LoPRo: Enhancing Low-Rank Quantization via Permuted Block-Wise Rotation

Anonymous authors
Paper under double-blind review

Abstract

Post-training quantization (PTQ) enables effective model compression with minimal accuracy loss. While most weight-only PTQ methods target the challenging sub-3-bit regime and often require fine-tuning, we propose LoPRo—a fine-tuning-free algorithm that improves residual matrix quantization via block-wise permutation and Walsh-Hadamard transforms to align columns of similar importance. We also introduce a mixed-precision, fast low-rank decomposition based on rank-1 sketching (R1SVD) to reduce quantization costs. Experiments show LoPRo outperforms existing fine-tuning-free PTQ methods at 2- and 3-bit (with its vector variant LoPRo$_v$ delivering the best 2-bit accuracy on 7B and Mixtral among the compared fine-tuning-free baselines), matching fine-tuned accuracy while achieving up to 5× quantization speedup over GPTVQ. On Mixtral-8x7B, it finishes quantization in under 2.5 hours, reducing perplexity by 0.75 over MoEQuant++ (3.19 vs. 3.00 avg. bits) and boosting BoolQ accuracy by 5.4 points. LoPRo also achieves higher accuracy than other low-rank methods at much lower ranks, with low latency overhead ($\leq$10.3% at batch 1, lower for larger batches). Code: https://anonymous.4open.science/r/LoPRo-8C83

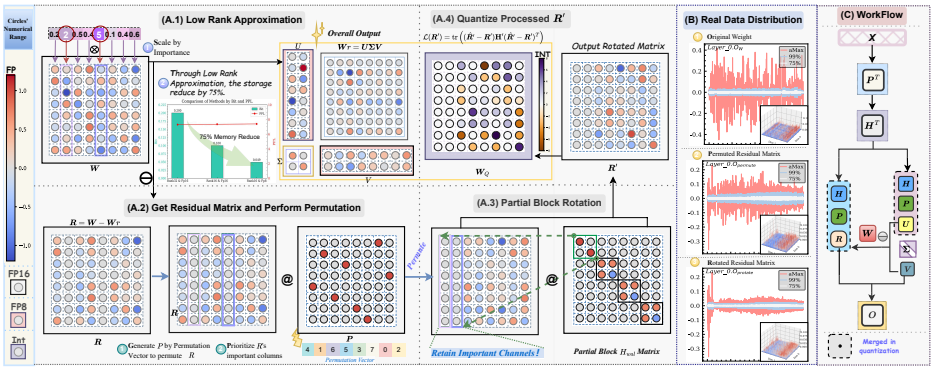

Figure 1: **LoPRo overview.** (A) A visualization of LoPRo transformation; (B) empirical distributions (App. G); (C) the inference workflow. Permutation-partial rotation relieves the residual's quantization burden while R1SVD preserves efficiency.

1 Introduction

Large-scale language models (LLMs) have achieved remarkable success across a wide range of tasks, including text processing and generation. However, as the scope of problems addressed by LLMs expands, these models have grown significantly in size, featuring increasingly complex architectures and parameter counts reaching into the hundreds of millions. To make models suitable for deployment on diverse devices, researchers have explored methods such as pruning, quantization, distillation and their combinations (Kuzmin et al., 2023).

In recent years, Post-Training Quantization (PTQ) has received considerable attention for quantizing models without requiring full model retraining (Ding et al., 2022; Hubara et al., 2021; Frantar et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2023). An important advancement in this area is the emergence of low-rank compensation (LoRC) PTQ methods (Zhang et al., 2024b), which keep the original model weights unchanged during compensation. This design enables lightweight, low-rank modules to be loaded dynamically when needed, enhancing flexibility and efficiency. (Kwon et al., 2023; Zheng et al., 2024). However, such convenience comes at the cost of performance. Existing approaches (Zhang et al., 2024a; Li et al., 2024) often perform poorly under low-bit or require task-specific fine-tuning (Zhang et al., 2024a) to achieve acceptable accuracy, limiting their applicability for rapid task adaptation. This motivates our central question: *Is it possible to design a fine-tuning-free low-rank PTQ method that simultaneously achieves optimal quantization accuracy?*

Challenge 1. An inherent low-rank–rotation conflict. Low-rank PTQ methods (SVD-Quant (Li et al., 2024), LQER (Zhang et al., 2024a)) degrade sharply at low bit, and a natural fix is to port the rotation trick (QuaRot, QuIP#) to the residual. *We show (§3.1) this naive combination is counterproductive*: low-rank decomposition with activation-aware scaling creates a *negative* coupling between Hessian importance $\boldsymbol{H}_{jj}$ and residual step ε_j (important columns have large $\boldsymbol{H}_{jj}$ but small ε_j), so a full Hadamard rotation homogenizes it and, by the negative $\boldsymbol{H}$–ε covariance, *increases* the proxy weighted loss (PPL $8.4 \rightarrow 9.49$ on LLaMA-2 7B at 2-bit; Table 5). Rotation must thus be carefully *tamed*, which is exactly what LoPRo resolves.

Challenges 2. Controlling memory overhead and quantization costs. The memory costs introduced by low-rank quantization are highly sensitive to rank selection, and performing Singular Value Decomposition (SVD) with a large rank introduces substantial inference latency (for example, using ***256*** *ranks reduces throughput by* ***45%!*** *(Saha et al., 2024)*). The challenge is thus to efficiently achieve high quantization accuracy with a minimal rank.

In summary, this work makes the following contributions:

- **A principled LoPRo design.** We formalize (Proposition 1, §3.1) why a full Hadamard rotation *increases* the Hessian-weighted loss on the low-rank residual, resolving a long-standing anomaly (Table 5) inherited by prior combination methods; permutation then partial block rotation emerges as the *minimal* principled repair, not an ad-hoc blend. The resulting fine-tuning-free LoPRo achieves state-of-the-art accuracy with $\sim$3% extra memory and $\leq$10.3% latency at batch size 1 (lower for larger batches).
- We introduce a rank-1 mixed-precision refinement of RSVD that halves the storage overhead while maintaining high computational efficiency. On a single GPU, quantizing a 7B model takes about half an hour, and an 8x7B model can be processed within 3 hours.
- Extensive experiments show that our method delivers state-of-the-art quantization accuracy while also preserving the option for fine-tuning when necessary. Notably, under 3-bit scalar quantization, LoPRo achieves performance that even surpasses the fine-tuned results and maintains good scalability on Mixture-of-Experts (MoE) model.

2 RELATED WORKS

In weight quantization, denote the original weight as $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and compressed into $\hat{\boldsymbol{W}}$; $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ is an input from a calibration set; the primary challenge lies in low-bit quantization at 3-bit and below. For PTQs, we summarize three key points that underlie their effectiveness.To align with the experimental setup, we use distinct colors bar to represent these methods, where the three color segments correspond to the fine-grained strategies (e.g., a, b, c) within ¶O1, ¶F1, ¶F2 and corresponding to the Tags in §4:

Optimization 1 (O1): $||\boldsymbol{W}\boldsymbol{X} - \hat{\boldsymbol{W}}\boldsymbol{X}||$**.** Optimizing the output of linear layers directly—i.e., minimizing the loss between full-precision and quantized outputs is the most straightforward way.

Feature 1 (F1): Weight distribution. Weights are low rank (Hu et al., 2022) and sub-Gaussian (Narkhede et al., 2022), with a few large-magnitude outliers amidst small values.

Feature 2 (F2): Important channel in activation. According to research (Lin et al., 2024b), a small subset of high-magnitude activations plays a crucial role in quantization performance.

Solution to O1: Minimizing loss. Aims to minimize proxy loss by optimizing (Nagel et al., 2020):

$$\mathcal{L}(W) = E_X\left[\|\boldsymbol{W}\boldsymbol{X} - \hat{\boldsymbol{W}}\boldsymbol{X}\|_F^2\right] = \mathrm{tr}\left((\hat{\boldsymbol{W}} - \boldsymbol{W})\boldsymbol{H}(\hat{\boldsymbol{W}} - \boldsymbol{W})^\top\right), \tag{1}$$

where $\boldsymbol{H} = E_{\boldsymbol{X}}[\boldsymbol{X}\boldsymbol{X}^T]$ is a proxy Hessian. The implementation of this principle follows two paths:

- **(a) Optimization in quantization:** GPTQ (Frantar et al., 2023) quantizes weight columns sequentially and compensates for the loss of each column in subsequent columns, GPFQ (Zhang et al., 2023) and GPTAQ (Li et al., 2025) extends this framework to asymmetric calibration. OmniQuant (Shao et al., 2023) applies the loss in weight clipping. MoEQuant (Chen et al., 2025) provides a better calibrate-set selection to balance the loss on experts.
- **(b) Optimization through fine-tuning:** This aspect typically performs fine-tuning by optimizing the loss function $\mathcal{L}$ in Eq. 1. Representative approaches include fine-tuning the codebook, as in QUIP# (Tseng et al., 2024a), and LoRA adaptation such as RILQ (Lee et al., 2025), QERA (Zhang et al., 2024b), and CALDERA (Saha et al., 2024).

Solution to F1: Adapting to the Distribution. Adjust input/output according to the distribution to facilitate quantization:

- **(a) Outlier Truncation:** To mitigate quantization loss by outliers, OmniQuant (Shao et al., 2023) introduces Learnable Weight Clipping (LWC), which optimizes asymmetric clipping thresholds by minimizing Eq. 1.
- **(b) Smoothing by Rotation:** Orthogonal transformations redistribute weight magnitudes more uniformly (Lin et al., 2024a): QuIP (Chee et al., 2023) adds an incoherence step, QuIP# (Tseng et al., 2024a) a randomized Hadamard transform, and SpinQuant (Liu et al., 2025) learnable rotations. QuaRot (Ashkboos et al., 2024) applies the Walsh-Hadamard Transformation (WHT) in Eq. 2 to exploit orthogonal invariance, where Q is the quantization operator and $\boldsymbol{H}_{\text{wal}} \in \mathbb{R}^{m\times m}$ is the normalized Walsh–Hadamard matrix (built recursively as $\boldsymbol{H}_{\text{wal}}^{(2^n)} = \boldsymbol{H}_{\text{wal}}^{(2)} \otimes \boldsymbol{H}_{\text{wal}}^{(2^{n-1})}$) acting on the *output/row* dimension; this row-wise rotation is distinct from the column-wise residual rotation introduced in §3.2.

$$\boldsymbol{W}\boldsymbol{X} = \boldsymbol{H}_{\text{wal}}^T\,\mathsf{Q}(\boldsymbol{H}_{\text{wal}}\boldsymbol{W})\boldsymbol{X}. \tag{2}$$

- **(c) Quantization Format:** Different quantization formats can be used according to the distribution, for instance, LQER adopts the `MaxInt` format (Zhang et al., 2024a). Instead of scalar quantization, vector quantization uses codebooks to better approximate weight patterns. GPTVQ (Van Baalen et al., 2024) improves GPTQ (Frantar et al., 2023) by multidimensional vector quantization. AQLM (Egiazarian et al., 2024) employs a learnable codebook, while QuIP# enhances codebook efficiency using the E_8 lattice structure (Viazovska, 2017).

Solution to F2: Preserving Accuracy for Important Activations. The heavy-tailed nature of activation distributions suggests that precision for significant activations is crucial, leading to two strategies:

- **(a) Activation-Aware Weight Scaling:** AWQ (Lin et al., 2024b) scales weights to preserve the quantization accuracy of weights corresponding to important activations and AffineQuant (Ma et al., 2024) introduces affine transformation quantization for LLMs.
- **(b) Low-Rank in quantization:** While scaling improves robustness for important weight channels, it may amplify quantization errors on less significant ones, especially in ultra-low-bit regimes. To address this, several methods apply low-rank decomposition quantization which can be formulated as:

$$i).\boldsymbol{W}\boldsymbol{X} = (\hat{\boldsymbol{W}} + (\boldsymbol{W} - \hat{\boldsymbol{W}})_r)\boldsymbol{X}; \quad ii).\boldsymbol{W}\boldsymbol{X} = (\boldsymbol{W}_r + \hat{\boldsymbol{R}})\boldsymbol{X}, \tag{3}$$

here $\hat{\boldsymbol{R}} = \mathsf{Q}(\boldsymbol{W} - \boldsymbol{W}_r)$ is the quantized residual matrix and $\boldsymbol{W}_r$ is rank r matrix stored in high precision. There are two forms in the Eq. 3, SVD-Quant (Li et al., 2024) applies form $ii)$ and extends the approach to 4-bits diffusion models and the potential at low bits has not been explored, while fine-tuning PTQ methods apply form $i)$ such as LQER (Zhang et al., 2024a) combines LoRC with MXINT (Darvish Rouhani et al., 2020) quantization; RILQ (Lee et al., 2025) and CALDERA (Saha et al., 2024) further refine it by layer sensitivity analysis and low-bit iteration respectively.

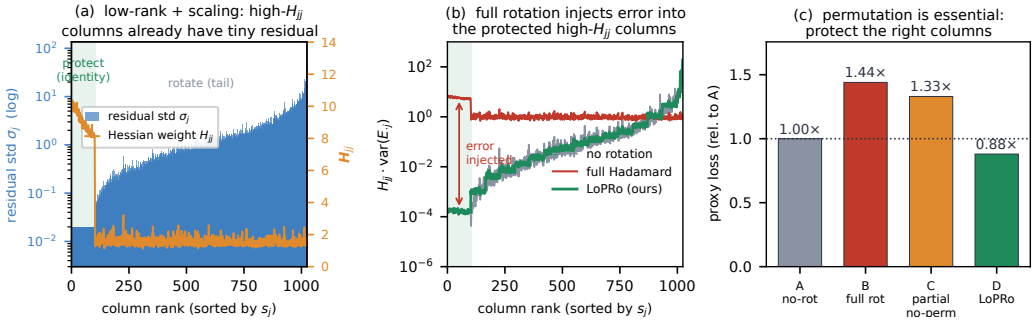

Figure 2: **The low-rank–rotation conflict; permutation is indispensable** (fixed-seed simulation, weighted loss). **(a)** high-$\boldsymbol{H}_{jj}$ columns have tiny residuals (negative coupling); **(b)** full rotation injects $\sim 10^4$–$10^5\times$ error into them; **(c)** loss: full $1.44\times$, no-perm $1.33\times$, LoPRo $0.88\times$.

We evaluate these methods across accuracy, cost, compression ratio, and latency: ¶**O1.a**, ¶**F1.a**, ¶**F2.a** are cheap but degrade below 3-bit; ¶**F1.b,c** and ¶**F2.b** are stronger at low-bit yet add latency (low-rank even hurting compression); and fine-tuning gains accuracy at longer runtime and reduced generalization. Therefore, Our objective is: *Designing a high-accuracy, fine-tuning-free quantization algorithm that simultaneously minimizes inference latency and additional memory overhead.*

3 Method

In this section, we analyze the bottlenecks of low-rank methods in quantization accuracy. Then we propose Low-rank Partial Rotation Quantization – **LoPRo**, a novel fine-tuning-free PTQ method that leverages low-rank quantization and applies partial rotation to efficiently redistribute outliers in the residual matrix. Finally, we enhance quantization performance by incorporating an improved R1SVD algorithm. The workflow of LoPRo is presented in Figure 1.

3.1 Problem Statement and the Low-Rank–Rotation Conflict

¶**O1** and ¶**F1** are not mutually exclusive, but rotation/truncation in ¶**F1** disrupts the important channels preserved in ¶**F2.a**. Low-rank compensation (¶**F2.b**) is thus the promising vehicle: its high-precision $\boldsymbol{W}_r$ fuses with rotation at no extra storage or compute, leaving only the residual $\boldsymbol{R}$ to quantize—our goal is to balance these for the highest fine-tuning-free accuracy.

Of the two forms in Eq. 3, the first suits LoRA-style fine-tuning, while for fine-tuning-free PTQ the second is preferable: quantization disrupts $\boldsymbol{W}$'s inherent low-rank structure, so the clean $\boldsymbol{W}$ is easier to approximate. Here $\boldsymbol{W}_r$ is:

$$\boldsymbol{W}_r = \boldsymbol{W}_L \boldsymbol{W}_R = (\boldsymbol{U}\boldsymbol{\Sigma})(\boldsymbol{V}'), \quad \{\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}\} = \mathrm{SVD}(\boldsymbol{W}\alpha), \quad \boldsymbol{V}' = \boldsymbol{V}^\top \alpha^{-1} = (\alpha^{-1}\boldsymbol{V})^\top, \tag{4}$$

where $\alpha \in \mathbb{R}^{n\times n}$ is a diagonal scaling matrix derived from the layer input (exploiting ¶**F2**); the thin SVD of $\boldsymbol{W}\alpha$ gives $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, hence $\boldsymbol{V}' \in \mathbb{R}^{r\times n}$ and $\boldsymbol{W}_r = (\boldsymbol{U}\boldsymbol{\Sigma})\boldsymbol{V}' \in \mathbb{R}^{m\times n}$; weights act on the right: output $= \boldsymbol{W}\boldsymbol{X}$ with $\boldsymbol{X} \in \mathbb{R}^{n\times b}$. Quantizing the residual $\boldsymbol{R} = \boldsymbol{W} - \boldsymbol{W}_r$ is then the key challenge; since low-rank approximation alone cannot fully smooth the distribution, a natural idea from QuaRot/QuIP# is to further apply a Hadamard rotation.

