoRA⁺ adapters are attached to this model, and the low-rank updates are merged into the corresponding pretrained weights before mixed-bit rewriting. For a linear module in layer i , the resulting merged weight is

$$\mathbf{W}_i^{\text{merge}} = \mathbf{W}_i^{\text{pre}} + \Delta\mathbf{W}_i^{\text{qLoRA}^+}, \quad (17)$$

where $\mathbf{W}_i^{\text{pre}}$ denotes the weight loaded from the original pretrained checkpoint and $\Delta\mathbf{W}_i^{\text{qLoRA}^+}$ denotes the learned low-rank update obtained from the qLoRA⁺ adaptation stage described in Section 3.3. In the reported deployment pipeline, the pretrained model is loaded in FP16 on CUDA, the trained PEFT adapters are attached, and the updates are merged using the PEFT merging operation before mixed-bit rewriting. NF4 is therefore used for the frozen base representation during qLoRA⁺ fine-tuning, but it is not the source weight representation into which the adapters are merged for the final mixed-bit rewrite. After merging, the adapter parameters are absorbed into the floating-point model weights and no separate adapter modules are required during the subsequent rewriting stage.

The merged model is then rewritten according to the block-wise mixed-precision allocation map $\mathcal{A} = \{b_{i,j}\}$ obtained from the allocation procedure. For each eligible weight matrix, the rewrite can be expressed as

$$\tilde{\mathbf{W}}_i = \mathcal{Q}(\mathbf{W}_i^{\text{merge}}; \mathcal{A}), \quad (18)$$

where $\mathcal{Q}(\cdot)$ denotes the block-wise quantization operator. For each eligible matrix, rows are partitioned into blocks of the specified block size, and block j in transformer layer i is quantized according to the bit-width $b_{i,j}$ provided by the allocation map $\mathcal{A}$. The same validation-derived allocation map is used when evaluating the mixed-bit-only and merged qLoRA⁺ plus mixed-bit configurations, so that the effect of adapter adaptation is not confounded with a different precision assignment.

Rewriting is controlled by a module scope $\mathcal{S} \in \{\text{attn}, \text{mlp}, \text{all}\}$ together with module-eligibility and boundary-layer constraints. The language-modeling head is excluded from rewriting, and the first and last transformer layers can be preserved through configurable boundary guards. Within the requested scope, a linear projection is eligible for rewriting only when its output dimension equals the model hidden dimension. Consequently, the effective set of rewritten modules is architecture dependent. For example, under the TinyLlama attention-scope configuration, this condition retains `q.proj` and `o.proj` for mixed-bit rewriting, whereas the grouped-query `k.proj` and `v.proj` matrices are excluded because their output dimensions are smaller than the model hidden dimension. Under the MLP scope, the same output-dimension condition retains `down.proj` while excluding the other MLP projections whose output dimensions do not satisfy the eligibility condition. In the principal TinyLlama configuration, the first and last three transformer layers are protected, so mixed-bit rewriting is applied only to eligible projections in the remaining transformer layers.

The reference backend used for the reported mixed-bit experiments employs dequantize-and-compute (DQ) execution. During module rewriting, the assigned block-wise precisions are applied to the eligible weight rows and the resulting quantized values are reconstructed into a floating-point weight tensor used for execution. This operation can be represented as

$$\mathbf{W}_i^{\text{exec}} = \mathcal{D}(\tilde{\mathbf{W}}_i), \quad (19)$$

where $\mathcal{D}(\cdot)$ denotes dequantization into the floating-point execution datatype. The forward-pass computation then uses the dequantized weight through the standard floating-point linear operation

$$\mathbf{y} = \mathcal{F}(\mathbf{W}_i^{\text{exec}}, \mathbf{x}), \quad (20)$$

where $\mathbf{x}$ denotes the input activation and $\mathcal{F}(\cdot)$ represents the standard linear transformation. Thus, the reference backend preserves the numerical effect of the assigned block-wise precision during weight reconstruction, while the actual forward computation is performed using floating-point weights.

Importantly, the DQ reference backend does not perform matrix multiplication directly on physically packed low-bit weights. We therefore distinguish the estimated packed-weight parameter payload implied by the mixed-bit allocation from the measured runtime GPU-memory footprint of the DQ implementation. The former estimates the weight-storage requirement that would be obtained if the selected quantized blocks were physically packed according to their assigned bit-widths, whereas the latter reflects the actual DQ execution used in the experiments. This distinction prevents the estimated mixed-bit storage reduction from being interpreted as the resident low-bit inference memory of the current backend.

The complete rewrite configuration is therefore determined by the allocation map $\mathcal{A}$, rewrite scope $\mathcal{S}$, block size, boundary-layer guards, module-eligibility condition, and execution datatype. Algorithm 1 summarizes the rewriting procedure used for the reported experiments.

3.5 Parameter-Payload Accounting and Runtime Memory Profiling

We distinguish between parameter-storage estimates and measured runtime memory because the two quantities represent different aspects of the current implementation. For conventional quantized baselines whose quantized tensors are physically instantiated, the parameter payload is measured directly from the stored tensor representation. For the proposed mixed-bit configurations, the packed parameter payload is estimated from the exact modules eligible for rewriting and their assigned block-wise precisions.

The latter quantity represents the parameter storage that would be required under ideal physical packing of the deployed 4/6/8-bit assignments. It does not represent the resident memory of the current reference backend, which reconstructs rewritten weights into floating-point buffers before inference. Runtime memory is therefore evaluated independently using peak GPU memory during generation. Throughout the

Algorithm 1 Mixed-Bit Rewriting with Dequantized-Buffer Execution

- 1: **Input:** Pretrained checkpoint weights W^{pre} , trained qLoRA⁺ adapter updates $\Delta W^{\text{qLoRA}^+}$, allocation map $\mathcal{A} = \{b_{i,j}\}$, rewrite scope $\mathcal{S}$, block size, and boundary guards
 - 2: Load the original pretrained checkpoint in the floating-point execution datatype
 - 3: Attach the trained qLoRA⁺ adapters
 - 4: Merge the learned updates: $W_i^{\text{merge}} \leftarrow W_i^{\text{pre}} + \Delta W_i^{\text{qLoRA}^+}$
 - 5: Remove the separate adapter modules after merging
 - 6: Select modules belonging to $\mathcal{S} \in \{\text{attn}, \text{mlp}, \text{all}\}$
 - 7: Exclude the language-modeling head and protected boundary layers
 - 8: Retain only modules satisfying the output-dimension eligibility condition
 - 9: **for** each eligible weight matrix W_i^{merge} **do**
 - 10: Partition W_i^{merge} into row blocks $\{B_{i,j}\}$
 - 11: **for** each block $B_{i,j}$ **do**
 - 12: Obtain assigned bit-width $b_{i,j}$ from $\mathcal{A}$
 - 13: Quantize: $\tilde{B}_{i,j} \leftarrow \mathcal{Q}(B_{i,j}, b_{i,j})$
 - 14: **end for**
 - 15: Reconstruct the rewritten matrix $\tilde{W}_i$ from $\{\tilde{B}_{i,j}\}$
 - 16: Construct the floating-point execution tensor: $W_i^{\text{exec}} \leftarrow \mathcal{D}(\tilde{W}_i)$
 - 17: **end for**
 - 18: **Output:** Rewritten model with DQ execution tensors and rewrite metadata
-

results, measured payloads and estimated packed payloads are explicitly distinguished to avoid conflating storage potential with physical runtime residency.

4 Experimental Setup

4.1 Models

We conducted experiments primarily on TinyLlama-1.1B-Chat-v1.0, a decoder-only transformer following a LLaMA-style architecture with a hidden size of 2048 and 22 transformer layers. To evaluate the generality of the proposed framework, we further include Qwen2.5-3B-Instruct, Phi-3-mini-4k-instruct (3.8B), Mistral-7B-Instruct-v0.3, and LLaMA-3.1-8B-Instruct, covering a range of model sizes, depths, and hidden dimensions. Specifically, Qwen2.5-3B has a hidden size of 2048 and 36 layers, Phi-3-mini-4k (3.8B) has a hidden size of 3072 and 32 layers, while Mistral-7B and LLaMA-3.1-8B each have a hidden size of 4096 and 32 layers. With a fixed block size of 128 rows along the hidden dimension, the corresponding numbers of blocks per layer are 16, 16, 24, 32, and 32 for TinyLlama-1.1B, Qwen2.5-3B, Phi-3-mini-4k, Mistral-7B, and LLaMA-3.1-8B, respectively.

All models are obtained from their official HuggingFace releases to ensure reproducibility and consistency. The exact model identifiers are TinyLlama/TinyLlama-1.1B-Chat-v1.0, Qwen/Qwen2.5-3B-Instruct,

microsoft/Phi-3-mini-4k-instruct, mistralai/Mistral-7B-Instruct-v0.3, and meta-llama/Meta-Llama-3.1-8B-Instruct.