The anomaly. However, Table 5 shows that a *full* Hadamard rotation on $\boldsymbol{R}$ *degrades* perplexity from 8.4 to 9.49 (LLaMA-2 7B, 2-bit)—a reversal of the classical incoherence effect that prior work has not explained and that any naive low-rank+rotation combination inherits. Below we pin down its cause and derive LoPRo's design, illustrated in Figure 2.

Write the layer-wise proxy quantization loss under a diagonal approximation of the proxy Hessian $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ (Eq. 1):

$$\mathcal{L}(\hat{\boldsymbol{R}}) \;=\; \mathbb{E}_{\boldsymbol{X}}\left[\|(\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{X}\|_F^2\right] \;\approx\; \sum_{j=1}^{n} \boldsymbol{H}_{jj} \left\|\hat{\boldsymbol{R}}_{\cdot j} - \boldsymbol{R}_{\cdot j}\right\|_2^2, \tag{5}$$

where $\boldsymbol{H}_{jj} = \mathbb{E}_{\boldsymbol{X}}[\boldsymbol{X}_{\cdot j}^2]$ is the importance of column j as measured by activations. Under round-to-nearest quantization with group size g and b bits, the per-column squared error equals $m\varepsilon_j^2/12$ with the true symmetric step size $\varepsilon_j = \frac{2}{2^b-1}\max_i |\boldsymbol{R}_{ij}|$ (so that the uniform-quantizer MSE $\varepsilon_j^2/12$ is exact), i.e. $\|\hat{\boldsymbol{R}}_{\cdot j} - \boldsymbol{R}_{\cdot j}\|_2^2 = m\varepsilon_j^2/12$.

Coupling I: low-rank + scaling creates a *negative* $\boldsymbol{H}$–ε coupling. The scaled decomposition (Eq. 4) absorbs important channels (large $\boldsymbol{H}_{jj}$) into $\boldsymbol{W}_r$, so their residuals $\boldsymbol{R}_{\cdot j}$ become very small and their ε_j shrink; we state this as a design premise (empirically verified across layers in Fig. 2a):

$$\mathrm{Cov}_j\big(\boldsymbol{H}_{jj},\, \varepsilon_j^2\big) \;\le\; 0. \tag{6}$$

Coupling II: a full Hadamard rotation destroys this coupling. Applying a full $\boldsymbol{H}_{\mathrm{wal}}^{(n)}$ mixes all columns, homogenizing the effective step sizes to $\bar{\varepsilon} := \sqrt{\frac{1}{n}\sum_j \varepsilon_j^2}$ and each diagonal entry to $\bar{\boldsymbol{H}} := \frac{1}{n}\sum_j \boldsymbol{H}_{jj}$ (as $h_{jk}^2 = 1/n$ for normalized Walsh–Hadamard). Since $\mathcal{L}_{\mathrm{rot}} - \mathcal{L}_{\mathrm{orig}} = -n\,\mathrm{Cov}_j(\boldsymbol{H}_{jj}, \varepsilon_j^2) \ge 0$ by Eq. 6 (equivalently, Chebyshev's sum inequality applied to the oppositely-sorted $\{\boldsymbol{H}_{jj}\}$ and $\{\varepsilon_j^2\}$), the loss reverses:

$$\underbrace{\mathcal{L}_{\mathrm{rot}} \;\approx\; n\,\bar{\boldsymbol{H}}\cdot\bar{\varepsilon}^2}_{\text{homogenized}} \quad\ge\quad \underbrace{\textstyle\sum_j \boldsymbol{H}_{jj}\,\varepsilon_j^2}_{\mathcal{L}_{\mathrm{orig}}}. \tag{7}$$

That is, *full rotation increases the* proxy *weighted loss whenever Eq. (6) holds*, reproducing the anomaly of Table 5. The classical incoherence argument assumes $\boldsymbol{H} \propto \boldsymbol{I}$.

Design principle. *Rotation helps within similar columns, hurts across disparate ones*. LoPRo's two steps (§3.2): (i) **permutation** sorts columns by s_j into homogeneous groups; (ii) **partial rotation** Hadamard-rotates within each group, an identity block shielding the leading b_I. Proposition 1: this gain needs permutation *first*.

3.2 Partial Rotation Quantization under Permutation

Building on §3.1, we instantiate the two-step design for quantizing the residual $\boldsymbol{R}$: after token-wise scaling, important columns have small values whose stability is crucial (i), yet standard quantization cannot single out the columns flagged by the proxy-Hessian diagonal (ii). Our block-wise partial rotation quantization has three components:

1. Column Permutation: Considering quantization difficulty and column importance, we introduce a reordering strategy formulated as Eq. 8, and leave a detailed discussion in Appendix A.4.

$$s_j = \frac{\boldsymbol{H}_{jj}}{\frac{1}{m}\sum_{i=1}^{m}|\boldsymbol{R}_{ij}|} \quad \text{and} \quad p = \mathrm{argsort}_j\big(-s_j\big) \quad , \quad j = 1, 2, \ldots, n, \tag{8}$$

where $\boldsymbol{H}$ is the proxy Hessian computed in Eq. 1, $\boldsymbol{R}$ is the residual matrix after low-rank approximation, and the denominator is the mean absolute value of column j. The negative sign in $\mathrm{argsort}(-s_j)$ sorts in *descending* order of s_j, placing the most important columns (high $\boldsymbol{H}_{jj}$, low $\overline{|R_{\cdot j}|}$) at the leading positions. This reordering can be expressed by applying permutation matrix $\boldsymbol{P}$, where $\boldsymbol{P}_{i,p[i]} = 1$ and other elements are 0. The matrix $\boldsymbol{P}$ is orthogonal and satisfies $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$. After permutation, important columns are moved to the leading positions of the residual matrix.

2. Partial block Rotation: Rotation enhances the incoherence between $\boldsymbol{R}$ and $\mathbf{H}$, thereby improving quantization accuracy. However, applying a full rotation would disrupt the column importance ordering and degrade accuracy. To address this, we propose a partial rotation in a block-wise manner:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{I} & 0 & \cdots & \cdots & 0 \\ 0 & \boldsymbol{H}_{\mathrm{wal}}^{(b_H)} & 0 & \cdots & \vdots \\ \vdots & 0 & \ddots & 0 & \vdots \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \boldsymbol{H}_{\mathrm{wal}}^{(b_H)} \end{bmatrix}, \boldsymbol{R} = \boldsymbol{RPQ} \tag{9}$$

where $\boldsymbol{I}$ is an identity matrix of size b_I and $\boldsymbol{H}_{\text{wal}}^{(b_H)} \in \mathbb{R}^{b_H \times b_H}$ the normalized Walsh–Hadamard matrix (b_H a power of 2). The leading (most important) columns stay unrotated to avoid degradation, while block-wise rotation *within* similarly-difficult groups recovers the incoherence gain without destroying the $\boldsymbol{H}$–ε coupling that low-rank+scaling established across groups. This is formalized as:

Proposition 1 (informal; full statement and proof in App. B.2). *Within any permuted block $\mathcal{B}$ of size b_H whose step-size and Hessian ratios are bounded by ρ, η, the intra-block Hadamard yields a net decrease in the block's contribution to the loss equation 5 whenever the classical incoherence gain $\Delta_{\text{inc}}(b_H) \geq 0$ exceeds the slack $\frac{m}{12}(\eta-1)(\rho^2-1)\,\bar{\boldsymbol{H}}_{\mathcal{B}}\, b_H\, \overline{\varepsilon^2}_{\mathcal{B}}$, which vanishes as $\rho, \eta \to 1$ (Eq. 22).*

Remark (why partial beats full). For the identity block (b_I leading columns), $\mathcal{L}_{\text{rot}} = \mathcal{L}_{\text{orig}}$ trivially, so those high-$\boldsymbol{H}$ / low-ε columns are not contaminated. For a *full* rotation, (A1)–(A2) fail with $\rho \gg 1$, $\eta \gg 1$, and the covariance view equation 7 yields the loss increase observed in Table 5. Permutation is thus not an add-on but the *enabling condition* for rotation to be beneficial under (A1)–(A3).

3. Quantization with Rotation Matrix

After the low-rank approximation and partial Walsh-Hadamard rotation described in the previous section, we effectively leverage the feature ¶**F1** and ¶**F2** to obtain $\boldsymbol{R}'$ that is more amenable to quantization. Recall the low-rank quantization form in Eq. 3 under transformation in Eq. 9.

$$\boldsymbol{W}\boldsymbol{X} = \boldsymbol{W}_r\boldsymbol{X} + \mathsf{Q}(\boldsymbol{R}')\boldsymbol{Q}^T\boldsymbol{P}^T\boldsymbol{X}, \tag{10}$$

This decouples the quantized and low-rank components, so methods from ¶**O1** apply directly to $\boldsymbol{R}'$. Under the substitution $\tilde{\boldsymbol{X}} := \boldsymbol{Q}^\top\boldsymbol{P}^\top\boldsymbol{X}$ and the rotated-frame error $\hat{\boldsymbol{E}} := \hat{\boldsymbol{R}}' - \boldsymbol{R}'$ (with $\hat{\boldsymbol{R}}' = \mathsf{Q}(\boldsymbol{R}')$ the quantized residual in the rotated frame), the loss equation 10 becomes $\mathcal{L} = \text{tr}(\hat{\boldsymbol{E}}\boldsymbol{H}'\hat{\boldsymbol{E}}^\top)$ with rotated Hessian $\boldsymbol{H}' = \boldsymbol{Q}^\top\boldsymbol{P}^\top\boldsymbol{H}\boldsymbol{P}\boldsymbol{Q}$ (Lemma 1, App. B.3); GPTQ-style column-wise optimization thus applies to $\boldsymbol{R}'$ at no extra cost.

3.3 A Light Low-Rank Algorithm by Randomized Sketching

Low-rank methods typically keep $\boldsymbol{W}_L, \boldsymbol{W}_R$ in full precision (Lee et al., 2025; Zhang et al., 2024a); CALDERA (Saha et al., 2024) instead quantizes them, but needs a large rank ($\sim$256) that incurs heavy compute. To reduce this overhead while preserving accuracy, we propose **R1SVD**, a rank-1 simplification of randomized SVD (Frieze et al., 2004; Musco & Musco, 2015). Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, a standard RSVD prototype consists of the following two steps:
Stage A: Generate an $\mathbb{R}^{n\times r}$ Gaussian test matrix $\boldsymbol{S}$ and form $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\boldsymbol{S}$, where it is the iteration times. Construct a matrix $\boldsymbol{Q} = QR(\boldsymbol{Y})$ by QR decomposition whose columns form an orthonormal basis of $\boldsymbol{Y}$.

Stage B: Form $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$. Compute an SVD of the small matrix: $\boldsymbol{B} = \boldsymbol{U}'\boldsymbol{\Sigma}\boldsymbol{V}^*$. Set $\boldsymbol{U} = \boldsymbol{Q}\boldsymbol{U}'$.

If a rank-1 matrix $\boldsymbol{S} \in \mathbb{R}^{n\times 1}$ is utilized for low-rank approximation of a matrix and substituted into the two stages, we also have $Y = (\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}S$, then for the matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times 1}$, the QR decomposition can be directly represented as follows.

$$\boldsymbol{Q} = \frac{\boldsymbol{Y}}{\|\boldsymbol{Y}\|} \in \mathbb{R}^{m\times 1}, \boldsymbol{R} = \|\boldsymbol{Y}\| \in \mathbb{R}^{1\times 1}. \tag{11}$$

Similarly, the SVD decomposition for rank-1 matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$ can be represented as follows:

$$\boldsymbol{U}' = \{1\}, \boldsymbol{\Sigma} = \|\boldsymbol{B}\|, \boldsymbol{V} = \frac{\boldsymbol{B}}{\|\boldsymbol{B}\|}. \tag{12}$$

Applying **Stage B** and Eq. 11, we have the rank-1 matrix $\boldsymbol{A}_1 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$:

$$\boldsymbol{U} = \frac{(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\boldsymbol{S}}{\|(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\boldsymbol{S}\|}, \quad \boldsymbol{\Sigma} = \frac{\|\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\|}{\|(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\boldsymbol{S}\|}, \quad \boldsymbol{V} = \frac{\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}}{\|\boldsymbol{S}^*\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{it}\boldsymbol{A}\|}. \tag{13}$$

Then, we set $\boldsymbol{A} = \boldsymbol{A} - \boldsymbol{A}_1$ and apply iteration to this process, which enables decomposition at any rank. Since the singular value matrix $\boldsymbol{\Sigma}$ is diagonal with low storage cost, by storing $\boldsymbol{U}$ and $\boldsymbol{V}$ in *fp8* while retaining $\boldsymbol{\Sigma}$ in *fp16*, the storage overhead can be reduced by half. Furthermore, the loss in precision is compensated in subsequent iterations, thereby preserving the overall accuracy.

Table 1: The Quantization Results of LoPRo and baselines. We report perplexity of WikiText2 and four zero-shot accuracy. MoEQ is short for MoEQuant++, see 1. The meaning of 'Tag' and abbreviations for zero-shot tasks follow §4.1. More detailed settings are given in Appendix C.

Model	Method	Tag	Bit	PPL↓	ZeroShot Acc↑				Bit	PPL↓	ZeroShot Acc↑			
					AC	AE	WI	QA			AC	AE	WI	QA
LLaMA2-7B	FP16		16	5.12	43.4	76.3	69.1	78.4	16	5.12	43.4	76.3	69.1	78.4
	GPTQ		2.13	50.8	20.9	34.9	52.3	57.2	3.13	8.06	31.1	58.5	59.2	71.5
	GPTVQ		2.13	6.89	30.2	64.3	64.1	72.1	3.13	5.61	39.9	74.1	**69.1**	76.2
	LQER		2.80	10.3	33.1	60.4	61.2	70.7	3.28	5.72	40.4	74.2	68.4	74.9
	OmniQ		2.13	15.0	28.8	58.1	59.1	70.2	3.13	5.81	40.8	74.5	67.5	77.7
	Quip#		2.00	8.22	29.9	61.3	61.7	69.6	3.00	5.79	40.2	**75.1**	67.0	76.2
	LoPRo		2.19	7.39	31.2	62.8	63.8	71.1	3.19	**5.43**	**41.0**	74.9	68.9	76.3
	LoPRo$_v$		2.19	**6.53**	**34.6**	**69.0**	**66.5**	**72.7**	3.19	5.45	**41.0**	74.8	**69.1**	**76.7**
LLaMA2-13B	FP16		16	4.57	49.1	77.4	73.9	81.4	16	4.57	49.1	77.4	73.9	81.4
	GPTQ		2.13	43.8	23.3	43.3	54.7	61.3	3.13	5.85	38.5	65.7	63.9	76.5
	GPTVQ		2.13	5.78	38.7	73.6	**68.5**	**75.4**	3.13	4.92	44.5	75.2	**72.0**	77.8
	LQER		2.64	8.42	33.2	65.8	66.4	73.1	3.24	5.12	42.3	76.7	71.2	77.4
	OmniQ		2.13	11.1	31.3	62.3	65.6	72.3	3.13	5.11	42.0	77.9	71.3	78.0
	Quip#		2.00	6.06	36.2	68.6	63.6	74.2	3.00	4.90	42.2	76.6	71.5	77.6
	LoPRo		2.19	6.48	33.6	69.0	66.3	72.4	3.19	**4.84**	**44.9**	**78.7**	71.1	**78.5**
	LoPRo$_v$		2.19	**5.79**	**38.8**	**74.2**	68.0	**75.4**	3.19	4.87	44.5	77.4	71.0	78.3
LLaMA3-8B	FP16		16	5.54	50.2	80.1	73.5	79.7	16	5.54	50.2	80.1	73.5	79.7
	GPTQ		2.13	2e2	21.1	29.3	52.1	54.4	3.13	7.81	37.7	70.5	71.1	74.9
	GPTVQ		2.13	9.32	31.3	57.3	**68.3**	67.8	3.13	6.78	45.1	76.7	72.5	77.8
	OmniQ		2.13	54.1	19.3	36.1	51.9	59.0	3.13	7.01	42.2	72.4	71.3	75.5
	Quip#		2.00	10.9	30.8	57.1	67.0	67.5	3.00	6.75	45.2	75.2	72.3	78.0
	LoPRo		2.19	10.8	30.5	58.9	62.2	68.2	3.19	**6.31**	44.1	75.9	**73.0**	77.5
	LoPRo$_v$		2.19	**8.95**	**37.0**	**69.2**	68.0	**71.4**	3.19	6.41	**45.3**	**76.8**	71.9	**78.4**
Mixtral-8x7B	FP16		16	3.84	62.0	87.3	75.3	83.5	16	3.84	62.0	87.3	75.3	83.5
	GPTQ		2.13	14.1	26.6	35.7	49.5	57.7	3.13	4.71	52.1	69.3	74.4	80.9
	GPTVQ		2.13	5.28	42.0	71.6	66.5	75.9	3.13	4.27	54.9	72.9	74.8	82.6
	MoEQ		3.00	4.90	-	-	-	-	3.00	4.90	-	-	-	-
	LoPRo		2.19	5.25	53.2	81.8	72.0	78.1	3.19	**4.15**	**61.0**	85.3	**77.0**	**83.3**
	LoPRo$_v$		2.19	**4.80**	**55.8**	**82.9**	**74.0**	**80.5**	3.19	4.16	60.6	**86.5**	76.7	82.9

4 Evaluation

4.1 Experiment Setup

Models: We evaluate the LLaMA-2 and LLaMA-3 dense families (Touvron et al., 2023), the Mixture-of-Experts (MoE) Mixtral-8x7B (Jiang et al., 2024), and the Qwen2,3 families (Team, 2024; 2025).

Baseline: Following the classification in §2, we compare with SOTA PTQs within each class. Fine-tuning-free methods include GPTQ (Frantar et al., 2023), GPTVQ (Van Baalen et al., 2024), OmniQuant (Shao et al., 2023), QuIP# (Tseng et al., 2024a), LQER (Zhang et al., 2024a), and MoEQuant (Chen et al., 2025) [1]. We also compare fine-tuning-based methods (QuIP#, CALDERA

[1]The implementation of MoEQuant is now unavailable, it does not report results on several datasets used in our main experiments. To ensure a comprehensive comparison, we provide additional evaluation in Appendix E.

Table 2: Fine-tuning results for the LLaMA-2 family. The notation "rank(bit)" denotes the rank and bitwidth used for the low-rank components. LoPRo employs RILQ as the fine-tuning backend; further implementation details are provided in Appendix C.

	Method	Tag	Bit	rank(bit)	PPL	ZeroShot Acc			
						AC	AE	WI	QA
LLaMA2-7B	CALDERA		2.2	128(4)	6.79	34.6	65.1	63.8	75.1
	RILQ		2.2	32(16)	6.28	37.4	70.1	**66.5**	75.0
	LoPRo$_v$-FT		2.2	32(8)	**6.06**	**38.1**	**70.9**	64.8	**76.1**
	RILQ		3.2	32(16)	5.47	40.6	75.8	67.9	76.9
	LoPRo$_v$-FT		3.2	32(8)	**5.37**	**42.8**	**76.0**	**68.9**	**77.2**
	Method	Tag	Bit	rank(bit)	PPL	ZeroShot Acc			
						AC	AE	WI	QA
LLaMA2-13B	CALDERA		2.2	128(4)	5.72	38.7	68.5	67.9	76.0
	RILQ		2.2	32(16)	5.42	40.1	71.2	**68.8**	78.1
	LoPRo$_v$-FT		2.2	32(8)	**5.34**	**42.8**	**75.6**	68.5	**78.9**
	RILQ		3.2	32(16)	4.88	45.4	76.4	71.4	79.2
	LoPRo$_v$-FT		3.2	32(8)	**4.80**	**46.2**	**77.9**	**71.7**	**80.1**

(Saha et al., 2024)). Methods are tagged per §2: for ¶**O1.a,b**, for ¶**F1.a,b,c**, for ¶**F2.a,b**, and for an unused strategy [2].

Metrics: We evaluate perplexity on WikiText2 (Merity et al., 2016) and zero-shot accuracy on ARC-challenge (AC), ARC-easy (AE)(Boratko et al., 2018), PIQA (QA)(Bisk et al., 2020), and Winogrande (WI)(Sakaguchi et al., 2021), using the lm-evaluation-harness (Gao et al., 2024). Other settings are in Appendix C.

4.2 Main Results

LoPRo consistently outperforms all SOTA baselines across settings on the LLaMA family; we analyze these results by category.

LLaMA-2 results: (1) vs. scalar: LoPRo beats GPTQ and OmniQuant, and unlike LQER (which needs r=256 at 2-bit) it matches them with only r=16 via importance-aware rotation. **(2) vs. vector:** at 2-bit, LoPRo$_v$ cuts GPTVQ's 7B PPL by **0.36**↓ (6.53 vs. 6.89) and gains **+3.0 points** average zero-shot (60.7 vs. 57.7) at only $\sim$3% extra memory, while scalar LoPRo remains competitive on zero-shot accuracy at the same 2-bit budget; at 3-bit LoPRo beats all fine-tuning-free baselines, where scalar-LoPRo already suffices. On 13B at 2-bit, LoPRo$_v$ is essentially tied with GPTVQ (5.79 vs. 5.78), and QTIP's iterative-codebook vector quantizer is stronger (App. E.1); our 2-bit edge thus concentrates on 7B and Mixtral.

LLaMA-3 & MoE Results: LLaMA-3 is less redundant and more sensitive to low-bit quantization (Huang et al., 2024), where clipping perturbs weights and worsens accuracy loss. As Table 1 shows, LoPRo's relative gain is even larger on LLaMA-3 than LLaMA-2. On Mixtral-8x7B, LoPRo beats GPTQ on zero-shot accuracy across all tasks, and LoPRo$_v$ at 2.19 average bits approaches MoEQuant++ at 3-bit (PPL 4.80 vs. 4.90; App. E.2), showing strong generalization.

Fine-tuning Results: LoPRo is a general PTQ method that extends easily to fine-tuning. Against RILQ/CALDERA+QuIP# (Table 2), RILQ fine-tuning on LoPRo$_v$ (whose pre-quantizer is LoPRo$_v$ itself rather than QuIP#) outperforms fine-tuned QuIP# across all bitwidths. Its 3-bit fine-tuning results are very close to the **3-bit fine-tuning-free** ones in Table 1 (**0.06** PPL on LLaMA-2 7B, **0.04** on 13B), indicating partial-block-wise rotation already removes most 3-bit quantization error.

4.3 Quantization Cost

LoPRo's quantization cost is dominated by ❶ low-rank sketching with the block-wise $\boldsymbol{H}_{wal}$ transform on $\boldsymbol{R}$ and ❷ quantizing $\mathbf{R}'$. It is far faster than OmniQuant and QuIP#, taking under **2.5** hours on Mixtral-8x7B, since R1SVD runs in $\mathcal{O}(N^2)$ time—linear in model size—versus $\mathcal{O}(N^3)$ for full

[2] For example, GPTVQ with means it employ loss method in ¶**O1.a** and vector quantization in ¶**F1.c**.

Table 3: Quantization time for methods, where 'h' denotes hours and 'm' denotes minutes. *NA* indicates that the method was not implemented or encountered OOM errors during execution.

Model	GPTQ	GPTVQ	LQER	AQLM	OmniQ	LoPRo	LoPRo$_v$
LLaMA2-7B	25.2m	1.5h	45.2m	11.1h	3.1h	**26.4m**↓↓	**32.2m**↓
LLaMA2-13B	40.5m	3.7h	1.2h	22.7h	5.3h	**44.5m**↓↓	**56m**↓
Mixtral-8x7B	2.6h	*NA*	*NA*	*NA*	*NA*	**2.0h**↓↓	**2.4h**↓

SVD. Even under vector quantization, LoPRo stays faster than GPTVQ using only the simplest codebook (no LoRA or fine-tuning), yet reaches higher accuracy.

4.4 Inference Efficiency

We take the W4A16 kernel in GPTQ-Model (qubitium, 2024) as baseline (Table 4). Only linear layers are affected; rotation is applied to $\mathbf{X}$ via a Fast Walsh-Hadamard Transform in $\mathcal{O}(n \log b_H)$ per token. With a small rank ($r = 16$), the low-rank components add up to 6.0% latency at batch size 1 (Table 4), and the *relative* overhead decreases with model scale (since $2r \ll n$), consistent with Appendix A.3 and the batch-size trend.

Table 4: Inference throughput and latency of the LoPRo, where 'Rot' is in short of Rotation.

Model	Batch	Decode (token/s)		Latency		
		Baseline	LoPRo	Rot	LoRA	Total
7b	1	93.3	83.7	4.3%	6.0%	10.3%
	16	368.6	334.5	3.7%	5.6%	9.3%
	64	450.0	412.8	3.2%	5.1%	8.3%
13b	1	62.4	56.1	3.9%	6.2%	10.1%
	16	274.4	252.4	3.3%	4.7%	8.0%
	64	388.4	358.2	2.9%	4.9%	7.8%

4.5 Ablation Studies

Table 5 ablates components at 2-bit: replacing full $\boldsymbol{H}_{wal}$ with permuted partial-block $\boldsymbol{H}_{wal}$ cuts perplexity by 22% (9.49 → 7.39), and vector quantization lowers it further to 6.53—the best 2-bit result on 7B among compared methods—validating structured rotation as analyzed in § 3.

Table 5: Ablation on different components in LoPRo under 2-bit LLaMA2-7b. 'NA' means non-use of such strategy. 'OQ' and 'VQ' denote use GPTQ and GPTVQ as the quantizer respectively. The last two bolded lines represent LoPRo and LoPRo$_v$

Bits	Rotation	Quant	Tag	PPL	Acc
16	NA	NA		5.12	66.8
2.2	NA	RTN		4.0e2	44.1
2.2	NA	OQ		8.4	53.0
2.2	Full $\boldsymbol{H}_{wal}$	OQ		9.49	51.5
2.2	Partial $\boldsymbol{H}_{wal}$	OQ		7.39	57.2
2.2	Partial $\boldsymbol{H}_{wal}$	VQ		6.53	60.7

Additional ablations on 'rank', 'iteration', SVD method, 'rotation block', and 'calibration set' are in Appendix D, together with further experiments: (1) baselines—QTIP (Tseng et al., 2024b) (App. E.1) and MoEQuant (App. E.2); (2) models—Qwen2.5/Qwen3 (App. E.3); and (3) benchmarks—Open LLM Leaderboard V1 (App. E.4). All corroborate our finding: **LoPRo maintains strong performance under high compression across diverse architectures and scales.**

5 Conclusion

In this work, we propose LoPRo, an efficient PTQ method that utilizes the characteristics of the residual matrix after low-rank decomposition. LoPRo introduces a block-wise partial Hadamard rotation under permutation, which effectively reduces the quantization difficulty of the residual matrix while preserving its importance structure. Furthermore, we employ a mixed-precision R1SVD approximation to replace SVD, significantly reducing computational error and accelerating decomposition. Through comprehensive evaluations, LoPRo achieves state-of-the-art performance across various tasks including MoE LLMs, achieving high accuracy, compression ratio and efficiency in fine-tuning-free quantization while preserving scalability for fine-tuning.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance during implementation and for polishing text. All AI-assisted work has been reviewed. We take responsibility for the final content of this work.

REFERENCES

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432–7439, Jun 2020. doi: 10.1609/aaai.v34i05.6239. URL http://dx.doi.org/10.1609/aaai.v34i05.6239.

Michael Boratko, Harshit Padigela, Divyendra Mikkilineni, Pritish Yuvraj, Rajarshi Das, Andrew McCallum, Maria Chang, Achille Fokoue-Nkoutche, Pavan Kapanipathi, Nicholas Mattei, et al. A systematic classification of knowledge, reasoning, and context within the arc dataset. *arXiv preprint arXiv:1806.00358*, 2018.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36: 4396–4429, 2023.

Zhixuan Chen, Xing Hu, Dawei Yang, Zukang Xu, XUCHEN, Zhihang Yuan, Sifan Zhou, and JiangyongYu. MoEQuant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0epuNvt5Dj.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

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

Bita Darvish Rouhani, Daniel Lo, Ritchie Zhao, Ming Liu, Jeremy Fowers, Kalin Ovtcharov, Anna Vinogradsky, Sarah Massengill, Lita Yang, Ray Bittner, et al. Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. *Advances in neural information processing systems*, 33:10271–10281, 2020.

Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. Towards accurate post-training quantization for vision transformer. In *Proceedings of the 30th ACM international conference on multimedia*, pp. 5380–5388, 2022.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 12284–12303, 2024.

Fino and Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, C-25(11):1142–1146, 1976. doi: 10.1109/TC.1976.1674569.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

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

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

Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. How good are low-bit quantized llama3 models? an empirical study. *CoRR*, abs/2404.14047, 2024. URL https://doi.org/10.48550/arXiv.2404.14047.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pp. 4466–4475. PMLR, 2021.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Andrey Kuzmin, Markus Nagel, Mart Van Baalen, Arash Behboodi, and Tijmen Blankevoort. Pruning vs quantization: Which is better? *Advances in neural information processing systems*, 36:62414–62427, 2023.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Geonho Lee, Janghwan Lee, Sukjin Hong, Minsoo Kim, Euijai Ahn, Du-Seong Chang, and Jungwook Choi. Rilq: Rank-insensitive lora-based quantization error compensation for boosting 2-bit large language model accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18091–18100, 2025.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. *arXiv preprint arXiv:2411.05007*, 2024.

Yuhang Li, Ruokai Yin, Donghyun Lee, Shiting Xiao, and Priyadarshini Panda. Gptaq: Efficient finetuning-free quantization for asymmetric calibration. *arXiv preprint arXiv:2504.02692*, 2025.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100, 2024b.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Representation Learning*, volume 2025, pp. 92009–92032, 2025.

Yuexiao Ma, Huixia Li, Xiawu Zheng, Feng Ling, Xuefeng Xiao, Rui Wang, Shilei Wen, Fei Chao, and Rongrong Ji. Affinequant: Affine transformation quantization for large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Advances in neural information processing systems*, 28, 2015.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Meenal V Narkhede, Prashant P Bartakke, and Mukul S Sutaone. A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322, 2022.

qubitium. Gpt-qmodel. https://github.com/modelcloud/gptqmodel, 2024. Contact: qubitium@modelcloud.ai.

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

Rajarshi Saha, Naomi Sagan, Varun Srivastava, Andrea Goldsmith, and Mert Pilanci. Compressing large language models using low rank and low precision decomposition. *Advances in Neural Information Processing Systems*, 37:88981–89018, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*, 2023.

Qwen Team. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10112.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip# : Even better llm quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning*, 2024a.

Albert Tseng, Qingyao Sun, David Hou, and Christopher M De Sa. Qtip: Quantization with trellises and incoherence processing. *Advances in Neural Information Processing Systems*, 37:59597–59620, 2024b.

Mart Van Baalen, Andrey Kuzmin, Ivan Koryakovskiy, Markus Nagel, Peter Couperus, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. Gptvq: The blessing of dimensionality for llm quantization. *arXiv preprint arXiv:2402.15319*, 2024.

Maryna S Viazovska. The sphere packing problem in dimension 8. *Annals of mathematics*, pp. 991–1015, 2017.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Cheng Zhang, Jianyi Cheng, George A Constantinides, and Yiren Zhao. Lqer: Low-rank quantization error reconstruction for llms. *arXiv preprint arXiv:2402.02446*, 2024a.

Cheng Zhang, Jeffrey TH Wong, Can Xiao, George A Constantinides, and Yiren Zhao. Qera: an analytical framework for quantization error reconstruction. *arXiv preprint arXiv:2410.06040*, 2024b.

Jinjie Zhang, Yixuan Zhou, and Rayan Saab. Post-training quantization for neural networks with provable guarantees. *SIAM Journal on Mathematics of Data Science*, 5(2):373–399, 2023.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37: 62557–62583, 2024.

Appendix Contents

We provide extended technical and experimental details in this appendix, outlined below:

- **LoPRo Details** (Sec. A)
  - Algorithm pseudo-code (Sec. A.1)
  - Average Bit-width (Sec. A.2)
  - Time Complexity (Sec. A.3)
  - Permutation (Sec. A.4)
- **Proofs** (Sec. B)
  - Notations and Setup (Sec. B.1)
  - Proof of Proposition 1: Intra-block rotation reduces weighted loss (Sec. B.2)
  - Proof of Lemma 1: Rotated-Hessian loss (Sec. B.3)
- **Additional Implementation Details** (Sec. C)
- **Extended Ablation Studies** (Sec. D)
  - Ablation on rank (Sec. D.1)
  - Ablation on Low-rank Decomposition (Sec. D.2)
  - Ablation on iteration (Sec. D.3)
  - Ablation on block size (Sec. D.4)
  - Ablation on Calibration dataset (Sec. D.5)
- **Additional Evaluations** (Sec. E)
  - Comparison with QTIP (Sec. E.1)
  - Comparison with MoEQuant (Sec. E.2)
  - Qwen Results (Sec. E.3)
  - Open LLM Leaderboard V1 (Sec. E.4)
- **Limitations** (Sec. F)
- **More Visualization** (Sec. G)

Table 6: Symbols and Description in this paper.