4.2 Calibration Corpus and Datasets

We use WikiText-2 consistently throughout the experimental pipeline. This dataset is widely adopted in the language model compression literature and enables direct and reproducible comparison of perplexity across methods. The training split is used for qLoRA and qLoRA⁺ fine-tuning, the validation split is used for sensitivity estimation, calibration, and layer-wise analysis, and the test split is reserved for final perplexity reporting.

For sensitivity estimation and mixed-bit allocation, we employ a small, fixed calibration corpus drawn from the WikiText-2 validation split. Specifically, we use the first 200 documents with a maximum sequence length of 512 tokens, and compute attention-based sensitivity statistics using forward-only passes without any weight updates. This calibration procedure is lightweight and incurs negligible computational overhead compared to fine-tuning, while providing stable and consistent sensitivity estimates across experiments. The resulting sensitivity scores are used by the allocator to assign block-wise 4-, 6-, and 8-bit precision under the target average-bit budget (5.7). Final perplexity values reported in the results are evaluated separately on the held-out WikiText-2 test split. All datasets are loaded using the HuggingFace `datasets` interface, and a fixed random seed is used for the main training and evaluation pipeline unless otherwise stated; the layer-wise KL analysis is repeated over five paired seeds.

4.3 Baselines

We compare the proposed method against a comprehensive set of baselines covering both PTQ and PEFT approaches:

1. **FP16 baseline**, using full-precision weights and optimized CUDA kernels.
2. **PTQ methods**, including:
 - **GPTQ** (weight-only quantization),
 - **AWQ** (activation-aware quantization), and
 - **NF4** (4-bit quantization with `bitsandbytes`).
3. **qLoRA (uniform rank)**, where a 4-bit NF4 base model is fine-tuned with a fixed LoRA rank across all layers.
4. **PEFT (qLoRA⁺)**, where a 4-bit NF4 base model is fine-tuned using adaptive low-rank allocation.
5. **Mixed-bit only (A²BQ)**, applying sensitivity-guided block-wise mixed-bit quantization without fine-tuning.
6. **A²BQ + qLoRA⁺ (ours)**, combining sensitivity-aware mixed-bit allocation, adaptive-rank fine-tuning, adapter merging, and mixed-bit rewriting.

We use the attention-based block-wise sensitivity estimation and mixed-bit allocation described in Sections 3.1-3.2. Sensitivity is computed on the validation split using

forward-only passes, producing a block-level sensitivity profile with a default block size of 128 rows. Based on this profile, A²BQ assigns block-wise precision from {8, 6, 4} under a fixed average-bit constraint, with optional protection of boundary layers and the most sensitive blocks.

Quantization-aware fine-tuning is then performed using qLoRA⁺ as described in Section 3.3. The base model is loaded in 4-bit NF4 format, and LoRA adapters are trained with layer-specific ranks determined by both the sensitivity statistics and the mixed-bit allocation plan. Unless otherwise stated, training uses the 8-bit AdamW optimizer, a learning rate of 2.5×10^{-4} , cosine scheduling with warmup, gradient accumulation of 16, a sequence length of 512, and a single epoch of fine-tuning.

After fine-tuning, the learned adapters are merged into the base weights, after which eligible projections are rewritten according to the mixed-bit allocation map described in Section 3.4. All proposed mixed-bit results reported in this study use the dequantized-buffer reference backend. The exact projections selected depend on the requested rewrite scope, boundary guards, and output-dimension eligibility condition.

4.4 Evaluation Metrics

We evaluate all models using the following metrics:

1. **Perplexity (PPL)** on the WikiText-2 test split, computed using a sliding-window evaluator.
2. **Inference performance**, reported as decode throughput (tokens/s) and latency (ms/token).
3. **Peak GPU memory**, measured as the maximum allocated memory during generation.
4. **Layer-wise KL divergence**, computed between FP16 and quantized models to analyze distributional shifts across depth.

4.5 Hardware and Software Environment

All experiments are conducted on a single-node system running Ubuntu 22.04 LTS. Unless otherwise stated, each experiment is executed on a single NVIDIA H100 PCIe GPU with 80 GB of memory to ensure fair and reproducible comparisons across all configurations. No form of multi-GPU or distributed parallelism is used in any experiment. The system is powered by dual-socket Intel Xeon Platinum 8562Y+ CPUs. The software stack consists of CUDA 12.2, NVIDIA driver 535.183.01, PyTorch, HuggingFace Transformers, PEFT, and BitsAndBytes. For the proposed mixed-bit configurations, all reported runtime measurements use the dequantized-buffer reference implementation. During model rewriting, the assigned block-wise precisions are applied and the resulting quantized weights are reconstructed once into floating-point execution buffers. Forward inference then uses standard floating-point linear operations. Consequently, the reported throughput and latency measure the behavior of this reference implementation and should not be interpreted as performance from native packed 4/6/8-bit GPU kernels.

5 Results and Analysis

5.1 Mixed-bit Allocation Patterns Across Models

To evaluate the consistency of the proposed allocation strategy across different architectures, we examined the mixed-bit precision assignments produced by A²BQ under a common target average bit-width of 5.7 bits. Despite substantial variation in model depth, hidden size, and total number of blocks, A²BQ consistently maintains the target budget while producing structured combinations of 4-, 6-, and 8-bit blocks. Specifically, the resulting average bit-width is 5.69 bits for TinyLlama-1.1B, Qwen2.5-3B, Phi-3-mini-4k (3.8B), Mistral-7B, and LLaMA-3.1-8B, compared with the target value of 5.7 bits.

The allocation patterns nevertheless vary across architectures. TinyLlama-1.1B assigns 123, 159, and 70 blocks to 4-, 6-, and 8-bit precision, respectively, out of 352 blocks. For Qwen2.5-3B, the corresponding allocation is 326 4-bit, 11 6-bit, and 239 8-bit blocks out of 576. Phi-3-mini-4k allocates 422, 40, and 306 blocks to 4-, 6-, and 8-bit precision, respectively, across 768 blocks. Mistral-7B and LLaMA-3.1-8B exhibit similar allocation patterns, with 572/34/418 and 574/30/420 blocks assigned to 4-, 6-, and 8-bit precision, respectively, across 1024 blocks. These results indicate that A²BQ adapts the distribution of precision levels to the characteristics of each model while maintaining a stable global bit-width budget. The variation in allocation proportions across architectures further suggests that the resulting mixed-precision patterns are determined by model-specific sensitivity rather than by a fixed, architecture-independent allocation rule.

5.2 Qualitative Analysis of Sensitivity-Guided Allocation

Fig. 4 visualizes how the allocation changes from a high-precision baseline to a sensitivity-shaped mixture. Before allocation, most blocks are kept at 8-bit, with guard regions around boundary layers. After allocation, mid-layers absorb most of the compression and become a structured mixture of 4/6/8-bit blocks, while protected regions remain higher precision. The resulting allocation aligns with the global target bit-width and yields consistent block-count distributions across model families, matching the summary statistics in the Section 5.1

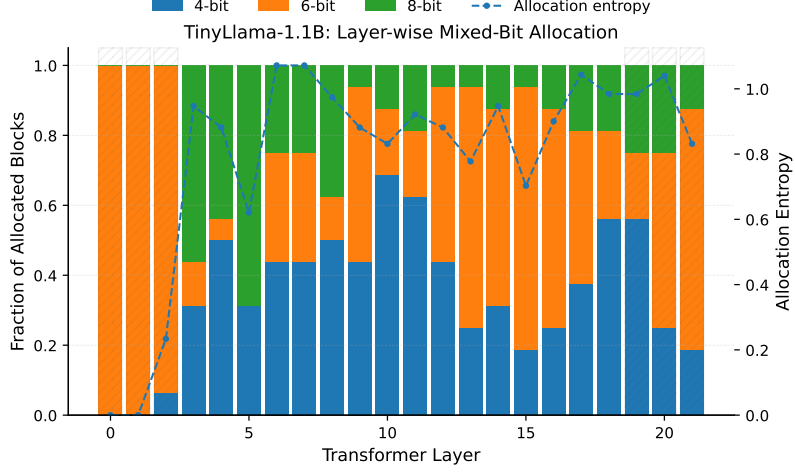


Fig. 5: Layer-wise bit distribution and allocation entropy after A²BQ. The stacked bars show the fraction of 4/6/8-bit blocks in each layer, while the entropy curve measures the diversity of bit-widths within a layer. Higher entropy indicates stronger intra-layer mixed-precision behavior.

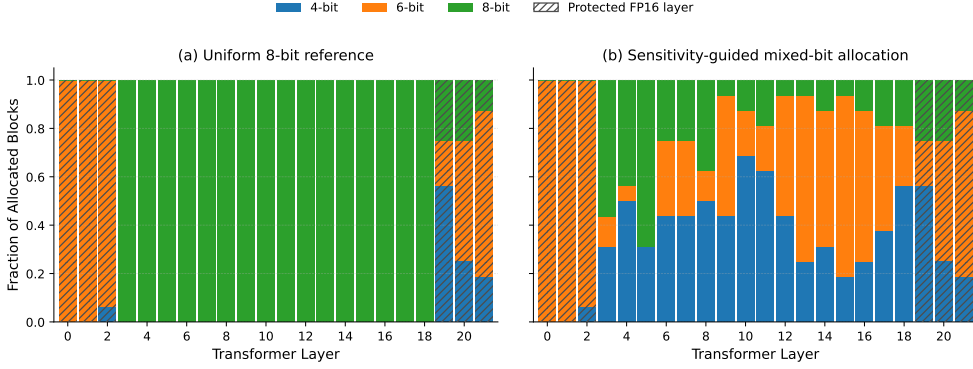


Fig. 4: Block-wise precision distribution across layers before allocation (8-bit-heavy baseline with guards) and after A²BQ sensitivity-based allocation.

To further quantify this behavior, Fig 5 reports the layer-wise bit distribution together with allocation entropy. Higher entropy indicates that a layer contains a more diverse mixture of bit-widths, whereas lower entropy indicates a more uniform precision assignment. The results show that A²BQ does not simply assign one precision per layer; instead, it produces heterogeneous block-level allocations within layers. This supports the proposed fine-grained design and confirms that the global target bit-width is achieved through structured intra-layer allocation rather than uniform compression.

To verify that the learned allocation is aligned with the proposed sensitivity signal, Fig 6 relates the layer-level attention-aware sensitivity to the rows-weighted average precision assigned by A²BQ. A positive association is observed for TinyLlama-1.1B (Pearson $r = 0.58$, Spearman $\rho = 0.39$), indicating that layers with higher estimated sensitivity tend to receive more conservative precision. The relationship is not perfectly monotonic because the allocator operates at block granularity under a global budget and protection constraints; therefore, individual layers may contain heterogeneous mixtures of 4-, 6-, and 8-bit blocks. Cross-model results are reported in Supplementary Section S4.

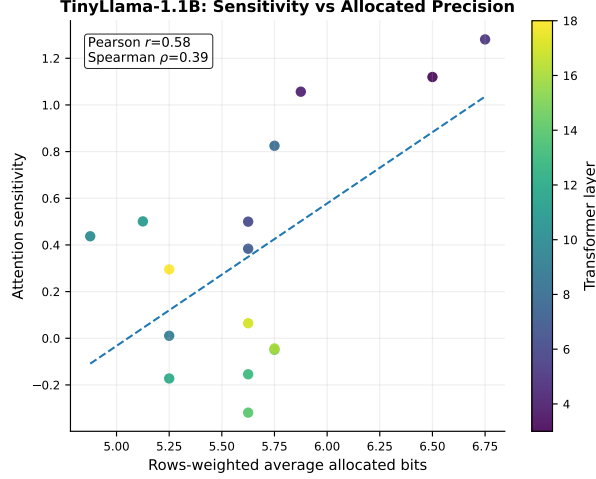


Fig. 6: Relationship between attention-aware sensitivity and rows-weighted average allocated precision for TinyLlama-1.1B. Each point denotes one actively analyzed transformer layer and is colored by layer index. The positive correlation (Pearson $r = 0.58$, Spearman $\rho = 0.39$) indicates that higher-sensitivity layers generally receive higher average precision.

5.3 Efficiency and Perplexity on TinyLlama-1.1B

Table 1 compares perplexity and inference efficiency for TinyLlama-1.1B under the evaluated compression and adaptation configurations. Uniform 4-bit NF4 quantization yields a perplexity of 10.208 and a decoding throughput of 37.09 tok/s. The fixed-rank qLoRA configuration improves perplexity to 8.688, while FP16 with fixed-rank LoRA achieves a PPL of 8.506. The adaptive qLoRA⁺ configuration achieves a PPL of 8.036.

Using the validation-derived allocation, the effect of boundary-layer protection is evaluated by progressively preserving the first and last transformer layers from mixed-bit rewriting. Without boundary protection, the attention-scope configuration achieves a test PPL of 7.899 at 71.20 tok/s. Protecting one and two layers at each model boundary results in PPL values of 7.897 and 7.894, respectively, while protecting three

Table 1: Efficiency and memory comparison on TinyLlama-1.1B. Mixed-bit configurations report different rewrite scopes and adaptive rank settings for qLoRA⁺.

Description	PPL↓	Tok/s↑	ms/token↓	Peak GB↓
FP16 baseline	10.440	62.87	15.91	2.09
NF4 4-bit (bnb4)	10.208	37.09	26.96	0.78
qLoRA (fixed $r = 8$)	8.688	19.60	51.00	0.82
FP16 + LoRA(fixed $r = 8$)	8.506	18.47	54.20	0.83
qLoRA ⁺	8.036	37.05	26.99	0.85
Mixed-bit + LoRA (Attn,3,3)	8.074	23.41	42.70	2.02
Mixed-bit only	8.994	60.60	16.5	2.08
Mixed-bit + qLoRA ⁺ (Attn(0,0))	7.899	71.20	14.0	2.08
Mixed-bit + qLoRA ⁺ (1,1)	7.897	73.27	13.6	2.08
Mixed-bit + qLoRA ⁺ (2,2)	7.894	74.01	13.5	2.08
Mixed-bit + qLoRA ⁺ (3,3)	7.884	73.11	13.7	2.08
Mixed-bit + qLoRA ⁺ (MLP (3,3))	7.863	73.16	13.7	2.08
Mixed-bit + qLoRA ⁺ (Attn+MLP)	7.930	70.65	14.2	2.08

layers at both boundaries yields a PPL of 7.884 at 73.11 tok/s. Although the differences are modest, the consistent improvement in perplexity suggests that preserving the boundary layers provides a small but measurable benefit under the allocation. The three-layer boundary configuration is therefore used as the principal attention-scope setting in the subsequent analysis.

We further examine the effect of rewrite scope under the same validation-derived allocation and three-layer boundary protection. As shown in Table 1, attention-scope rewriting selects the eligible Q and O projections and achieves a test PPL of 7.884 at 73.11 tok/s. Under the MLP scope, the output-dimension eligibility condition selects the `down_proj` modules, yielding a slightly lower PPL of 7.863 at 73.16 tok/s. Rewriting all eligible attention and MLP projections results in a PPL of 7.930 at 70.65 tok/s. Peak GPU memory remains approximately 2.09 GB across these configurations. These results show that rewrite scope has a measurable effect on model quality, whereas the differences in decoding throughput among the configurations are comparatively small. The MLP/`down_proj` configuration achieves the lowest perplexity in this ablation, while the attention configuration is retained as the principal setting for analyzing the attention-derived mixed-precision allocation.

5.4 Efficiency, Memory, and Throughput Trade-offs

Table 2 compares the proposed A²BQ + qLoRA⁺ configuration with FP16, representative PTQ baselines (GPTQ, AWQ, and NF4), qLoRA⁺, and mixed-bit-only configurations across five model families. The results reveal different trade-offs between model quality, runtime performance, and parameter storage, emphasizing that these quantities should be evaluated separately. Across all five models, combining the mixed-bit allocation with qLoRA⁺ consistently improves perplexity relative to mixed-bit

rewriting alone. On TinyLlama-1.1B, perplexity decreases from 8.994 for mixed-bit-only to 7.884 for A²BQ + qLoRA⁺. Similar improvements are observed for Qwen2.5-3B (9.077 to 8.335), Phi-3-mini-4k (7.884 to 6.922), Mistral-7B (6.938 to 6.571), and LLaMA-3.1-8B (8.258 to 7.501). These results indicate that adaptive low-rank fine-tuning provides complementary accuracy recovery after the mixed-precision allocation is applied. The runtime results show an architecture-dependent pattern. Relative to mixed-bit-only rewriting, A²BQ + qLoRA⁺ produces similar decoding throughput on Qwen2.5-3B, Phi-3-mini-4k, Mistral-7B, and LLaMA-3.1-8B, while TinyLlama exhibits a larger increase from 60.60 to 73.11 tok/s. Compared with the FP16 baselines, the proposed configuration achieves higher decoding throughput on four of the five evaluated models, while Qwen2.5-3B shows a small reduction from 35.77 to 34.07 tok/s. However, these measurements correspond to the dequantized-buffer reference backend, which executes standard floating-point linear operations after reconstructing the rewritten weights. The observed runtime differences should therefore be interpreted as implementation-level measurements rather than as evidence of acceleration from native packed low-bit matrix-multiplication kernels.

The storage comparison requires a separate interpretation. GPTQ, AWQ, NF4, and qLoRA⁺ report directly measured parameter payloads, whereas the mixed-bit configurations report the estimated whole-model packed payload implied by the deployed rewrite scope. Because only eligible projections are rewritten in the current implementation, the estimated whole-model payload reduction of A²BQ ranges from 4.02% to 7.90% across the evaluated models. This is smaller than the approximately 56–72% reductions obtained by several globally quantized PTQ configurations, but reflects the actual scope of the proposed mixed-bit rewrite rather than assuming that the entire model is stored at the allocation’s average precision. Mixed-bit-only and A²BQ + qLoRA⁺ have the same estimated packed payload for each model because the learned qLoRA⁺ updates are merged into the base weights before applying the same mixed-bit deployment scope. Consequently, qLoRA⁺ does not introduce a separate adapter payload in the final merged configuration; its primary contribution is the recovery of model quality under the assigned mixed-precision policy.