Symbols	Description
$\boldsymbol{W}$	The weight matrix $\boldsymbol{W} \in \mathbb{R}^{m\times n}$.
$\boldsymbol{X}$	The input of each linear layer
Q	Quantization operator (sans-serif; distinct from the rotation matrix $\boldsymbol{Q}$ below)
$\hat{\boldsymbol{W}}$	A pseudo quantized matrix
$\boldsymbol{H}$	A proxy hessian equal to $E_{\boldsymbol{X}}[\boldsymbol{X}\boldsymbol{X}^T]$
$\boldsymbol{H}_{\text{wal}}$	Normalized Walsh–Hadamard matrix ($\boldsymbol{H}_{\text{wal}} \in \mathbb{R}^{m\times m}$ in Eq. 2; $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ blocks in Eq. 9)
$\boldsymbol{W}_r = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}$	The rank-r approximate matrix of $\boldsymbol{W}$
$\mathcal{L}(\cdot)$	The loss function, we use the proxy Hessian in this paper.
$\boldsymbol{R}$	The residual matrix of low-rank approximation $\boldsymbol{R} = \boldsymbol{W} - \boldsymbol{W}_{\boldsymbol{r}}$
$\boldsymbol{Q}$	The partial block Walsh–Hadamard rotation matrix (bold; distinct from the quantization operator Q above); its $b_H \times b_H$ blocks are $\boldsymbol{H}_{\text{wal}}^{(b_H)}$
$\boldsymbol{P}$	The permutation matrix from index vector $perm$.
$\boldsymbol{I}$	An identity matrix
$\boldsymbol{E}$	The residual quantization error $\boldsymbol{E} = \hat{\boldsymbol{R}} - \boldsymbol{R}$ (App. B.1).
$\mathcal{O}(\cdot)$	The upper bound of time complexity
$\text{argsort}(-\cdot)$	Return indices sorting the array in descending order.
$amean(\cdot, axis = 0)$	Take the average of the absolute values along the dimension 0.
$\boldsymbol{E}_j$	j-th column of the error matrix $\boldsymbol{E}$ (Eq. 16).
$\hat{\boldsymbol{E}}$	Rotated-frame error $\hat{\boldsymbol{E}} = \hat{\boldsymbol{R}}' - \boldsymbol{R}' = (\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{P}\boldsymbol{Q}$ (Lemma 1).
$\text{Cov}_{\mathcal{B}}(\boldsymbol{H}, \varepsilon^2)$	Block covariance $\sum_{j\in\mathcal{B}}(\boldsymbol{H}_{jj} - \bar{\boldsymbol{H}}_{\mathcal{B}})(\varepsilon_j^2 - \bar{\varepsilon}_{\mathcal{B}}^2)$ (Eq. 19).
$\Delta_{\text{inc}}(b_H)$	Incoherence gain $\mathcal{L}_{\text{rot}}^{(\mathcal{B}),\text{var}} - \mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max} \geq 0$ (Eq. 20).

A LoPRo Details:

Algorithm 1 Flexible Low-Rank Matrix Sketching Quantization. The residual rotation $\boldsymbol{R}' = \boldsymbol{RPQ}$ uses the *deterministic* normalized Walsh–Hadamard block $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ of Eq. 9 (no random-sign diagonal); the only randomness is the R1SVD sketch. Proposition 1's guarantee holds under the incoherence Assumption (A3) of App. B.2.

Require: $Module$(module weight), $Sample$(calibration data), d(quantization bit),$rank$(the rank)
Ensure: $QModule$(quantized module weight), $LoraModule$(low-rank component)

1: **for** $layers$ in $Module$ **do**
2: Obtain the activation for each layer during model inference: $Acts \leftarrow layers.forward()$.
3: **for** l in $layers$ **do**
4: Obtain the weights and Acts: $\{\boldsymbol{W}, \boldsymbol{X}\} \leftarrow \{layers[l], Acts[l]\}$
5: Calculate the mean of the activation values: $\overline{X} = mean(|\boldsymbol{X}|, axis = 0)$
6: Calculate the scale from $\boldsymbol{X}$: $s_j \leftarrow \overline{X}_j^{\gamma}$, where $\gamma = \arg\min_{\gamma' \in [0,1]} \|(\boldsymbol{W} - \mathrm{Q}(\boldsymbol{W}\,\mathrm{diag}(\overline{X}^{\gamma'}))\,\mathrm{diag}(\overline{X}^{-\gamma'})\boldsymbol{X}\|_F^2$ (grid-searched over $\{0, 0.25, 0.5, 0.75, 1\}$ as in AWQ)
7: Get $\boldsymbol{W}_s$: $\boldsymbol{W}_s \leftarrow \boldsymbol{W} \cdot diag(s)$
8: **for** r in $rank$ **do**
9: draw a random sketch vector: $v = random(\boldsymbol{W}.shape[1])$
10: calculate $2 + 2 \cdot iter$ times $GEMV$: $y = (\boldsymbol{W}_s \boldsymbol{W}_s^T)^{iter} \boldsymbol{W}_s v,\ p = \boldsymbol{W}_s^T y$
11: Get rank-1 component: $U_1 = (y/\|y\|).to(fp8)\ ,\ V_1 = (p/\|p\|).to(fp8)$
12: Add to the low-rank component: $\boldsymbol{U}.add(U_1), \boldsymbol{V}.add(V_1), \boldsymbol{\Sigma}.add(\sigma)$
13: Update the matrix: $\boldsymbol{W}_s = \boldsymbol{W}_s - \sigma U_1 V_1$
14: **end for**
15: Calculate Residual matrix and proxy Hessian: $\boldsymbol{R} = \boldsymbol{W}_s \cdot diag(s)^{-1}\ , H = E_X[XX^T]$
16: Get permutation index p: $p = \text{argsort}(-\text{diag}(\boldsymbol{H})/\text{amean}(|\boldsymbol{R}|, \text{dim} = 0))$ *(descending, Eq.8)*
17: Build permutation matrix: $\boldsymbol{P} = 0, \boldsymbol{P}_{k,p[k]} = 1$
18: Apply partial block hadamard transformation in Eq. 9: $\boldsymbol{R}' = \boldsymbol{RPQ}$
19: Quantize the residual matrix by methods like GPTQ/GPTVQ... by new loss: $\boldsymbol{W}_q = \mathrm{Q}(\boldsymbol{R}')\ , \mathcal{L}(\boldsymbol{R}) = \text{tr}\left((\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{H}'(\hat{\boldsymbol{R}} - \boldsymbol{R})^T\right),\ \boldsymbol{H}' = \boldsymbol{Q}^T\boldsymbol{P}^T\boldsymbol{HPQ}$
20: Add quantization results:$qlayer.add(\boldsymbol{W}_q, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\Sigma}, p, s)$
21: **end for**
22: Save the layer result: $QModule.layer[name] = qlayer$
23: **end for**
24: **return** $QModule$

In this section, we provide a detailed description of the LoPRo execution pipeline,

A.1 Algorithm pseudo-code

The workflow of LoPRo is outlined in Algorithm 1:

Stage A. Low-Rank Approximation under Calibration Set:

1. Perform forward inference on the calibration dataset to collect layer-wise input activations $\boldsymbol{X}$ corresponding to each weight matrix $\boldsymbol{W}$ (line 1-4).
2. Compute the importance-aware scaling vector with the activations $\boldsymbol{X}$ (line 5-6).
3. Apply the scaling to $\boldsymbol{W}$, obtaining scaled weight (line 7).
4. Draw a sketch vector and form exponentiation to the Gram matrix (line 8-10).
5. Calculate the rank-1 matrix according to Eq 13, and update the matrix for low-rank approximation (line 11-14).

Stage B. Structured Residual Rotation:

6. Compute the permutation array according to Eq 8, which groups columns by importance; then construct the corresponding permutation matrix $\boldsymbol{P}$ (line 15-17).
7. Apply the block-wise partial Hadamard transformation to the residual matrix to minimize quantization error (line 18).

Stage C. Quantization of Transformed Residual:

8. Quantize the rotated residual matrix $\boldsymbol{R}'$ by quantization tools (e.g., GPTQ for scalar quantization or GPTVQ for vector quantization) (line 19).
9. Save the final quantized components: $\boldsymbol{W}_q$, $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$, scaling vector $\boldsymbol{a}$, and permutation index $\boldsymbol{p}$, for deployment (line 20-22).

A.2 Average Bit-width

For a weight matrix $\boldsymbol{W} \in \boldsymbol{R}^{m\times n}$ with original precision d_o (in bits) and quantized to target d_q with group size g. Suppose the low-rank components of rank r, stored in precision d_r.

- For the quantized part, the bit-width of matrix W_q is d_q, and the average bit of scale is $\frac{d_o * m * n/g}{m*n}$. Then the average bits in this part is $d_q + \frac{d_o}{g}$.
- For the low-rank matrix: $\boldsymbol{U}$ and $\boldsymbol{V}$ are stored in d_r costs $\frac{d_r*(m+n)*r}{m*n}$. The singular is a diagonal matrix and can transform to a vector of size r that costs $\frac{r*d_0}{m*n}$.
- For the scale vector s in 1 the average bit is $\frac{d_o}{n}$ (one scale per column, n columns over mn weights, at d_o bits); the permutation p stores n integer indices at $\lceil \log_2 n \rceil$ bits each, i.e. $\frac{\log_2 n}{m}$ bits per weight. Both are negligible against d_q.

Combining these three items, the overall average compression bit d_C is

$$d_C = \overbrace{d_q + \frac{d_o}{g}}^{Quant\ \boldsymbol{W}} + \overbrace{\frac{rd_r}{n} + \frac{rd_r}{m}}^{\boldsymbol{U}\ \text{and}\ \boldsymbol{V}} + \overbrace{\frac{2d_o}{n}}^{\alpha\ \text{and}\ \boldsymbol{P}} + \overbrace{\frac{rd_o}{mn}}^{\boldsymbol{\Sigma}}. \tag{14}$$

Specifically, the last two terms in Eq. 14 ($2d_o/n$ for α and $\boldsymbol{P}$, and $rd_o/(mn)$ for $\boldsymbol{\Sigma}$) are negligible since r and $d_o \ll n$. The dominant low-rank overhead is $rd_r(1/n + 1/m)$. For a 7B model (n=m=4096, d_r=8, r=16): $16\times8(1/4096+1/4096) = 0.0625$ bits; for 13B (n=m=5120): $16\times8(2/5120) = 0.05$ bits. In practice, we set r=16 or 32 and store $\boldsymbol{U}$, $\boldsymbol{V}$ in *fp8*. This overhead further diminishes as model scale increases.

A.3 Time Complexity

The algorithm introduces two main part of latency during inference: low-rank matrix multiplication and reordering with rotation transformations. Assume the linear weight dimension is equal, and the input is $\boldsymbol{X} \in \mathbb{R}^{n\times b}$.

- For the original quantization linear layer $O = Dequant(\boldsymbol{W})\boldsymbol{X}$. The complexity is $\mathcal{O}((b+1)n^2)$.
- For the low-rank part, we first $\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{X}$ followed by $\boldsymbol{O} = \boldsymbol{U}(\boldsymbol{\Sigma}\boldsymbol{Y})$. Since $\boldsymbol{\Sigma}$ is diagonal, its multiplication takes linear time, and the overall complexity is $\mathcal{O}(2rnb)$.
- For the permutation and rotation part, we apply it to the input $\boldsymbol{X}$ to minimize computational overhead: $\tilde{\boldsymbol{X}} = \boldsymbol{Q}^T\mathbf{P}^T\boldsymbol{X}$. The permutation has complexity $\mathcal{O}(nb)$ and the rotation can be computed efficiently by the Fast Walsh-Hadamard Transform (Fino & Algazi, 1976) since the block size $b_H = 2^i$, running in $\mathcal{O}(nb\log b_H)$.

Together, the complexity *additional* to the $\mathcal{O}(n^2b)$ GEMM baseline is

$$\Delta C = \mathcal{O}\big(nb(2r + \log b_H)\big). \tag{15}$$

Here the permutation's own $\mathcal{O}(nb)$ cost is absorbed into the $\log b_H \geq 1$ term; the $2r$ term is the low-rank matmul ($\boldsymbol{W}_r\boldsymbol{X}$) and $\log b_H$ the Fast Walsh–Hadamard transform.

A.4 PERMUTATION

The construction of the rearrangement formula in Eq 8 is motivated by the following two key observations:

- For the residual matrix $\boldsymbol{R}$, after scaled low-rank approximation, columns corresponding to more important weight channels are approximated with higher accuracy, resulting in smaller absolute values. In other words, columns in $\boldsymbol{R}$ with smaller $\overline{|R_{\cdot j}|}$ correspond to more critical channels; preserving the quantization precision of these top-ranked important columns is crucial for overall accuracy. Sorting columns by $s_j = \boldsymbol{H}_{jj}/\overline{|R_{\cdot j}|}$ in *descending* order (Eq. 8) places the most important columns at the front, allowing the identity block to protect them from uniform quantization degradation.
- In loss optimization methods such as GPTQ, it is preferable to first quantize columns with larger quantization errors, followed by those with smaller errors. This ordering is determined by the diagonal elements of the Hessian matrix, $\mathrm{diag}(\boldsymbol{H})$.

Combining these two principles, we derive the final column ranking function as presented in Eq 8. Under the permutation, we can preserve the quantization precision of the most critical columns while jointly optimizing the quantization of smoothly varying and less critical columns, thereby achieving superior overall quantization performance.

B PROOFS

B.1 NOTATIONS AND SETUP

Let $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ be a linear weight, and after scaled low-rank decomposition (Eq. 4) write $\boldsymbol{W} = \boldsymbol{W}_r + \boldsymbol{R}$ with $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ the residual to be quantized. The proxy Hessian is $\boldsymbol{H} = \mathbb{E}_{\boldsymbol{X}}[\boldsymbol{X}\boldsymbol{X}^\top] \in \mathbb{R}^{n\times n}$ and its diagonal $\boldsymbol{H}_{jj}$ is the importance of column j as measured by activations. The quantization error is $\boldsymbol{E} := \hat{\boldsymbol{R}} - \boldsymbol{R}$ and the layer-wise loss admits the expansion

$$\mathcal{L}(\hat{\boldsymbol{R}}) = \mathbb{E}_{\boldsymbol{X}}\left[\|(\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{X}\|_F^2\right] = \mathrm{tr}\left(\boldsymbol{E}\,\boldsymbol{H}\,\boldsymbol{E}^\top\right) = \underbrace{\textstyle\sum_{j=1}^{n} \boldsymbol{H}_{jj}\|\boldsymbol{E}_{\cdot j}\|_2^2}_{\mathcal{L}_{\mathrm{diag}}} + \underbrace{\textstyle\sum_{j\neq l} \boldsymbol{H}_{jl}\,\boldsymbol{E}_{\cdot j}^\top \boldsymbol{E}_{\cdot l}}_{\text{cross terms}}. \quad (16)$$

Throughout this appendix we work with the diagonal term $\mathcal{L}_{\mathrm{diag}}$, which is the standard proxy used in GPTQ/QuaRot-style analyses and is tight once the residual has small column-wise cross-correlation – precisely the effect that block Hadamard is designed to induce within each block.

Per-column step size. Under round-to-nearest quantization with group size g and b bits, the per-column squared error satisfies

$$\mathbb{E}\left[\|\boldsymbol{E}_{\cdot j}\|_2^2\right] = \tfrac{m\,\varepsilon_j^2}{12}, \quad \varepsilon_j = \tfrac{2}{2^b-1}\max_i |\boldsymbol{R}_{ij}|, \quad (17)$$

up to lower-order corrections from the intra-group distribution.

B.2 PROOF OF PROPOSITION 1 (INTRA-BLOCK ROTATION REDUCES WEIGHTED LOSS)

Let $\mathcal{B} \subseteq \{1,\dots,n\}$ be a block of b_H consecutive columns after permutation, satisfying (A1): $\max_{j\in\mathcal{B}} \varepsilon_j / \min_{j\in\mathcal{B}} \varepsilon_j \le \rho$ and (A2): $\max_{j\in\mathcal{B}} \boldsymbol{H}_{jj} / \min_{j\in\mathcal{B}} \boldsymbol{H}_{jj} \le \eta$. Denote the rotated block by $\boldsymbol{R}_{\mathcal{B}} := \boldsymbol{R}_{\cdot\mathcal{B}}\boldsymbol{H}_{\mathrm{wal}}^{(b_H)}$.

Finally, (A3) *post-permutation incoherence*: the deterministic block $\boldsymbol{H}_{\mathrm{wal}}^{(b_H)}$ of Eq. 9 (used by Algorithm 1) spreads each block's energy so that the peak-to-RMS ratio of the rotated steps is bounded, $\kappa := \max_{k\in\mathcal{B}} \varepsilon'_k/\bar{\varepsilon}_{\mathcal{B}} \le \kappa_0$ (with $\bar{\varepsilon}_{\mathcal{B}}$ the common post-rotation RMS step of Step 1). Unlike the randomized-sign concentration invoked by QuIP#/QuaRot, (A3) is a property of the *actual* permuted residual rather than a worst-case guarantee over adversarial inputs: it holds whenever a block is outlier-dominated rather than aligned with a single Hadamard row, and $\kappa_0 \approx \sqrt{2\log(b_H m)}$ is the value predicted by that classical argument.

Step 1: post-rotation step homogenization. Because $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ is orthogonal with entries $\pm 1/\sqrt{b_H}$, for each input row i the rotated entries $(\boldsymbol{R}_{\mathcal{B}})_{ik} = \sum_{j\in\mathcal{B}} h_{jk}\boldsymbol{R}_{ij}$ have *block-averaged* second moment $\frac{1}{b_H}\sum_{k\in\mathcal{B}}(\boldsymbol{R}_{\mathcal{B}})_{ik}^2 = \frac{1}{b_H}\sum_{j\in\mathcal{B}}\boldsymbol{R}_{ij}^2$, which is *exact* by orthogonality ($\sum_k h_{jk}h_{j'k} = \delta_{jj'}$); for an individual k the value concentrates around this mean under the incoherence assumption (A3). Averaging over i, every rotated column thus shares the same variance $\bar{v}_{\mathcal{B}} := \frac{1}{b_H}\sum_{j\in\mathcal{B}}\|\boldsymbol{R}_{\cdot j}\|_2^2/m$, and consequently the same effective step (a variance-proxy step; the rigorous max-step loss is recovered via Δ_{inc} in Step 3)

$$\varepsilon'_k \;\approx\; \bar{\varepsilon}_{\mathcal{B}} := \sqrt{\tfrac{1}{b_H}\textstyle\sum_{j\in\mathcal{B}}\varepsilon_j^2}, \quad k\in\mathcal{B}. \tag{18}$$

Step 2: post-rotation Hessian diagonal. The block-restricted rotated Hessian is $\boldsymbol{H}'_{\mathcal{B}} = (\boldsymbol{H}_{\text{wal}}^{(b_H)})^\top \boldsymbol{H}_{\mathcal{B}}\,\boldsymbol{H}_{\text{wal}}^{(b_H)}$, whose diagonal expands as $\boldsymbol{H}'_{kk} = \underbrace{\sum_j h_{jk}^2\boldsymbol{H}_{jj}}_{=\bar{\boldsymbol{H}}_{\mathcal{B}}} + \sum_{j\neq l} h_{jk}h_{lk}\boldsymbol{H}_{jl}$ with $\bar{\boldsymbol{H}}_{\mathcal{B}} := \frac{1}{b_H}\sum_{j\in\mathcal{B}}\boldsymbol{H}_{jj}$ (using $h_{jk}^2 = 1/b_H$). The exact diagonal term $\bar{\boldsymbol{H}}_{\mathcal{B}}$ is *deterministic*—it uses only $h_{jk}^2 = 1/b_H$, no randomness—while the *dropped* off-diagonal cross term $\sum_{j\neq l} h_{jk}h_{lk}\boldsymbol{H}_{jl}$ has magnitude $\leq \frac{1}{b_H}\sum_{j\neq l}|\boldsymbol{H}_{jl}|$ and is absorbed into the (A1)(A2) slack of Step 4; this is the diagonal (block-restricted) approximation, consistent with the diagonal proxy already used in Eq. 5.

Step 3: variance-proxy loss difference (exact identity). Substituting (17), (18) and Step 2 into $\mathcal{L}_{\text{diag}}$ defines the *variance-proxy* losses

$$\mathcal{L}_{\text{orig}}^{(\mathcal{B}),\text{var}} = \tfrac{m}{12}\textstyle\sum_{j\in\mathcal{B}}\boldsymbol{H}_{jj}\varepsilon_j^2, \quad \mathcal{L}_{\text{rot}}^{(\mathcal{B}),\text{var}} = \tfrac{m}{12}\,b_H\,\bar{\boldsymbol{H}}_{\mathcal{B}}\,\bar{\varepsilon}_{\mathcal{B}}^2 = \tfrac{m}{12}\,\bar{\boldsymbol{H}}_{\mathcal{B}}\textstyle\sum_{j\in\mathcal{B}}\varepsilon_j^2.$$

Their difference is *exactly* the block covariance, with no remainder term:

$$\mathcal{L}_{\text{orig}}^{(\mathcal{B}),\text{var}} - \mathcal{L}_{\text{rot}}^{(\mathcal{B}),\text{var}} = \tfrac{m}{12}\underbrace{\textstyle\sum_{j\in\mathcal{B}}\big(\boldsymbol{H}_{jj}-\bar{\boldsymbol{H}}_{\mathcal{B}}\big)\big(\varepsilon_j^2-\bar{\varepsilon}_{\mathcal{B}}^2\big)}_{=:\,\text{Cov}_{\mathcal{B}}(\boldsymbol{H},\varepsilon^2)}. \tag{19}$$

The *true* (max-step) rotated loss $\mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max}$ is lower than this variance proxy; we *define* the gap as the classical incoherence (peak-shrinkage) gain

$$\Delta_{\text{inc}}(b_H) \;:=\; \mathcal{L}_{\text{rot}}^{(\mathcal{B}),\text{var}} - \mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max} \;\geq\; 0, \tag{20}$$

which is non-negative *deterministically* under (A3): the rotated peak step obeys $\varepsilon'_{\max} \leq \kappa_0\,\bar{\varepsilon}_{\mathcal{B}}$ with $\kappa_0 \approx \sqrt{2\log(b_H m)}$, strictly smaller than the pre-rotation peak $\max_{j\in\mathcal{B}}\varepsilon_j$ whenever $\boldsymbol{R}_{\cdot\mathcal{B}}$ is outlier-dominated—the incoherence effect exploited by QuIP/QuaRot, here guaranteed by (A3) for the deterministic $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ rather than by random signs. Combining the identity and the definition:

$$\mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max} = \mathcal{L}_{\text{orig}}^{(\mathcal{B}),\text{var}} \;-\; \tfrac{m}{12}\,\text{Cov}_{\mathcal{B}}(\boldsymbol{H},\varepsilon^2) \;-\; \Delta_{\text{inc}}(b_H). \tag{21}$$

Step 4: bounding the covariance under (A1)(A2). Because all ε_j^2 lie within a factor ρ^2 of one another and $\min_j\varepsilon_j^2 \leq \bar{\varepsilon}_{\mathcal{B}}^2 \leq \max_j\varepsilon_j^2$, (A1) gives $|\varepsilon_j^2 - \bar{\varepsilon}_{\mathcal{B}}^2| \leq (\rho^2-1)\bar{\varepsilon}_{\mathcal{B}}^2$; similarly (A2) gives $|\boldsymbol{H}_{jj} - \bar{\boldsymbol{H}}_{\mathcal{B}}| \leq (\eta-1)\bar{\boldsymbol{H}}_{\mathcal{B}}$. Applying these two enclosures term-wise to (19) yields the self-contained bound

$$\big|\text{Cov}_{\mathcal{B}}(\boldsymbol{H},\varepsilon^2)\big| \;\leq\; (\eta-1)(\rho^2-1)\,\bar{\boldsymbol{H}}_{\mathcal{B}}\,b_H\,\overline{\varepsilon^2}_{\mathcal{B}}.$$

Substituting into (21) yields the block-wise statement of Proposition 1:

$$\mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max} \;\leq\; \mathcal{L}_{\text{orig}}^{(\mathcal{B}),\text{var}} \;-\; \Delta_{\text{inc}}(b_H) \;+\; \tfrac{m}{12}(\eta-1)(\rho^2-1)\cdot\bar{\boldsymbol{H}}_{\mathcal{B}}\,b_H\,\overline{\varepsilon^2}_{\mathcal{B}}, \tag{22}$$

whose slack term vanishes as $\rho,\eta\to 1$, giving $\mathcal{L}_{\text{rot}}^{(\mathcal{B}),\max} \leq \mathcal{L}_{\text{orig}}^{(\mathcal{B}),\text{var}} - \Delta_{\text{inc}}(b_H)$ in that limit.

Remark (failure probability and magnitude of Δ_{inc}). Under (A3) the peak-shrinkage of Step 3 holds *uniformly* over rotated columns for the deterministic $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ used in Eq. 9, with the adopted, not directly measured, concentration factor κ_0; the post-rotation step then obeys $\varepsilon'_{\max} \leq \frac{2}{2^b-1}\kappa_0\sqrt{\bar{v}_{\mathcal{B}}}$ (choosing $\kappa_0 = \sqrt{2\log(b_H m/\delta)}$ recovers the randomized-Hadamard high-probability form as a

special case), so $\Delta_{\text{inc}}(b_H) \geq \frac{m}{12}\bar{\boldsymbol{H}}_{\mathcal{B}} b_H \left(\overline{\varepsilon^2}_{\mathcal{B}} - (\varepsilon'_{\max})^2\right)_+$. A net decrease in (22) therefore follows whenever this gain exceeds the slack $\frac{m}{12}(\eta-1)(\rho^2-1)\bar{\boldsymbol{H}}_{\mathcal{B}} b_H \overline{\varepsilon^2}_{\mathcal{B}}$—precisely the regime $\rho, \eta \approx 1$ that □

Scope of the guarantee (relation to the real loss). Proposition 1 is a statement about the *variance-proxy, diagonal* loss equation 5 under (A1)–(A3); it is not a theorem on the exact max-step loss. We therefore read the full-rotation comparison as a prediction of this proxy rather than a strict theorem. (i) *(A3) is assumed, not proven.* The deterministic block $\boldsymbol{H}_{\text{wal}}^{(b_H)}$ carries no random signs, so the peak-to-RMS bound $\kappa \leq \kappa_0$ is adopted at its classically predicted value $\kappa_0 \approx \sqrt{2\log(b_H m)}$ rather than measured per layer; a direct empirical κ distribution is left to future work, and (A3) is supported only qualitatively by the anomaly of Table 5 and the fixed-seed simulation of Fig. 2. (ii) *Full rotation also has an incoherence gain.* A full Hadamard likewise shrinks peak steps, contributing its own $\Delta_{\text{inc}}^{(\text{full})} \geq 0$ that the proxy $\mathcal{L}_{\text{rot}} \approx n\bar{\boldsymbol{H}}\bar{\varepsilon}^2$ omits; the net "full rotation increases the loss" conclusion therefore holds for the proxy whenever $\Delta_{\text{inc}}^{(\text{full})}$ does not offset the covariance term, and it is precisely the resulting *net* effect—the $8.4 \to 9.49$ of Table 5—that permutation-based partial rotation avoids.

Discussion: why partial beats full. Proposition 1 makes no useful claim when $\rho\eta \gg 1$, i.e., when a block mixes columns of vastly different difficulty. This is precisely what a **full** Hadamard rotation does: it takes $\mathcal{B} = \{1, \ldots, n\}$, whose ρ, η reflect the full heterogeneity of the residual created by low-rank+scaling, driving the slack term to dominate Δ_{inc}. Formally, applying Chebyshev's sum inequality to the oppositely-sorted sequences $\{\boldsymbol{H}_{jj}\}$ and $\{\varepsilon_j^2\}$ on the whole matrix gives

$$\mathcal{L}_{\text{orig}} = \textstyle\sum_j \boldsymbol{H}_{jj}\varepsilon_j^2 \;\leq\; n\,\bar{\boldsymbol{H}}\,\overline{\varepsilon^2} \;=\; \mathcal{L}_{\text{rot}},$$

so a full rotation strictly **increases** the weighted loss whenever $\text{Cov}(\boldsymbol{H}_{jj}, \varepsilon_j^2) < 0$ – reproducing the empirical anomaly in Table 5 (PPL $8.4 \to 9.49$ at 2-bit on LLaMA-2 7B). LoPRo's design resolves this: (i) the leading b_I columns are placed in an identity block, so their high-$\boldsymbol{H}$, low-ε contribution is preserved (loss unchanged, no contamination); (ii) the remaining columns are permutation-grouped so that each rotated block has $\rho\eta \approx 1$, and Proposition 1 applies block-wise with a strict net decrease. Summing over all blocks yields $\mathcal{L}_{\text{LoPRo}} \leq \mathcal{L}_{\text{orig}} - \sum_{\mathcal{B}} \Delta_{\text{inc}}(b_H) + o(1)$, i.e., the total incoherence gain is fully harvested.

B.3 Proof of Lemma 1 (Rotated-Hessian loss)

Recall from Eq. (9) that the quantizer operates on the *rotated* residual $\boldsymbol{R}' = \boldsymbol{R}\boldsymbol{P}\boldsymbol{Q}$ with $\boldsymbol{P}, \boldsymbol{Q}$ both orthogonal. Write $\hat{\boldsymbol{R}}' := \mathsf{Q}(\boldsymbol{R}')$ for its quantization in the rotated frame, and $\hat{\boldsymbol{R}} := \hat{\boldsymbol{R}}'\boldsymbol{Q}^\top\boldsymbol{P}^\top$ for the reconstruction mapped back to the original frame; these are two *distinct* objects related by the fixed orthogonal change of basis. Define the *rotated-frame error*

$$\hat{\boldsymbol{E}} \;:=\; \hat{\boldsymbol{R}}' - \boldsymbol{R}' \;=\; \hat{\boldsymbol{R}}' - \boldsymbol{R}\boldsymbol{P}\boldsymbol{Q}.$$

Setting $\tilde{\boldsymbol{X}} := \boldsymbol{Q}^\top\boldsymbol{P}^\top\boldsymbol{X}$ (which is invertible with $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{Q}\tilde{\boldsymbol{X}}$), the original-frame reconstruction error telescopes as $(\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{X} = (\hat{\boldsymbol{R}}'\boldsymbol{Q}^\top\boldsymbol{P}^\top - \boldsymbol{R})\boldsymbol{P}\boldsymbol{Q}\tilde{\boldsymbol{X}} = \hat{\boldsymbol{E}}\tilde{\boldsymbol{X}}$, and the layer loss in Eq. (10) becomes

$$\begin{aligned}\mathcal{L}(\hat{\boldsymbol{R}}) &= \mathbb{E}_{\boldsymbol{X}}\left[\|(\hat{\boldsymbol{R}} - \boldsymbol{R})\boldsymbol{X}\|_F^2\right] = \mathbb{E}_{\tilde{\boldsymbol{X}}}\left[\|\hat{\boldsymbol{E}}\tilde{\boldsymbol{X}}\|_F^2\right] \\ &= \text{tr}\big(\hat{\boldsymbol{E}}\underbrace{\mathbb{E}[\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top]}_{=\boldsymbol{H}'}\hat{\boldsymbol{E}}^\top\big),\end{aligned}$$

where $\boldsymbol{H}' = \boldsymbol{Q}^\top\boldsymbol{P}^\top\boldsymbol{H}\,\boldsymbol{P}\,\boldsymbol{Q}$ follows from $\mathbb{E}[\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^\top] = \boldsymbol{Q}^\top\boldsymbol{P}^\top\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top]\boldsymbol{P}\boldsymbol{Q}$. The two factors $\boldsymbol{P}, \boldsymbol{Q}$ are computed once per layer at quantization time and are fixed during GPTQ column-wise optimization on $\boldsymbol{R}'$, so the standard GPTQ update applies verbatim to $\boldsymbol{R}'$ under the transformed Hessian $\boldsymbol{H}'$. □

C Additional Implementation Details

Set up: Experiments with the MoE model Mixtral-8x7B were conducted on a single NVIDIA A800 80GB GPU, while all other experiments were performed on a single NVIDIA A100 40GB GPU. The

difference in hardware only leads to minor variations in quantization costs and inference latency; all accuracy metrics and other numerical results remain identical across platforms, ensuring fair and consistent evaluation.

Calibration: We use a calibration dataset consisting of 128 randomly sampled sequences, each containing 2048 tokens, from c4 (Raffel et al., 2020), a sampling strategy shown to be effective in OmniQuant (Shao et al., 2023) and AffineQuant (Ma et al., 2024).

Evaluation: For the perplexity (PPL) evaluation, we set the context length to match the maximum sequence length used during model training: 4096 for LLaMA-2 and 8192 for LLaMA-3. In zero-shot evaluations, we report the `acc` metric (rather than `acc_norm`) from the `lm-eval-harness` (Gao et al., 2024). All results are rounded to one or two decimal places as appropriate. Here are brief introductions to the zero-shot datasets:

- **ARC-Challenge (AC)** and **ARC-Easy (AE)**(Boratko et al., 2018): The AI2 Reasoning Challenge (ARC) dataset consists of multiple-choice science questions from grade school level. ARC-Challenge contains questions that are difficult for both retrieval and word co-occurrence methods, focusing on genuine reasoning, while ARC-Easy includes questions that are more amenable to simpler methods.
- **PIQA (QA)**(Bisk et al., 2020): The Physical Interaction: Question Answering dataset evaluates a model's ability to understand physical commonsense reasoning. It presents questions about the physical properties and interactions of everyday objects, requiring models to choose the most plausible solution between two options.
- **Winogrande (WI)**(Sakaguchi et al., 2021): Winogrande is a large-scale dataset designed to test commonsense reasoning, specifically tackling the Winograd Schema Challenge. It features a new adversarial filtering approach to create difficult multiple-choice questions that require understanding of context and pronoun resolution.
- **BoolQ (BQ)**(Clark et al., 2019): The Boolean Questions dataset contains yes/no questions derived from real search queries paired with paragraphs from Wikipedia. The task is to determine the correct boolean answer based on the information in the given passage.
- **Hellaswag (HS)**(Zellers et al., 2019): The HELLA SWAG dataset evaluates commonsense inference in sentence completion tasks. Given a partial sentence describing a situation, models must select the most plausible continuation from multiple choices, with adversarially generated distractors making the task challenging.
- **OpenbookQA (OB)**(Mihaylov et al., 2018): OpenBookQA is a multiple-choice question-answering dataset that uses a "fact" from an open book as a basis for questions requiring both the provided fact and general knowledge to answer, aiming to test deeper understanding and reasoning.
- **MathQA (MQ)**(Amini et al., 2019): MathQA is a dataset of math word problems paired with annotated solutions in a Python-like programming language. It is designed to evaluate and improve the ability of models to perform multi-step mathematical reasoning and solve quantitative problems.