Table 2: Comprehensive comparison across PTQ, PEFT, and mixed-bit methods.

Configuration	PPL ↓	Tok/s ↑	ms/tok ↓	Peak GB ↓	Parameter Payload (GiB) ↓	Payload Reduction (%) ↑
<i>TinyLlama-1.1B</i>						
Baseline FP16	10.440	62.87	15.91	2.09	2.049	–
GPTQ	11.979	44.91	22.27	1.16	0.715 ^m	65.09%
AWQ	10.905	47.61	21.00	0.76	0.713 ^m	65.20%
NF4	10.208	37.09	26.96	0.78	0.695 ^m	66.06%
qLoRA ⁺	8.036	37.05	26.99	0.85	0.741 ^m	63.83%
Mixed-bit	8.994	60.60	16.5	2.08	1.887 ^e	7.90%
A²BQ + qLoRA⁺	7.884	73.11	13.7	2.08	1.887 ^e	7.90%
<i>Qwen-2.5-3B</i>						
Baseline FP16	10.473	35.77	27.95	5.79	5.748	–
GPTQ	12.366	38.45	26.00	3.26	2.506 ^m	56.39%
AWQ	11.042	22.72	44.01	1.97	1.923 ^m	66.55%

Configuration	PPL ↓	Tok/s ↑	ms/tok ↓	Peak GB ↓	Parameter Payload (GiB) ↓	Payload Reduction (%) ↑
NF4	11.131	21.97	45.51	2.00	1.872 ^m	67.43%
qLoRA ⁺	8.519	18.29	54.67	2.18	1.980 ^m	65.55%
Mixed-bit	9.077	33.79	29.6	5.79	5.448 ^e	5.22%
A²BQ + qLoRA⁺	8.335	34.07	29.3	5.79	5.448 ^e	5.22%
<i>Phi-3-mini-4k</i>						
Baseline FP16	7.832	29.18	34.27	7.21	7.117	—
GPTQ	N/A	N/A	N/A	N/A	N/A	N/A
AWQ	8.317	21.39	46.74	2.22	2.121 ^m	70.20%
NF4	8.528	20.58	48.60	2.25	2.054 ^m	71.13%
qLoRA ⁺	7.238	24.40	41.28	2.52	2.147 ^m	69.83%
Mixed-bit	7.884	37.05	27.0	7.21	6.831 ^e	4.02%
A²BQ + qLoRA⁺	6.922	36.65	27.3	7.21	6.831 ^e	4.02%
<i>Mistral-7B</i>						
Baseline FP16	6.894	25.79	38.78	13.54	13.489	—
GPTQ	7.335	30.02	33.31	5.44	3.873 ^m	71.29%
AWQ	7.053	24.61	40.64	3.92	3.865 ^m	71.34%
NF4	7.151	26.61	37.57	4.00	3.739 ^m	72.28%
qLoRA ⁺	6.749	15.53	64.38	4.36	3.890 ^m	71.16%
Mixed-bit	6.938	27.92	35.8	13.54	12.436 ^e	7.80%
A²BQ + qLoRA⁺	6.571	28.28	35.4	13.54	12.436 ^e	7.80%
<i>LLaMA-3.1-8B</i>						
Baseline FP16	8.090	24.67	40.53	15.01	14.958	—
GPTQ	N/A	N/A	N/A	N/A	N/A	N/A
AWQ	8.602	17.67	56.58	5.39	5.334 ^m	64.34%
NF4	8.746	20.18	49.55	5.47	5.207 ^m	65.18%
qLoRA ⁺	7.733	15.39	65.0	5.83	5.360 ^m	64.16%
Mixed-bit	8.258	27.42	36.5	15.01	13.914 ^e	6.97%
A²BQ + qLoRA⁺	7.501	27.17	36.8	15.01	13.914 ^e	6.97%

Note: ^m denotes directly measured parameter payload, whereas ^e denotes the estimated whole-model packed-weight payload under the deployed mixed-bit rewrite scope. FP16 denotes the uncompressed reference payload used to compute payload reduction. GPTQ is reported as N/A for Phi-3-mini-4k because AutoGPTQ 0.7.1 does not natively support the Phi-3 architecture, and for LLaMA-3.1-8B because the kernel-compatible GPTQ environment does not support its newer rotary-scaling configuration. Unsupported cases are reported as N/A rather than using architecture-specific modifications solely to obtain baseline values.

The results further illustrate why parameter payload and runtime GPU memory should not be interpreted interchangeably. For the proposed mixed-bit configurations, the estimated packed payload decreases by 4.02–7.90% relative to the corresponding FP16 representation, whereas the measured peak GPU memory remains close to the FP16 runtime footprint. This behavior is consistent with the reference implementation: rewritten weights are reconstructed into floating-point buffers before inference, so the theoretical packed representation is not physically resident during the measured forward pass. Accordingly, the payload reduction reported for A²BQ represents the storage potential of the learned mixed-precision assignment under physical packing, while the Peak GB column characterizes the actual DQ-backend runtime.

Overall, Table 2 shows that the main empirical benefit of combining A²BQ with qLoRA⁺ under the current rewrite scope is improved model quality without materially changing the estimated packed payload. The combined configuration achieves lower perplexity than mixed-bit-only rewriting across all five model families while generally retaining comparable decoding performance. At the same time, globally quantized PTQ baselines provide substantially larger physical parameter compression because they quantize a greater fraction of the model. These complementary results distinguish the quality-recovery contribution of sensitivity-aligned adaptation from the storage and runtime characteristics of the current reference backend.

5.5 Downstream Task Performance

As shown in Table 3, downstream task performance across five model families remains generally close to the corresponding FP16 baselines on MMLU, GSM8K, TruthfulQA, and ARC-Challenge. TinyLlama-1.1B shows a slight improvement on MMLU (0.2480 vs. 0.2496) and TruthfulQA (0.3700 vs. 0.3906), while Qwen2.5-3B improves on MMLU (0.6545 vs. 0.6567) and ARC-Challenge (0.4590 vs. 0.4863), with only a small decrease on GSM8K (0.6657 vs. 0.6596). Phi-3-mini-4k exhibits highly stable performance across all four tasks, including an unchanged GSM8K score of 0.7900. Mistral-7B and LLaMA-3.1-8B show somewhat larger but still moderate decreases on GSM8K and TruthfulQA, while MMLU and ARC-Challenge remain largely stable. Overall, these results demonstrate that A²BQ + qLoRA⁺ largely preserves downstream task performance under the fixed bit budget across diverse model architectures and evaluation benchmarks.

Table 3: Task performance comparing FP16 baselines against A²BQ + qLoRA⁺.

Model	Config	MMLU (0-shot)	GSM8K (8-shot)	TruthfulQA (0-shot, MC2)	ARC-C (0-shot)
TinyLlama-1.1B	FP16	0.2480	0.0273	0.3700	0.3268
TinyLlama-1.1B	A ² BQ+qLoRA ⁺	0.2496	0.0235	0.3906	0.3242
Qwen2.5-3B	FP16	0.6545	0.6657	0.5867	0.4590
Qwen2.5-3B	A ² BQ+qLoRA ⁺	0.6567	0.6596	0.5847	0.4863
Phi-3-mini-4k	FP16	0.7016	0.7900	0.5447	0.5648
Phi-3-mini-4k	A ² BQ+qLoRA ⁺	0.7008	0.7900	0.5445	0.5640
Mistral-7B	FP16	0.5983	0.4928	0.5969	0.6007
Mistral-7B	A ² BQ+qLoRA ⁺	0.5979	0.4769	0.5829	0.6007
LLaMA-3.1-8B	FP16	0.6844	0.7885	0.5403	0.5572
LLaMA-3.1-8B	A ² BQ+qLoRA ⁺	0.6837	0.7801	0.5273	0.5555

5.6 Layer-wise Distribution Shift Under Quantization

To understand how mixed-bit quantization affects internal model representations, we analyze the layer-wise KL divergence between the FP16 model and the quantized

model. In this analysis, mixed-bit rewriting is applied only to attention projections, while MLP projections remain in full precision. Fig 7 reveals a strongly non-uniform depth-dependent KL profile for TinyLlama-1.1B. Divergence is small in the protected and early portions of the network and becomes more pronounced within several intermediate and deeper active layers, although the profile is not strictly monotonic. As shown in Supplementary Section S2 the precise depth-dependent pattern varies across architectures.