Implementation: In the main pipeline of LoPRo, during the R1SVD low-rank approximation phase, we set the rank size r to 16 and perform 8 iterations it. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are stored in the 8-bit floating-point format *e4m3*, as they are orthogonal matrices with a narrow dynamic range in $[-1, 1]$; compared to *e5m2*, *e4m3* provides higher precision for such distributions. In the partial rotation phase under rearrangement, the block sizes for the identity matrix and the rotation matrix, denoted as b_I and b_H, are both set to 256. A detailed ablation study of these hyperparameters and strategies is presented in Appendix D. In the implementation of the quantize residual matrix, for vector-level quantization, we do not employ advanced techniques such as block-wise scaling or learnable codebooks. Instead, we adopt the simplest form: a randomly initialized codebook combined with quantization based on the rotated-Hessian loss of Lemma 1 (App. B.3). Furthermore, we use $4D$ codebook for 2-bit quantization and $2D$ codebook for 3-bit quantization, following the optimal configuration used in GPTVQ. However, in comparison to our quantization scheme in LoPRo, for the GPTVQ baseline in main evaluation, we respect the integrity of the original method and only distinguishing whether fine-tuning (FT) is applied while adopting all other techniques (e.g., learnable

codebooks, block-wise scaling) as specified in the original paper to ensure fair comparison using their best-reported configurations.

Baseline: Following the taxonomy introduced in § 2, we select representative and state-of-the-art baselines from each strategy category (Tag in) to ensure comprehensive and fair comparisons. Our selection principle is twofold: (1) choose the strongest-performing method within each category, and (2) ensure full coverage of all major strategy tags. Specifically, for rotation-based quantization, we select QuIP#(Tseng et al., 2024a) — with the stronger performance than rotation methods like QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2025), and DuQuant (Lin et al., 2024a), which underperform QuIP# at sub-3-bit settings. For low-rank fine-tuning assisted quantization, we include RILQ (Lee et al., 2025) and CALDERA (Saha et al., 2024), which represent the most competitive results in this category. For low-rank compensation quantization, we select LQER (Zhang et al., 2024a) as both methods leverage low-rank components to recover quantization error. For loss and clipping method, we choose OmniQuant (Shao et al., 2023). Finally , MoEQuant (Chen et al., 2025) is chosen as the baseline in MoE model quantization. Additionally, we include GPTQ (Frantar et al., 2023) and GPTVQ (Van Baalen et al., 2024), which are leveraged within LoPRo to enable ablation and component-wise analysis. The multidimensional baseline selection ensures that LoPRo is evaluated against the strongest existing approaches and providing a holistic assessment of its effectiveness.

Fine tuning: We adopt CALDERA and RILQ, two state-of-the-art LoRA fine-tuning PTQ methods as baselines, and apply RILQ as the backend for fine-tuning in our LoPRo. In Table 2, both compared methods are implemented with their optimal configurations as reported in the original papers, specifically using QuIP# as the pre-quantization method prior to fine-tuning; in contrast, the LoPRo$_v$-FT rows use LoPRo$_v$ itself as the pre-quantizer, so the comparison isolates the fine-tuning backends under their respective best pre-quantization.

D Extended Ablation Studies

Overall: In addition to the ablation on rotation strategy in §4.5, we conducted a comprehensive set of ablation studies to validate the effectiveness of key components and the hyperparameter chosen in LoPRo. Based on empirical evidence from these experiments, we finalize the following hyperparameter settings:

- **Low-Rank Approximation Method:** R1SVD
  - Low-rank approximation rank: $r = 16$
  - Number of R1SVD iterations: $iter = 8$
- **Rotation Block Parameters:** $b_I = b_H = 256$
- **Calibration Dataset:** 128 randomly sampled sequences from c4 dataset.

These configurations are supported by ablation results across multiple models and tasks, demonstrating robustness and consistent performance. Unless otherwise specified, all experiments in this work use the above settings to ensure fairness and reproducibility. The final strategies adopted in the experiment are marked with dark colors.

D.1 Ablation on rank

In LoPRo, we performed an ablation study on the choice of rank size, with results shown in Table 7. The results indicate that quantization accuracy generally improves as the rank increases. However, under vector quantization and 3-bit scalar quantization, the accuracy gains from increasing rank are marginal, suggesting that even small ranks are sufficient to capture the dominant structure in these settings.

Using an excessively large rank yields diminishing returns in accuracy while significantly reducing the compression ratio and increasing memory footprint. Therefore, for models of various sizes, we find that a rank of 16 strikes an optimal balance: it achieves high quantization accuracy with minimal overhead—adding approximately 3% extra memory cost—while maintaining fast decomposition and inference. This overhead further decreases as model scale increases, making $r = 16$ a practical and scalable choice across architectures.

Table 7: Ablation study on rank selection in LoPRo. PPL (wiki2,ctx=4096) and Accuracy are measured under different settings. 'r.bit' denote for the average bit of low-rank component. Abbreviations for zero-shot tasks follow those defined in §4.1 and 'Avg' stands for the average accuracy of four tasks.

model	bit	method	rank	r.bit	PPL	AC	AE	QA	WI	Avg
LLaMA2-7B	2	LoPRo	8	0.02	7.51	29.4	63.9	70.6	66.8	57.7
			16	0.06	7.39	31.2	62.8	71.1	63.8	57.2
			32	0.10	7.35	29.9	64.0	71.2	63.7	57.2
			64	0.19	7.30	31.8	65.2	70.8	64.3	58.0
		LoPRo$_v$	8	0.02	6.56	33.6	69.4	73.0	65.7	60.4
			16	0.06	6.53	34.6	69.0	72.7	66.5	60.7
			32	0.10	6.54	34.9	69.5	73.6	65.8	60.9
			64	0.19	6.49	34.0	69.5	73.1	66.2	60.7
	3	LoPRo	8	0.02	5.44	41.3	74.8	76.7	68.4	65.3
			16	0.06	5.43	41.0	74.9	76.3	68.9	65.3
			32	0.10	5.42	40.6	74.4	76.8	68.4	65.1
			64	0.19	5.41	41.0	75.0	77.3	69.1	65.6
		LoPRo$_v$	8	0.02	5.46	42.0	75.3	77.5	67.1	65.5
			16	0.06	5.45	41.0	74.8	76.7	69.1	65.4
			32	0.10	5.45	41.6	75.0	77.0	68.9	65.6
			64	0.19	5.44	39.7	74.5	77.3	68.0	64.9
LLaMA2-13B	2	LoPRo	8	0.02	6.49	33.3	66.9	71.2	65.5	59.2
			16	0.05	6.48	33.6	69.0	72.4	66.3	60.3
			32	0.08	6.40	33.9	68.6	72.9	66.1	60.4
			64	0.15	6.40	35.3	69.4	74.3	67.9	61.7
		LoPRo$_v$	8	0.02	5.87	38.3	73.1	74.4	68.7	63.6
			16	0.05	5.79	38.8	74.2	75.4	68.0	64.1
			32	0.08	5.76	38.3	73.7	74.6	67.6	63.6
			64	0.15	5.71	38.6	73.1	75.6	68.2	63.9
	3	LoPRo	8	0.02	4.85	46.1	78.0	78.0	70.4	68.1
			16	0.05	4.84	44.9	78.7	78.5	71.1	68.3
			32	0.08	4.84	44.1	77.3	78.0	72.4	67.9
			64	0.15	4.81	46.2	78.2	78.5	72.9	68.9
		LoPRo$_v$	8	0.02	4.91	43.3	76.8	77.6	71.9	67.4
			16	0.05	4.87	44.5	77.4	78.5	71.0	67.9
			32	0.08	4.86	44.8	77.3	78.2	70.0	67.6
			64	0.15	4.85	44.6	77.9	78.2	70.2	67.7

D.2 Ablation on Low-rank Decomposition

We conduct an ablation study on R1SVD, with results presented in Table 8. Due to fluctuations in GPU computational performance, the reported execution times exhibit minor variability. Nevertheless, the results clearly show that the runtime of R1SVD scales nearly linearly with model size—requiring only about 1 minute for the 7B model and approximately 2 minutes for the 13B model.

The computational cost of R1SVD is dominated by the sketch: each rank-1 component requires $2\,iter + 2$ GEMV (General Matrix-Vector Multiplication) operations ($2\,iter$ for the $(\boldsymbol{W}_s\boldsymbol{W}_s^\top)^{iter}\boldsymbol{W}_s\boldsymbol{v}$ power iteration plus two for $\boldsymbol{p} = \boldsymbol{W}_s^\top\boldsymbol{y}$), so extracting rank r costs $\approx r(2\,iter + 2)$ GEMV in total (e.g. r=16, $iter$=8 $\Rightarrow$ 288 per layer). Since each GEMV is far cheaper than a full forward pass, this remains a small fraction of the overall quantization cost and scales nearly linearly with model size.

In contrast, SVD-based decomposition is significantly slower: it takes 14 minutes for the 7B model and over 30 minutes for the 13B model. Moreover, SVD suffers from poor GPU parallelization, and most standard libraries (e.g., cuSOLVER) do not support *fp16* computation. As a result, SVD runs exclusively in *fp32* or higher precision, further increasing its computational burden. Critically, R1SVD maintains high numerical accuracy despite using mixed *fp16*/*fp8* arithmetic. This is because the precision loss from each *fp16*-to-*fp8* conversion is compensated in subsequent iterations, akin

to the error feedback mechanism in OBS (Optimal Brain Surgeon). Consequently, the accuracy of R1SVD closely approaches that of full-precision SVD.

In summary, the R1SVD achieves approximation quality comparable to that of SVD while being orders of magnitude faster. Given the relative tolerance to numerical error in deep learning compared to scientific computing, we consider that R1SVD holds strong potential for broad application in efficient model compression and large-scale training.

Table 8: Ablation study on low-rank method and bit in LoPRo. 'LoBit' stand for the precision of $\boldsymbol{U}, \boldsymbol{V}$ in low-rank matrix. Time$_{tot}$ and Time$_{low}$ respectively represent the total execution time of the algorithm and the execution time of the low-rank approximation.

Model	Method	LoBit	LoRA	Time$_{tot}$	Time$_{low}$	PPL	AC	AE	QA	WI
LLaMA2-7B	LoPRo	8	R1SVD	26.4m	0.8m	7.39	31.2	62.8	71.1	63.8
			SVD	42.2m	14.0m	7.44	31.1	62.0	69.6	64.6
		16	R1SVD	27.3m	1.2m	7.38	30.0	64.1	71.5	62.4
			SVD	42.3m	14.0m	7.40	31.4	62.2	70.0	64.5
	LoPRo$_v$	8	R1SVD	31.7m	1.1m	6.53	34.6	69.0	72.7	66.5
			SVD	47.2m	14.2m	6.52	35.0	67.9	73.2	66.0
		16	R1SVD	31.5m	1.1m	6.55	35.5	70.7	73.7	66.2
			SVD	47.2m	14.0m	6.56	34.8	70.5	73.9	64.7
LLaMA2-13B	LoPRo	8	R1SVD	45.1m	2.2m	6.48	33.6	69.0	72.4	66.3
			SVD	1.4h	31.2m	6.54	34.0	68.2	72.1	66.3
		16	R1SVD	45.8m	3.2m	6.52	32.8	66.2	71.7	65.2
			SVD	1.4h	31.4m	6.48	33.5	69.3	72.5	64.6
	LoPRo$_v$	8	R1SVD	56m	2.1m	5.79	38.8	74.2	75.4	68.0
			SVD	1.6h	31.8m	5.88	37.6	72.4	75.4	67.8
		16	R1SVD	57m	2.8m	5.77	38.6	73.5	75.0	67.8
			SVD	1.6h	31.7m	5.75	38.8	73.5	74.9	67.9

D.3 Ablation on iteration

In the R1SVD algorithm, the sketch computation involves iteratively applying the $S^*\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{A}^*)^{it}$ operation in Eq. 13 for *it* times (corresponding to Line 13 in Algorithm 1). A higher iteration count improves approximation accuracy but incurs additional computational cost.

To evaluate the impact of this parameter, we present the performance of LoPRo under different iteration values in Table 9. The results show that quantization accuracy improves with increasing *it* and eventually plateaus. For the LLaMA-2 7B model, the accuracy stabilizes when $it \geq 8$, with perplexity (PPL) fluctuating by at most 0.02 — indicating diminishing returns beyond this point.

Furthermore, although each additional iteration increases the low-rank approximation time, this stage constitutes only a small fraction of the overall quantization pipeline. For instance, even with 16 iterations, the R1SVD step takes about 1.6 minutes, making it highly efficient in practice. Therefore, we set *it* = 8 as the default in all other experiments, achieving near-optimal accuracy while maintaining high computational efficiency and scalability.

D.4 Ablation on block size

In LoPRo's rotation stage, we introduce two block size parameters: b_I and b_H, which respectively denote the block size of the identity matrix in the upper matrix and the block size of the Hadamard-Walsh (Hwal) matrix in the lower matrix of the partial block rotation matrix.

We evaluate the impact of different block configurations on the LLaMA-2 7B model, with results presented in Table 10. Since $Hwal$ block size must satisfy $b_H \in \mathbb{R}^{2^i}$, and the second dimension of weight matrices (e.g., in MLP layers) is typically a multiple of 128, we restrict our test cases to $256 > b_I \geq b_H > 64$. This constraint ensures that: (1) b_H remains a power of two, (2) the total number of blocks is an integer, and (3) since the MLP layer dimensions in LLaMA-2 7B are not divisible by b_H when $i > 9$ ($b_H = 512$). We further enforce b_I are b_H multiples of 64 to guarantee valid integer partitioning.

Table 9: Ablation study on iteration it (R1SVD - Eq 13) in LLaMA2-7B. Time$_{low}$ stands for the execution time of low-rank approximation.

Method	Bit	Iteration	PPL	AC	AE	QA	WI	Time$_{low}$
LoPRo	2	1	7.44	31.0	64.9	70.0	64.1	0.1m
		2	7.51	29.8	63.9	69.9	64.0	0.2m
		4	7.41	30.3	64.4	70.6	64.1	0.4m
		8	7.39	31.2	62.8	71.1	63.8	0.8m
		16	7.40	31.3	62.0	71.1	63.5	1.6m
		32	7.40	31.3	62.1	71.6	64.1	3.2m
	3	1	5.43	41.1	74.8	76.4	68.0	0.1m
		2	5.42	41.0	73.9	77.2	67.9	0.2m
		4	5.43	40.5	74.3	77.1	69.1	0.4m
		8	5.43	41.0	74.9	76.3	68.9	0.8m
		16	5.42	41.2	74.2	77.3	67.9	1.6m
		32	5.42	41.0	74.6	76.6	67.9	3.2m
LoPRo$_v$	2	1	6.58	34.5	70.5	73.5	66.7	0.1m
		2	6.56	34.0	68.7	73.7	64.6	0.2m
		4	6.54	34.0	70.8	73.7	65.2	0.4m
		8	6.53	34.6	69.0	72.7	66.5	0.8m
		16	6.55	34.2	70.8	73.1	65.8	1.6m
		32	6.53	34.0	70.8	73.7	65.9	3.2m
	3	1	5.46	41.1	75.1	76.1	67.9	0.1m
		2	5.45	39.6	74.9	76.7	68.0	0.2m
		4	5.45	39.9	74.6	76.8	67.8	0.4m
		8	5.45	41.0	74.8	76.7	69.1	0.8m
		16	5.45	41.6	75.1	77.3	69.8	1.6m
		32	5.45	39.9	74.9	76.4	67.8	3.2m

The results indicate that block size parameters have a measurable, though moderate, impact on quantization accuracy. Specifically, smaller block sizes (e.g., $b_I, b_H < 128$) lead to slightly degraded performance, while configurations with block sizes larger than 128 yield comparable accuracy — with $b_I = b_H = 256$ among the best (PPL 7.39, numerically on par with the 7.38 of $(b_I, b_H) = (256, 128)$).

This behavior can be attributed to two factors: (1) When b_I is too small, the more important channels (typically concentrated in the leading segment after permutation) are subjected to excessive rotation, which disrupts their numerical structure and increases quantization error. (2) Smaller rotation blocks do not adequately balance the distribution within each block, as they cannot effectively smooth out the quantization error.

Therefore, for consistency and simplicity across all experiments, we fix $b_I = b_H = 256$ as the default configuration, ensuring reproducible and stable performance without sacrificing accuracy.

D.5 Ablation on Calibration Dataset

We evaluated the quantization performance of LoPRo in different calibration datasets, with results presented in Table 11. The results demonstrate that LoPRo maintains consistent performance advantages across WikiText-2, c4, Pile with minimal metric fluctuations. This indicates strong generalization capability across diverse domains and text styles, and confirms that Lo-

Table 11: Ablation on calibration dataset in 2bit LLaMA2-7B quantization..

Method	Dataset	PPL	AC	AE	QA	WI
LoPRo	Wiki2	7.49	32.1	63.5	71.1	64.2
	C4	7.39	31.2	62.8	71.1	63.8
	Pile	7.43	31.3	63.3	71.4	64.2
LoPRo$_v$	Wiki2	6.55	33.8	68.8	73.5	65.2
	C4	6.53	34.6	69.0	72.7	66.5
	Pile	6.53	34.7	69.1	72.9	65.1

Table 10: Ablation on block size parameters in 2bit LLaMA2-7B quantization.

Method	b_I	b_H	PPL	AC	AE	QA	WI
LoPRo	64	64	7.51	32.9	64.1	70.2	64.5
	128	64	7.43	29.9	64.4	70.7	64.5
	128	128	7.46	31.6	64.3	70.2	64.9
	256	64	7.40	30.8	63.9	71.7	63.1
	256	128	7.38	32.1	62.8	71.2	64.0
	256	256	7.39	31.2	62.8	71.1	63.8
LoPRo$_v$	64	64	6.60	34.1	69.8	73.2	65.9
	128	64	6.56	35.5	68.1	72.5	66.3
	128	128	6.53	34.0	69.4	72.9	65.4
	256	64	6.55	33.6	69.2	72.0	65.9
	256	128	6.53	34.8	69.0	72.5	65.5
	256	256	6.53	34.6	69.0	72.7	66.5

PRo is not sensitive to the specific statistical properties of any single calibration set. This robustness aligns with the design philosophy of LoPRo as a fine-tuning-free quantization framework. Consequently, for all other experiments in this work, we adopt c4 as the default calibration dataset.

E Additional Evaluations

E.1 Comparison with QTIP

We note that QTIP (Tseng et al., 2024b) is similar to QuIP# and belongs to the rotation-based vector quantization family, achieving superior zero-shot accuracy; accordingly, we conduct additional experiments to compare LoPRo against QTIP. As shown in Table 12, in the 2-bit no-fine-tuning setting, LoPRo achieves *lower* perplexity than QTIP on LLaMA2-7B (6.53 vs. 6.82), while QTIP retains a slight edge on LLaMA2-13B (5.52 vs. 5.79). With fine-tuning, QTIP's optimized codebook recovers more effectively (5.86 vs. 6.06 on 7B; 5.11 vs. 5.34 on 13B), which we attribute to differences in codebook design: QTIP employs a sophisticated iterative codebook optimization, whereas we adopt a simpler GPTVQ-style vector quantization with randomly initialized codebooks refined only through our low-rank update procedure. This design prioritizes simplicity, speed, and compatibility with existing quantization pipelines at the cost of some representational capacity in the extreme 2-bit regime. Nevertheless, under 3-bit scalar quantization, LoPRo matches QTIP's vector-quantized performance (5.43 vs. 5.40 on 7B)—a result that is particularly valuable for hardware deployment, as scalar quantization imposes far fewer constraints on accelerator design. Consistent with this, the main-text "best 2-bit" claim is scoped to the compared *fine-tuning-free scalar and vector* baselines, where LoPRo$_v$ leads on 7B and Mixtral and is on par with GPTVQ on 13B (5.79 vs. 5.78).

Speed: LoPRo is at least 5× faster in quantization runtime compared to both QTIP and QuIP# (26 min vs 3 h for a 7B model).

Flexibility & Hardware Friendliness: LoPRo supports both scalar and non-scalar quantization schemes. Notably, under 3-bit scalar weight-only quantization, LoPRo matches the accuracy of QTIP's vector-quantized model—a result that is particularly valuable for hardware deployment, as scalar quantization imposes far fewer constraints on accelerator design.

E.2 Comparison with MoEQuant

Since the MoEQuant (Chen et al., 2025) paper neither reports results on several Zero-Shot benchmarks used in our main experiments nor provides open-source code, we introduce additional evaluation datasets — including BoolQ (BQ) (Clark et al., 2019), Hellaswag (HS) (Zellers et al., 2019), OpenbookQA (OB) (Mihaylov et al., 2018), MathQA (MQ) (Amini et al., 2019) to ensure a fair and comprehensive comparison. Results are summarized in Table 13.

Table 12: Performance of LoPRo vs. QTIP. LoPRo uses scalar quantization at 3-bit and vector quantization at 2-bit.

Model	Method	Wiki(2bit)	Wiki(3bit)	Wiki(2bit)-FT	Wiki(3bit)-FT
LLaMA2-7b	QTIP	6.82	5.4	5.86	5.28
	LoPRo	6.53	5.43	6.06	5.37
LLaMA2-13b	QTIP	5.52	4.74	5.11	4.69
	LoPRo	5.79	4.84	5.34	4.8

LoPRo consistently outperforms MoEQuant++ across most evaluation tasks at matched bitwidth. At 3-bit, LoPRo (3.19 avg bits) achieves lower perplexity than MoEQuant++ (4.15 vs. 4.90) and a +5.4 point improvement on BoolQ (88.2 vs. 82.8). At 2-bit, LoPRo reaches competitive BoolQ accuracy (87.3, +4.5 points *above* MoEQuant++ at 3-bit) while using one full bit less, validating the effectiveness of the low-rank residual rotation at aggressive compression. On other tasks (HS, MQ), scalar LoPRo at 2-bit trails MoEQuant++ at 3-bit modestly, as expected given the half-bitwidth disadvantage.

These results confirm that LoPRo at 3-bit avg clearly surpasses MoEQuant++ at 3-bit across all metrics, and at 2-bit avg retains competitive BoolQ accuracy with half the effective storage. Moreover, quantizing the full 47B-parameter model (Mixtral-8x7B) takes only approximately 2.5 h, highlighting the efficiency of our method for deploying massive MoE models in resource-constrained environments.

Table 13: Performance of LoPRo and MoeQuant in Mixtral-8x7B model. Context length is 4096; zero-shot abbreviations (AC/AE/WI/QA) follow §4.1 and (HS/OB/BQ/MQ) App. E.4.

Method	Bit	PPL	AC	AE	WI	QA	HS	OB	BQ	MQ
MoeQuant++	3	4.90	-	-	-	-	60.1	31.2	82.8	38.8
LoPRo	2.19	5.25	53.2	81.8	72.0	78.1	56.1	30.6	87.3	33.8
	3.19	4.15	61.0	85.3	77.0	83.3	65.4	35.0	88.2	43.3
LoPRo$_v$	2.19	4.80	55.8	82.9	74.0	80.5	59.4	32.2	87.3	37.5
	3.19	4.16	60.6	86.5	76.7	82.9	65.0	36.4	87.7	42.9

E.3 QWEN RESULTS

We evaluate LoPRo and its variant LoPRo$_v$ on three recent Qwen models — Qwen2.5-7B, Qwen2.5-14B, and Qwen3-8B — under both 2-bit and 3-bit weight quantization (with 16-bit activations). As shown in Table 14, LoPRo consistently preserves strong performance even at aggressive 2-bit compression. For Qwen2.5-7B, LoPRo$_v$ outperforms LoPRo by up to 4.5% in accuracy (e.g., on AC), demonstrating the benefit of variance-aware vector quantization. At 3-bit precision, both LoPRo and LoPRo$_v$ nearly recover full FP16 performance across all benchmarks, with LoPRo$_v$ matching or exceeding FP16 on AE for Qwen2.5-7B. The trend holds for larger models. In Qwen2.5-14B, LoPRo$_v$ at W2A16 reduces perplexity by 1.13 compared to LoPRo and achieves +2.8 higher accuracy on AC. Even in the more compact Qwen3-8B architecture, LoPRo$_v$ significantly narrows the gap to FP16: its W2A16 configuration attains 47.9 on AC versus 55.6 for FP16.

These results confirm that *LoPRo enables near-FP16 quality at 3-bit and usable performance at 2-bit across diverse Qwen architectures*. The consistent gains from LoPRo$_v$ further validate our design choice of incorporating GPTVQ-style vector quantization on the rotated residual. Combined with fast quantization runtime (empirically under 1 hour for all models on a single A100-40G), LoPRo offers a practical solution for deploying high-performance, compressed Qwen models in memory- and latency-constrained scenarios.

Table 14: Performance of LoPRo Qwen2 and Qwen3 model families.

Models	Methods	Q Config	Wiki	AC	AE	WI	QA
Qwen2.5-7b	FP16	W16A16	6.86	52.6	81.9	71.1	79.7
	LoPRo	W2A16	9.43	39.6	70.2	65.5	72.5
		W3A16	7.22	51.2	82.9	70	78.8
	LoPRo$_v$	W2A16	8.53	44.1	68.6	68	75.4
		W3A16	7.23	53.5	82.8	70.6	78.9
Qwen2.5-14b	FP16	W16A16	5.24	60.7	85.7	75.6	81.6
	LoPRo	W2A16	7.75	47.4	76.7	71.9	75.7
		W3A16	5.44	57.8	84.4	75	79.4
	LoPRo$_v$	W2A16	6.62	50.2	78.8	72.5	77.3
		W3A16	5.42	57.2	83.9	74.7	79.8
Qwen3-8b	FP16	W16A16	9.01	55.6	83.5	68.1	76.7
	LoPRo	W2A16	12.59	40.6	70.5	62.9	70.8
		W3A16	9.58	54.3	81.7	66.9	75.9
	LoPRo$_v$	W2A16	11.22	47.9	77.8	66.9	72.4
		W3A16	9.59	55.3	82.3	68.2	76

E.4 Open LLM Leaderboard V1

Open LLM Leaderboard V1 provides a standardized evaluation suite for assessing language models on core academic benchmarks. It includes GSM8k (Cobbe et al., 2021) for grade-school math reasoning, MMLU (Hendrycks et al., 2020) and ARC-Challenge (Boratko et al., 2018) for measuring world knowledge and logical reasoning, Winogrande (Sakaguchi et al., 2021) and HellaSwag (Zellers et al., 2019) for commonsense and language understanding, and TruthfulQA (Lin et al., 2021) for evaluating factual correctness and resistance to generating false statements. Following Meta's prompt guidelines for Llama-3.1, the leaderboard treats MMLU and ARC-Challenge as text-generation tasks and applies chain-of-thought prompting to GSM8k, offering a consistent and reproducible protocol for comparing quantized and full-precision models.

We perform a OpenLLM V1 evaluation on Qwen3-8B, results show that both LoPRo and LoPRo$_v$ achieve consistently strong performance across quantization settings. Even with aggressive 2-bit weight quantization (W2A16), the methods retain reasonable capability—particularly on tasks like Winogrande—while W3A16 configurations recover over 94% of the FP16 baseline on average, demonstrating the effectiveness and robustness of the quantization approach across diverse benchmarks.

Table 15: Performance comparison of different quantization methods applied to the Qwen3-8B model across a suite of standard language modeling benchmarks. Recovery percentage is computed relative to the FP16 baseline (100% recovery). All evaluations follow the prompt and evaluation protocols aligned with the Open LLM Leaderboard V1, including chain-of-thought prompting for MMLU$_{\text{Cot}}$ and zero-shot settings for ARC-Challenge and TruthfulQA.

Methods	Q Config	Recovery %	Average Score	MMLU 5-shot	MMLU$_{Cot}$ 0-shot	ARC-C 0-shot	GSM8K 8-shot	HellaSwag 10-shot	Winogrande 5-shot	TruthfulQA 0-shot
FP16	-	100.00%	69.6	74.9	80.2	55.6	88.8	58.1	70.1	59.7
LoPRo	W2A16	73.89%	51.4	55.7	62.5	40.6	52.3	44.3	64.4	40.2
	W3A16	94.40%	65.7	71.1	75.3	54.3	81.2	53.9	69.9	54.2
LoPRo$_v$	W2A16	79.89%	55.6	61.8	67.9	47.9	55.5	47.2	66.2	42.7
	W3A16	94.97%	66.1	71.7	75.1	55.3	80.9	54.9	69.6	54.9

F Limitations

This work primarily focuses on weight-only quantization. However, weight and activation quantization including KV-cache quantization — remains an active and critical area in model compression.

We believe that the proposed rotation framework can be naturally extended to activation quantization, where structured rotation may similarly improve quantization efficiency by aligning activation distributions with quantization grids. Furthermore, the low-rank matrix multiplication and the reordering-rotation operations in LoPRo can be fused during inference, potentially enabling near-lossless computational efficiency with minimal overhead. We leave these directions for detailed exploration to future work.