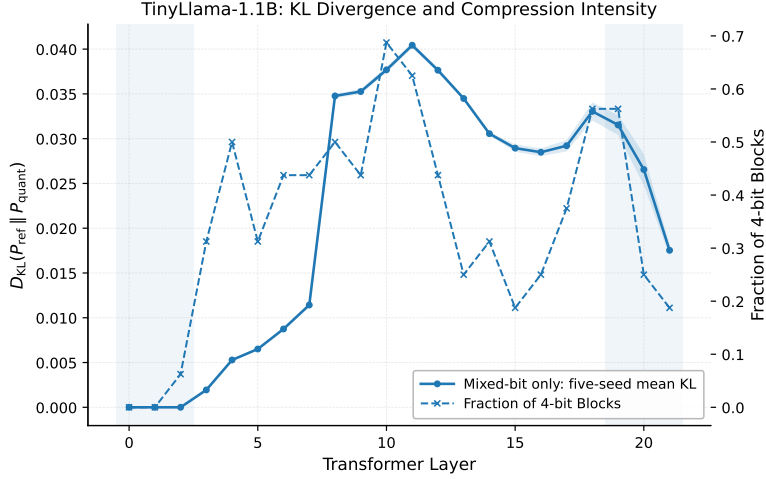


Fig. 7: Layer-wise KL divergence under attention-only mixed-bit quantization, shown alongside the fraction of 4-bit blocks per layer (compression intensity). Distributional divergence is non-uniform across transformer depth, with larger shifts observed in several intermediate and deeper active layers.

An important question is whether the attention-derived sensitivity signal alone explains where this divergence arises. For TinyLlama-1.1B, Fig 8 shows a moderate negative Pearson association between attention-aware sensitivity and quantization-induced KL divergence ($r = -0.667$), whereas the rank-based Spearman correlation is weaker ($\rho = -0.400$, $n = 16$ active layers). Thus, the relationship is not a simple monotonic correspondence: layers assigned high sensitivity do not necessarily exhibit the largest predictive-distribution shift after quantization. The cross-model analysis in Supplementary Section S3 further shows that both the magnitude and sign of this relationship are architecture dependent. We therefore interpret attention-derived sensitivity primarily as an allocation signal rather than as a direct predictor of post-quantization KL divergence.

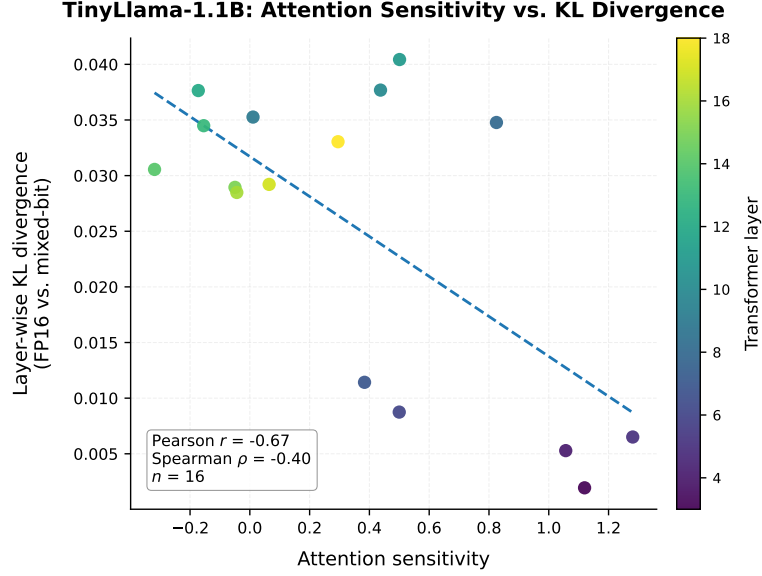


Fig. 8: Attention-aware sensitivity versus quantization-induced layer-wise KL divergence for TinyLlama-1.1B over the 16 active transformer layers. The dashed line shows the linear fit; Pearson $r = -0.667$ and Spearman $\rho = -0.400$. Point color indicates transformer layer.

To better account for this behavior, we examine KL divergence as a function of compression intensity, measured through the average bit-width assigned to each layer. Fig 9 jointly visualizes the layer-wise KL profile and effective allocated precision. Their relationship is depth dependent rather than strictly monotonic, indicating that quantization-induced divergence reflects both the location of compression and accumulated transformations across the network.

5.7 Theoretical Packed Payload and Runtime Memory

We distinguish between the storage reduction implied by the mixed-precision allocation and the physical memory footprint of the reference inference backend, as these quantities are not equivalent in the current implementation.

For the principal TinyLlama-1.1B configuration, mixed-bit rewriting is applied to the Q and O attention projections in transformer layers 3–18, while the first and last three layers remain unrevised. Because TinyLlama employs grouped-query attention, the K and V projections have output dimensions smaller than the model hidden size and are therefore excluded by the current rewrite eligibility condition. The rewritten Q/O projections contain approximately 134.2M parameters, corresponding to 12.20% of the complete model.

Using the exact block-wise bit assignments for the rewritten projections, the estimated whole-model parameter payload under ideal physical bit packing is 1.887 GiB,

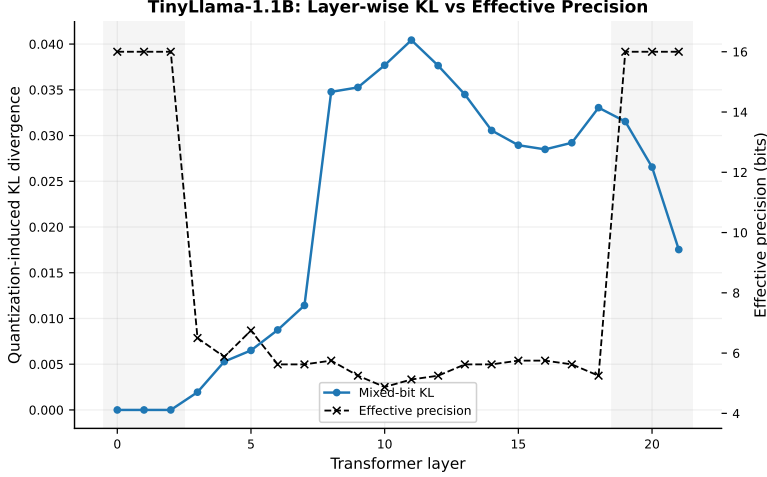


Fig. 9: Layer-wise quantization-induced KL divergence and effective allocated precision for TinyLlama-1.1B. The figure visualizes how distributional shift co-varies with the precision pattern across transformer depth; protected boundary layers remain at full precision.

compared with 2.049 GiB for an FP16 parameter representation, corresponding to a theoretical whole-model storage reduction of 7.90%. The reduction is limited because only a subset of model parameters is eligible for mixed-bit rewriting, while the remaining parameters retain their original precision. Fig 10 further shows the layer-wise theoretical packed-weight reduction across the evaluated model architectures.

The reference runtime backend does not execute directly on the theoretically packed representation. Instead, the selected mixed-bit weights are reconstructed once into floating-point dequantized buffers, after which inference uses standard floating-point linear operations. Consequently, the theoretical packed payload should not be interpreted as the resident GPU memory footprint of the current implementation. For the principal TinyLlama configuration, the measured peak GPU memory is approximately 2.08 GB. We therefore report the theoretical packed parameter payload and measured peak runtime memory as separate quantities, with the former characterizing the storage potential of the learned allocation and the latter reflecting the memory behavior of the implemented inference backend.

6 Ablation Study

We ablate the choice of bit allocator at a fixed budget (avg bits ≈ 5.7) on TinyLlama-1.1B, holding scope (attention-only) and keep (3 + 3) constant. Table 4 reports perplexity and decoding performance across four baselines Depth-only, Random, Weight-Norm, and Weight-Var, as well as alternative sensitivity definitions. In addition to the proposed attention-based sensitivity used in A²BQ, we evaluate attention-only mixed-bit allocation without adaptive rank, the attention-only mixed-bit baseline lags

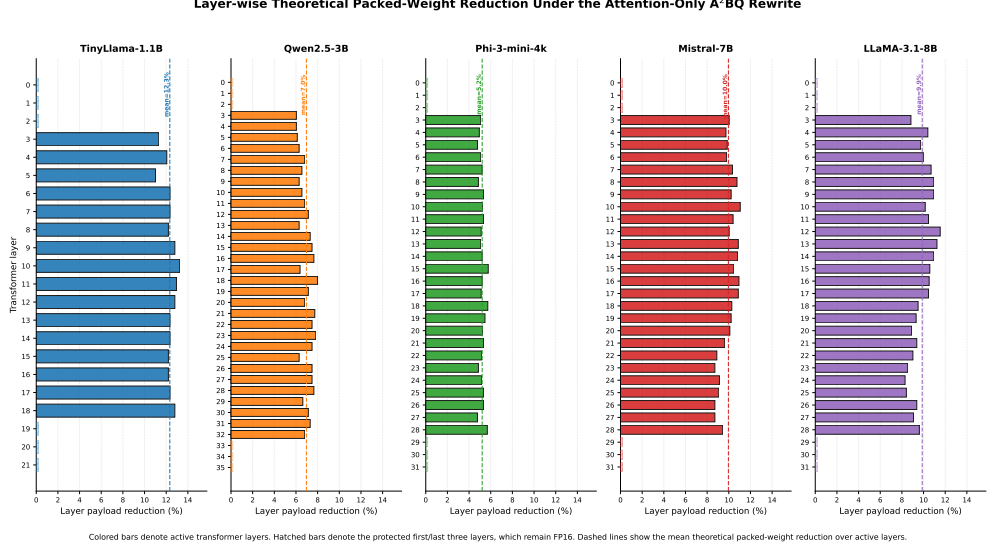


Fig. 10: Layer-wise theoretical packed-weight reduction under the attention-only A²BQ rewrite across the evaluated model architectures.

Table 4: Allocator ablation on TinyLlama-1.1B at fixed average bit budget (5.7 ± 0.05), attention-only scope, keep=3+3. Values are mean $\pm$ std where $n > 1$; Depth-only $n=2$, Random $n=5$, others $n=1$.

Allocator	PPL $\downarrow$	Tok/s $\uparrow$	ms/tok $\downarrow$	Peak GB $\downarrow$
Depth-only	8.006 ± 0.003	54.165 ± 23.815	22.850 ± 10.050	2.080
Random	8.029 ± 0.005	23.944 ± 1.013	41.840 ± 1.785	2.030
Weight-Norm	8.006	30.600	32.700	2.080
Weight-Var	8.006	33.540	29.800	2.080
Mixed-bit (Attn. Sensitivity)	8.994	60.60	16.5	2.08
Sensitivity (A²BQ) + qLoRA⁺	7.884	73.11	13.7	2.08

behind (PPL 8.994). In contrast, A²BQ + qLoRA⁺ achieves the best perplexity (7.884) and highest throughput (73.11 tok/s) at comparable peak VRAM (~ 2.04 –2.09 GB). These results demonstrate that the proposed sensitivity-aware allocation, when combined with adaptive rank allocation, provides a more effective strategy for preserving model quality and decoding efficiency under a fixed mixed-bit precision budget.

6.1 Impact of Quantization-aware Adaptation (qLoRA⁺)

We isolate the effect of quantization-aware adaptation by comparing mixed-bit inference with and without qLoRA⁺, while keeping the mixed-bit allocation fixed. To quantify how adaptation alters internal representations, we measure the *layer-wise KL divergence between output token distributions* produced by the two models. For each layer, hidden states are projected through the shared final normalization and language modeling head, and KL divergence is computed between the resulting distributions.

Fig 11 reports the per-layer change in KL divergence between the mixed-bit-only model and the mixed-bit + qLoRA⁺ model. Larger positive ΔKL values indicate that qLoRA⁺ reduces the quantization-induced distributional divergence, whereas negative values indicate increased divergence after adaptation. For TinyLlama, the ΔKL profile becomes predominantly positive in the middle and deeper active layers, indicating that qLoRA⁺ reduces predictive-distribution divergence in those regions. A small number of early active layers show negative ΔKL , demonstrating that the recovery is not uniform across depth. This behavior suggests that qLoRA⁺ provides greater representational recovery in deeper portions of the actively rewritten network, while its effect varies across individual layers depending on their sensitivity to quantization.

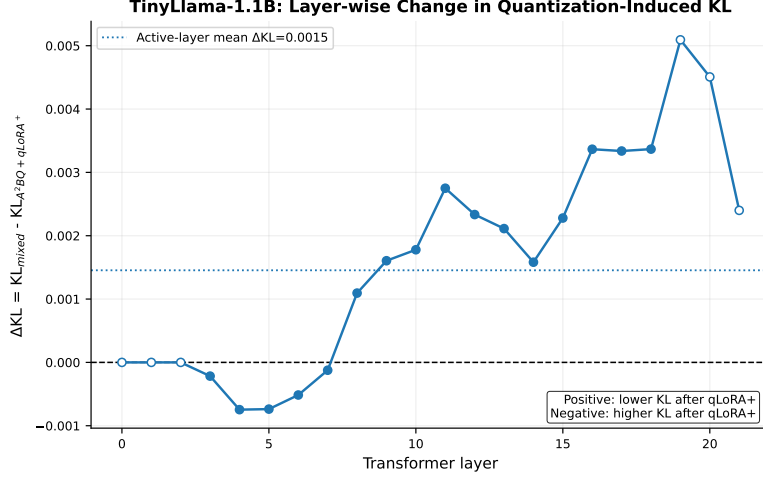


Fig. 11: Per-layer change in quantization-induced KL divergence, defined as $\Delta KL = KL_{\text{mixed}} - KL_{A^2BQ+qLoRA^+}$. Positive values indicate lower divergence after qLoRA⁺ adaptation, whereas negative values indicate higher divergence. The dotted line denotes the mean over actively rewritten layers.

The predominance of positive ΔKL values in the middle and deeper active layers is consistent with the earlier observation that quantization pressure and distributional fragility are more pronounced in later layers Fig 7. Rather than indicating a strictly monotonic improvement with depth, the results suggest that qLoRA⁺ provides greater distributional recovery in layers that are more sensitive to quantization, while its effect remains layer-dependent. This behavior is consistent with the motivation for allocating greater adaptation capacity to more compressed and sensitive regions.

6.2 Block Size

We evaluate row-block sizes $b \in \{64, 128, 256\}$ while keeping the allocator target fixed at approximately 5.7 global average bits and using the same attention-Q/O rewrite scope with three protected layers at each model boundary. The resulting global average

precisions remain closely matched (5.6989, 5.6989, and 5.6932 bits for $b = 64, 128, 256$, respectively). However, because the boundary layers are excluded from active rewriting, the average precision within the active region differs across configurations, yielding 5.9844 bits for $b = 64$, 5.6406 bits for $b = 128$, and 6.0313 bits for $b = 256$, respectively.

The intermediate block size $b = 128$ obtains the lowest test perplexity (7.884), compared with 8.281 for $b = 64$ and 8.290 for $b = 256$. It also provides the highest measured decode throughput among the evaluated settings (73.11 tok/s versus 70.75 and 71.10 tok/s for $b = 64$ and $b = 256$, respectively). While peak GPU memory remains approximately 2.09 GB in all cases. Notably, $b = 128$ achieves the best perplexity despite having the lowest active-region average precision, indicating that allocation granularity and the placement of precision are more important than simply increasing the average number of bits assigned to active layers.

These results suggest that very fine blocks do not necessarily improve model quality, while coarse blocks reduce the allocator’s ability to distinguish rows with different sensitivity. Under the evaluated TinyLlama configuration, $b = 128$ provides the most favorable quality–throughput trade-off and is therefore retained as the default block size. Because the active-region average precision is not identical across the three settings, this experiment should be interpreted as an ablation of the complete block-size-dependent allocation pipeline under a fixed global budget, rather than as a comparison in which the active quantized payload is exactly matched.

6.3 Rewrite Scope and Reference Backend

The rewrite implementation applies mixed-precision quantization only to linear modules satisfying both the requested rewrite scope and an output-dimension eligibility condition. For TinyLlama, under the attention scope, this condition results in rewriting the Q and O projections. The grouped-query K and V projections are excluded because their output dimensions differ from the model hidden size. Under the MLP scope, the same eligibility condition selects the `down_proj` modules rather than all three MLP projections.

With the validation-derived allocation and three protected layers at both model boundaries, the attention Q/O configuration achieves a test PPL of 7.884, a throughput of 73.11 tok/s, and a peak GPU memory usage of 2.09 GB. The corresponding MLP `down_proj` configuration achieves a PPL of 7.863 and a throughput of 73.16 tok/s, while rewriting all eligible projections results in a PPL of 7.930 and a throughput of 70.65 tok/s. These results indicate that the rewrite scope influences model quality, whereas the differences in decoding throughput among these configurations are comparatively small.

The reference backend used in these experiments employs dequantized-buffer (DQ) execution. The assigned mixed-precision weights are reconstructed once into floating-point buffers before inference, after which the forward computation is performed using standard `F.linear` operations. Thus, the current backend evaluates the numerical effects of the learned mixed-precision allocation but does not implement physically packed low-bit matrix multiplication during inference. Accordingly, the measured runtime behavior should not be interpreted as acceleration arising from reduced

packed-weight memory traffic. Native integer-packed kernels and hardware-specific mixed-precision execution are left for future implementation.

7 Discussion

This study investigates how block-wise mixed-precision quantization and parameter-efficient adaptation can be coordinated to account for heterogeneous quantization sensitivity across transformer models. Rather than treating quantization and fine-tuning as independent stages, the proposed framework first allocates 4-, 6-, and 8-bit precision according to block-wise sensitivity and then aligns low-rank adaptation capacity with both sensitivity and quantization pressure. The experimental results show that this combination consistently improves model quality relative to mixed-bit rewriting alone across the evaluated model families, while generally retaining comparable inference behavior under the same reference execution backend.

A central observation is that quantization-induced perturbations are non-uniform across transformer depth. The layer-wise KL analysis shows that distributional divergence can vary substantially across depth and architecture, with several models exhibiting stronger divergence in deeper layers, while the precise profile remains model dependent [7, 28]. At the same time, the weak and model-dependent correlation between attention-derived sensitivity and KL divergence indicates that these quantities capture different aspects of quantization behavior. Attention-derived sensitivity is therefore better interpreted as an allocation signal rather than as a complete predictor of quantization-induced distribution shift. This observation supports the use of complementary information, including compression intensity and adaptive recovery capacity, when constructing mixed-precision configurations.

The correlation analysis further separates two related but distinct questions. First, sensitivity is positively associated with the precision ultimately assigned by the allocator, confirming that A²BQ behaves according to its intended design: blocks with higher estimated sensitivity tend to receive more conservative precision. Second, the relationship between sensitivity and post-quantization KL divergence is weaker and architecture dependent. These observations are not contradictory because the sensitivity score is designed to guide resource allocation rather than to serve as a calibrated estimator of distributional divergence. The allocator can therefore consistently prioritize high-sensitivity regions even when the resulting KL response depends on model depth, architecture, and accumulated quantization effects.

The A²BQ allocator produces structured mixtures of 4-, 6-, and 8-bit blocks while satisfying a global precision budget, with the exact allocation pattern varying across architectures. Similar inter-layer variability is observed across models ranging from 1.1B to 8B parameters, indicating that heterogeneous precision requirements are not specific to a single backbone. However, the practical impact of the allocation also depends on the rewrite scope. In the current implementation, only projections satisfying the scope and output-dimension eligibility conditions are rewritten. For example, in the principal TinyLlama attention configuration, Q and O projections are rewritten whereas the grouped-query K and V projections are excluded. The scope of ablation further shows that attention Q/O, MLP `down_proj`, and the combined eligible scope

yield different perplexity values while exhibiting comparatively small differences in decoding throughput.

Quantization-aware low-rank adaptation provides a complementary source of recovery. Across all five evaluated model families, $A^2BQ + qLoRA^+$ achieves lower perplexity than mixed-bit rewriting alone under the same mixed-bit deployment scope. This behavior indicates that adaptive rank allocation can recover part of the quality affected by mixed-precision rewriting without requiring a separate adapter payload in the final merged configuration. However, the cross-model KL analysis shows that $qLoRA^+$ does not universally reduce quantization-induced distributional divergence. Lower mean KL divergence is observed for TinyLlama, Qwen2.5-3B, Phi-3-mini-4k, and LLaMA-3.1-8B, whereas Mistral-7B exhibits increased KL divergence despite achieving improved perplexity after adaptation. This result further demonstrates that proximity to the FP16 predictive distribution is not itself a sufficient indicator of downstream model quality. The benefit of $qLoRA^+$ should therefore be interpreted primarily as quality recovery under a fixed mixed-precision policy, with changes in internal distributional divergence remaining architecture dependent rather than serving as a universal proxy for task performance.

An important practical finding is the distinction between theoretical packed parameter storage and measured runtime GPU memory. For the proposed mixed-bit configurations, the scope-correct estimated whole-model packed-payload reduction ranges from approximately 4.02% to 7.90% across the evaluated architectures. These values are smaller than those of globally quantized 4-bit baselines because only a restricted subset of eligible projections is rewritten in the current implementation. Moreover, the reference backend reconstructs the rewritten weights into floating-point dequantized buffers before inference and executes standard floating-point linear operations. Consequently, the estimated packed payload does not represent the resident runtime representation, and the measured throughput should not be attributed to reduced packed-weight memory traffic. This distinction highlights the importance of reporting storage potential and runtime behavior as separate system characteristics when evaluating mixed-precision LLM methods.

7.1 Limitations and Future Work

The present work has several limitations. First, the current rewrite procedure operates on a restricted subset of model projections rather than quantizing the complete network. Module selection is determined by the requested rewrite scope together with an output-dimension eligibility condition. In architectures using grouped-query attention, this condition can exclude K and V projections, while the MLP scope may select only specific projections such as `down_proj`. Consequently, the achievable whole-model packed-payload reduction is limited by the fraction of parameters that are actually rewritten. Extending the allocation and rewrite procedure to a broader set of projections is therefore an important direction for future work.

Second, the reference implementation does not execute directly from a physically packed 4/6/8-bit weight representation. Instead, mixed-bit weights are reconstructed into floating-point buffers before inference, after which standard floating-point linear operations are used. Accordingly, the packed-payload reductions reported in this work

represent theoretical storage estimates for the learned allocation rather than measured resident model sizes during inference. Developing native packed mixed-precision kernels would enable direct evaluation of physical storage reduction, memory-bandwidth behavior, and hardware-level inference efficiency.

Third, attention-derived sensitivity provides a lightweight signal for precision allocation, but it does not fully explain quantization-induced distribution shift. The observed weak and architecture-dependent relationship between attention-based sensitivity and layer-wise KL divergence indicates that no single sensitivity proxy captures all aspects of quantization fragility. Moreover, the cross-model qLoRA⁺ results show that reductions in KL divergence are not universally coupled with improvements in perplexity, as demonstrated by Mistral-7B, where perplexity improves despite increased mean KL divergence after adaptation. Future work could investigate gradient-, activation-, or hybrid sensitivity measures, as well as depth-aware and module-specific constraints. Importantly, such alternatives should be evaluated not only according to their correlation with internal distributional changes but also by their effect on downstream model quality.

Finally, the current experiments focus on a fixed set of model families, datasets, and hardware conditions. Although the results show consistent quality recovery across models from 1.1B to 8B parameters, broader evaluation on additional architectures, longer-context workloads, and native low-bit inference backends would provide a more complete assessment of the framework’s generality and system-level benefits.

8 Conclusion

This work presented A²BQ, a sensitivity-aware block-wise mixed-precision quantization framework, together with qLoRA⁺, an adaptive low-rank fine-tuning strategy for quantized LLMs. The framework departs from uniform precision and uniform adaptation by assigning 4-, 6-, and 8-bit precision at block granularity and varying adaptation capacity according to sensitivity and quantization pressure.

Experiments across five model families ranging from 1.1B to 8B parameters show that combining A²BQ with qLoRA⁺ consistently improves perplexity relative to mixed-bit rewriting alone while generally preserving comparable decoding performance. The evaluation protocol separates training, validation-based sensitivity estimation and allocation, and final held-out test evaluation, ensuring that the reported perplexity results are obtained without test-set information being used during allocation. Layer-wise analysis further shows that quantization-induced distribution shift varies substantially across depth, with depth-dependent profiles that often become more pronounced within intermediate and deeper active layers, while adaptive low-rank tuning provides complementary recovery in affected regions.

The system-level analysis also demonstrates the importance of distinguishing algorithmic compression potential from runtime implementation behavior. Under the actual rewrite scopes used in the experiments, the estimated whole-model packed-payload reduction ranges from approximately 4.02% to 7.90%. These values represent theoretical storage requirements under physical packing and should not be interpreted

as measured runtime memory reductions, because the current dequantized-buffer backend reconstructs rewritten weights into floating-point buffers before inference.

Overall, the results support sensitivity-aligned coordination of mixed-precision allocation and adaptive low-rank tuning as a useful approach for improving model quality under structured precision constraints. Extending the framework to broader rewrite scopes and native packed mixed-precision execution represents a natural next step toward realizing its full storage and hardware-efficiency potential.

Declarations

Acknowledgements & Funding. This research was supported by Global-Learning & Academic research institution for Master’s-PhD students, and Postdocs(G-LAMP) Program of the National Research Foundation of Korea(NRF) grant funded by the Ministry of Education (RS-2026-25610465). This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2026-RS-2026-25615817, AI Star Fellowship Support Program).

Conflict of interests. The authors declare that they have no competing interests.

Data availability. The datasets used in this study are publicly available. Wikitext-2-raw-v1 is accessible through the HuggingFace datasets library (<https://huggingface.co/datasets/Salesforce/wikitext>). All models evaluated are obtained from their respective official open-source repositories. No new datasets were generated during this study.

Author contributions. The authors conceived the study, developed the proposed methodology, conducted the experiments, analyzed the results, and wrote the manuscript.

References

- [1] Dhar, N., Deng, B., Lo, D., Wu, X., Zhao, L., Suo, K.: An empirical analysis and resource footprint study of deploying large language models on edge devices. In: Proceedings of the 2024 ACM Southeast Conference, pp. 69–76 (2024)
- [2] Yu, Z., Wang, Z., Li, Y., Gao, R., Zhou, X., Bommur, S.R., Zhao, Y., Lin, Y.: Edge-llm: Enabling efficient large language model adaptation on edge devices via unified compression and adaptive layer voting. In: Proceedings of the 61st ACM/IEEE Design Automation Conference, pp. 1–6 (2024)
- [3] Hu, M., He, Q., Wu, D.: Qllms: Quantization-adaptive llm scheduling for partially informed edge serving systems. In: IEEE INFOCOM 2025-IEEE Conference on Computer Communications, pp. 1–10 (2025). IEEE
- [4] Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: Gptq: Accurate post-training quantization for generative pre-trained transformers. arXiv preprint arXiv:2210.17323 (2022)
- [5] Yao, Z., Yazdani Aminabadi, R., Zhang, M., Wu, X., Li, C., He, Y.: Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems* **35**, 27168–27183 (2022)
- [6] Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L.: Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems* **35**, 30318–30332 (2022)
- [7] Roy, S.: Understanding the impact of post-training quantization on large language models. arXiv preprint arXiv:2309.05210 (2023)
- [8] Wu, X., Li, C., Aminabadi, R.Y., Yao, Z., He, Y.: Understanding int4 quantization for language models: Latency speedup, composability, and failure cases. In: International Conference on Machine Learning, pp. 37524–37539 (2023). PMLR
- [9] Liu, Y., Wen, J., Wang, Y., Ye, S., Zhang, L.L., Cao, T., Li, C., Yang, M.: Vptq: Extreme low-bit vector post-training quantization for large language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 8181–8196 (2024)
- [10] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., Han, S.: Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems* **6**, 87–100 (2024)
- [11] Dettmers, T., Svirschevski, R.A., Egiastian, V., Kuznedelev, D., Frantar, E., Ashkboos, S., Borzunov, A., Hoefler, T., Alistarh, D.: Spqr: A sparse-quantized representation for near-lossless llm weight compression. In: The Twelfth

- [12] Kim, Y.J., Henry, R., Fahim, R., Awadalla, H.H.: Finequant: Unlocking efficiency with fine-grained weight-only quantization for llms. CoRR (2023)
- [13] Adepu, H., Zeng, Z., Zhang, L., Singh, V.: Framequant: Flexible low-bit quantization for transformers. In: International Conference on Machine Learning, pp. 203–227 (2024). PMLR
- [14] Chmiel, B., Fishman, M., Banner, R., Soudry, D.: Fp4 all the way: Fully quantized training of large language models. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
- [15] Chen, S., Khisti, A.J.: Adaquantlm: Llm quantization with adaptive bit-widths. In: Workshop on Machine Learning and Compression (NeurIPS 2024)
- [16] Li, Y., Yu, Y., Liang, C., Karampatziakis, N., He, P., Chen, W., Zhao, T.: Loftq: Lora-fine-tuning-aware quantization for large language models. In: ICLR (2024)
- [17] Li, Q., Zhang, Y., Li, L., Yao, P., Zhang, B., Chu, X., Sun, Y., Du, L., Xie, Y.: Fptq: Fine-grained post-training quantization for large language models. arXiv preprint arXiv:2308.15987 (2023)
- [18] Dumitru, R.-G., Yadav, V., Maheshwary, R., Clotan, P.I., Madhusudhan, S.T., Surdeanu, M.: Variable layerwise quantization: A simple and effective approach to quantize llms. In: Findings of the Association for Computational Linguistics: ACL 2025, pp. 534–550 (2025)
- [19] Liu, Z., Zhao, C., Huang, H., Chen, S., Zhang, J., Zhao, J., Roy, S., Jin, L., Xiong, Y., Shi, Y., et al.: Paretoq: Scaling laws in extremely low-bit llm quantization. arXiv preprint arXiv:2502.02631 (2025)
- [20] Guan, Z., Huang, H., Su, Y., Huang, H., Wong, N., Yu, H.: Aptq: Attention-aware post-training mixed-precision quantization for large language models. In: Proceedings of the 61st ACM/IEEE Design Automation Conference, pp. 1–6 (2024)
- [21] Chen, Z., Xie, B., Li, J., Shen, C.: Channel-wise mixed-precision quantization for large language models. arXiv preprint arXiv:2410.13056 (2024)
- [22] Zheng, Z., Song, X., Liu, C.: Mixllm: Llm quantization with global mixed-precision between output-features and highly-efficient system design. arXiv preprint arXiv:2412.14590 (2024)
- [23] Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L.: Qlora: Efficient fine-tuning of quantized llms. *Advances in neural information processing systems* **36**, 10088–10115 (2023)

- [24] Xu, Y., Xie, L., Gu, X., Chen, X., Chang, H., Zhang, H., Chen, Z., Zhang, X., Tian, Q.: Qa-lora: Quantization-aware low-rank adaptation of large language models. In: ICLR (2024)
- [25] Lee, C., Jin, J.-g., Cho, Y., Park, E.: Qeft: Quantization for efficient fine-tuning of llms. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp. 13823–13837 (2024)
- [26] Qin, H., Ma, X., Zheng, X., Li, X., Zhang, Y., Liu, S., Luo, J., Liu, X., Magno, M.: Accurate lora-finetuning quantization of llms via information retention. In: Proceedings of the 41st International Conference on Machine Learning, pp. 41498–41516 (2024)
- [27] Lang, J., Guo, Z., Huang, S.: A comprehensive study on quantization techniques for large language models. In: 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC), pp. 224–231 (2024). IEEE
- [28] Bondarenko, Y., Nagel, M., Blankevoort, T.: Understanding and overcoming the challenges of efficient transformer quantization. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 7947–7969 (2021)
- [29] Kurtic, E., Marques, A.N., Pandit, S., Kurtz, M., Alistarh, D.: “Give me BF16 or give me death”? Accuracy-performance trade-offs in llm quantization. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 26872–26886 (2025)
- [30] Shi, T., Ding, Y.: Systematic characterization of llm quantization: A performance, energy, and quality perspective. arXiv preprint arXiv:2508.16712 (2025)
- [31] Malinovskii, V., Panferov, A., Ilin, I., Guo, H., Richtárik, P., Alistarh, D.: Higgs: Pushing the limits of large language model quantization via the linearity theorem. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 10857–10886 (2025)
- [32] Wei, X., Gong, R., Li, Y., Liu, X., Yu, F.: Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. In: International Conference on Learning Representations (2022)
- [33] Huang, W., Liu, Y., Qin, H., Li, Y., Zhang, S., Liu, X., Magno, M., Qi, X.: Billm: Pushing the limit of post-training quantization for llms. In: International Conference on Machine Learning, pp. 20023–20042 (2024). PMLR
- [34] Zhang, F., Liu, Y., Li, W., Wang, X., Bai, Q.: A mixed quantization approach for data-free quantization of llms. In: ICAART (2), pp. 353–363 (2025)

- [35] Guo, J., Wu, J., Wang, Z., Liu, J., Yang, G., Ding, Y., Gong, R., Qin, H., Liu, X.: Compressing large language models by joint sparsification and quantization. In: Forty-first International Conference on Machine Learning (2024)
- [36] Gong, Z., Liu, J., Wang, Q., Yang, Y., Wang, J., Wu, W., Xian, Y., Zhao, D., Yan, R.: Prequant: A task-agnostic quantization approach for pre-trained language models. In: Findings of the Association for Computational Linguistics: ACL 2023, pp. 8065–8079 (2023)
- [37] Shao, W., Chen, M., Zhang, Z., Xu, P., Zhao, L., Li, Z., Zhang, K., Gao, P., Qiao, Y., Luo, P.: Omniquant: Omnidirectionally calibrated quantization for large language models. In: The Twelfth International Conference on Learning Representations (2024)
- [38] Ding, X., Liu, X., Tu, Z., Zhang, Y., Li, W., Hu, J., Chen, H., Tang, Y., Xiong, Z., Yin, B., *et al.*: Cbq: Cross-block quantization for large language models. In: ICLR (2025)
- [39] Czakó, P., Kertész, G., Szénási, S.: Addressing activation outliers in llms: A systematic review of post-training quantization techniques. IEEE Access (2025)
- [40] Gong, Z., Liu, J., Wang, J., Cai, X., Zhao, D., Yan, R.: What makes quantization for large language model hard? An empirical study from the lens of perturbation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 18082–18089 (2024)
- [41] Heo, J.H., Kim, J., Kwon, B., Kim, B., Kwon, S.J., Lee, D.: Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models. In: ICLR (2024)
- [42] Chen, M., Shao, W., Xu, P., Wang, J., Gao, P., Zhang, K., Luo, P.: 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), pp. 10081–10100 (2025)
- [43] Lin, Y., Tang, H., Yang, S., Zhang, Z., Xiao, G., Gan, C., Han, S.: Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. Proceedings of Machine Learning and Systems **7** (2025)
- [44] Kang, H., Zhang, Q., Kundu, S., Jeong, G., Liu, Z., Krishna, T., Zhao, T.: Gear: An efficient error reduction framework for kv cache compression in llm inference. In: NeurIPS Efficient Natural Language and Speech Processing Workshop, pp. 305–321 (2024). PMLR
- [45] Zhu, X., Li, J., Liu, Y., Ma, C., Wang, W.: A survey on model compression for large language models. Transactions of the Association for Computational Linguistics **12**, 1556–1577 (2024)

- [46] Shen, A., Lai, Z., Li, D.: Exploring quantization techniques for large-scale language models: Methods, challenges and future directions. In: Proceedings of the 2024 9th International Conference on Cyber Security and Information Engineering, pp. 783–790 (2024)