G MORE VISUALIZATION

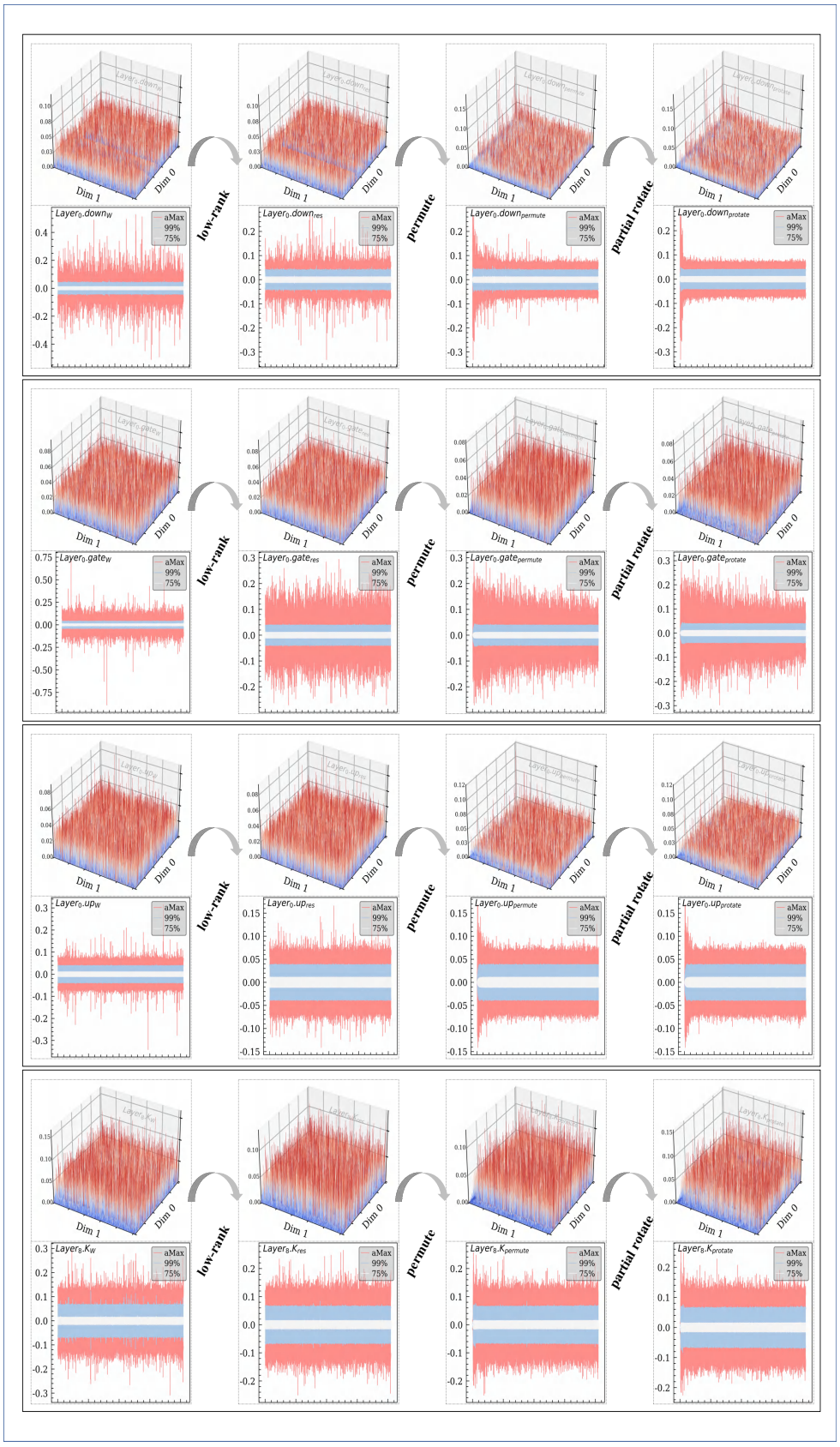

Figure 3: Visualization of layers in LLaMA2-7B

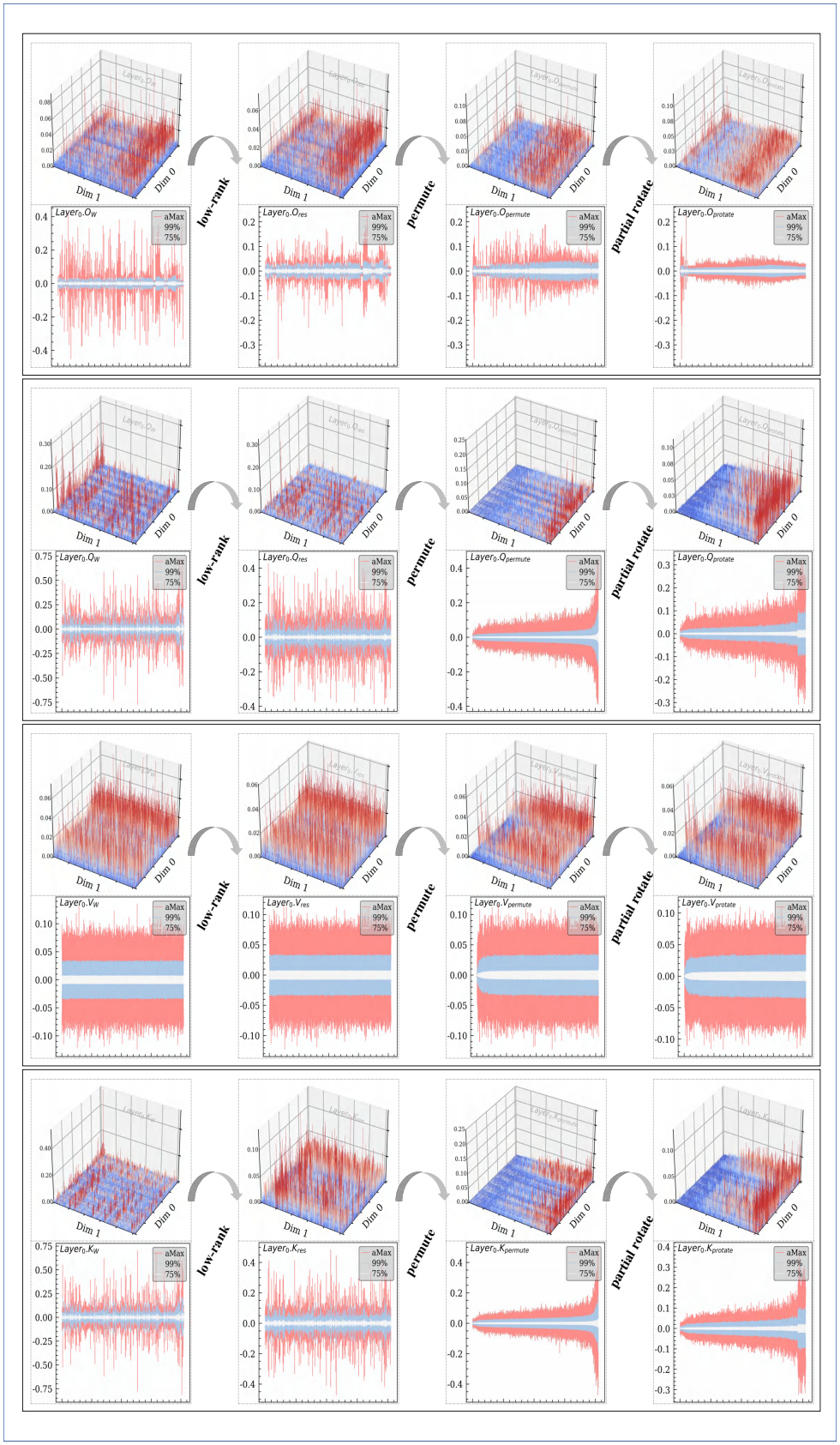

Figure 4: Visualization of layers in LLaMA2-7B

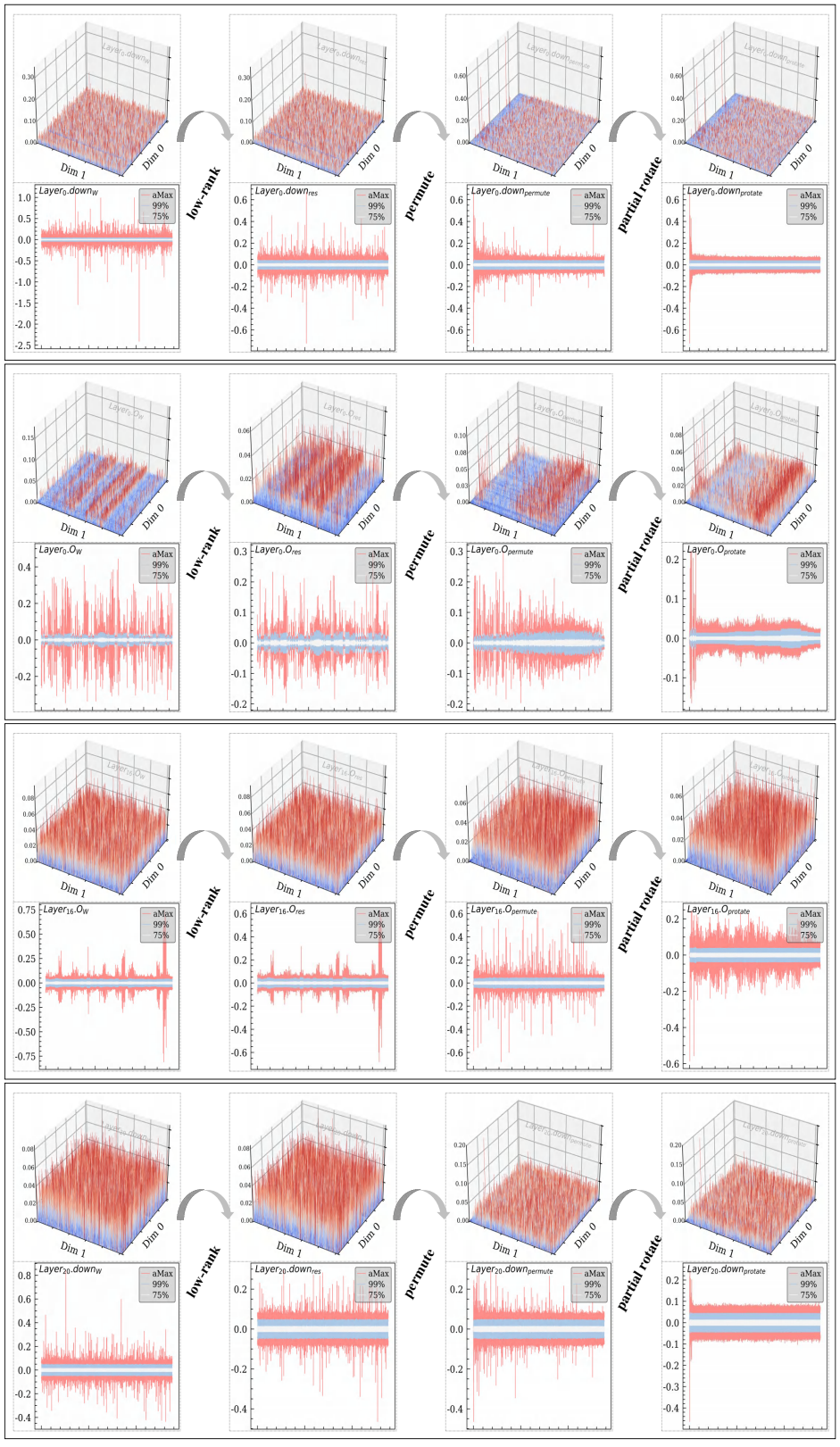

Figure 5: Visualization of layers in LLaMA2-13B

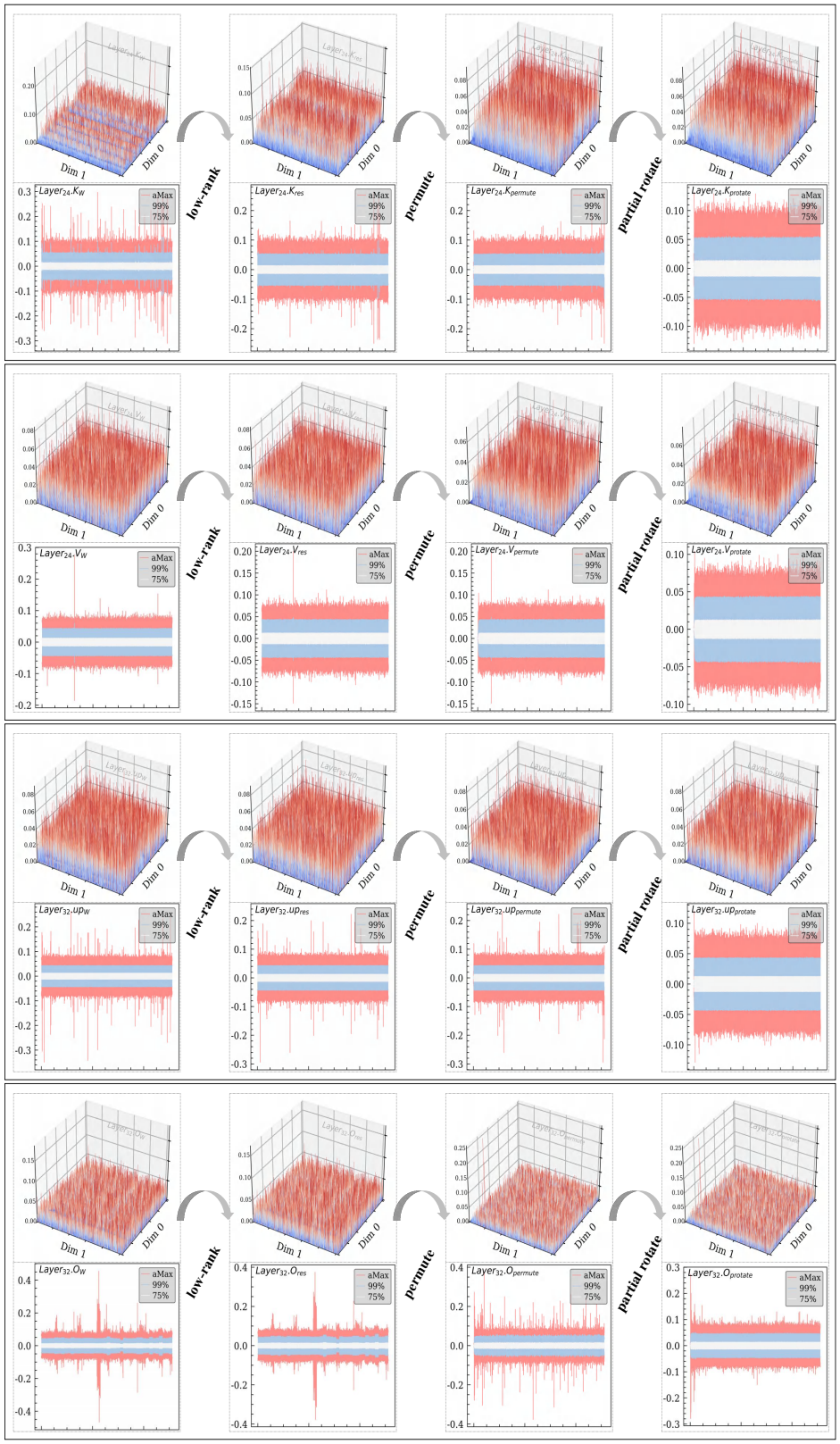

Figure 6: Visualization of layers in LLaMA2-13B

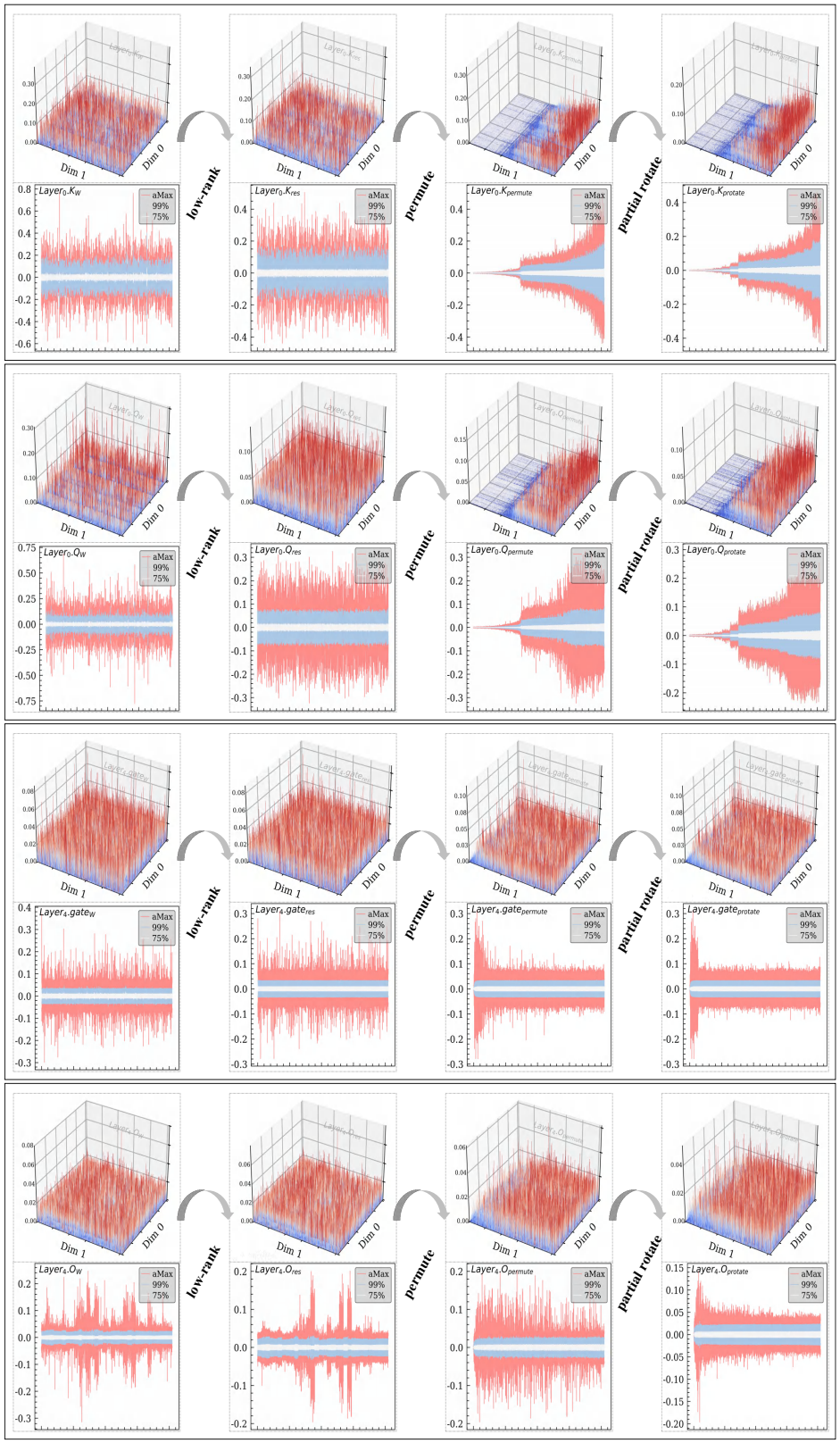

Figure 7: Visualization of layers in LLaMA3-8B

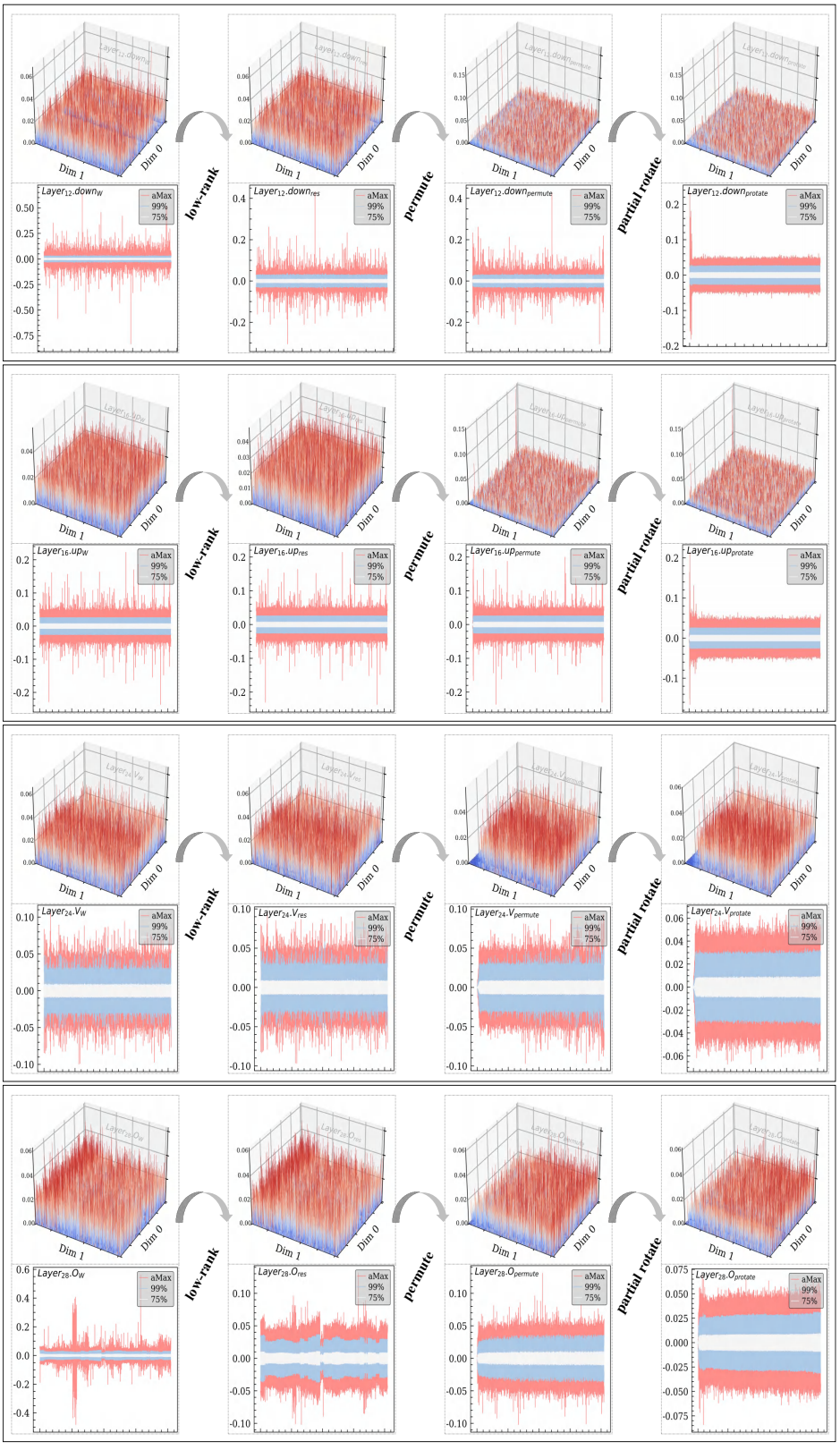

Figure 8: Visualization of layers in LLaMA3-8B

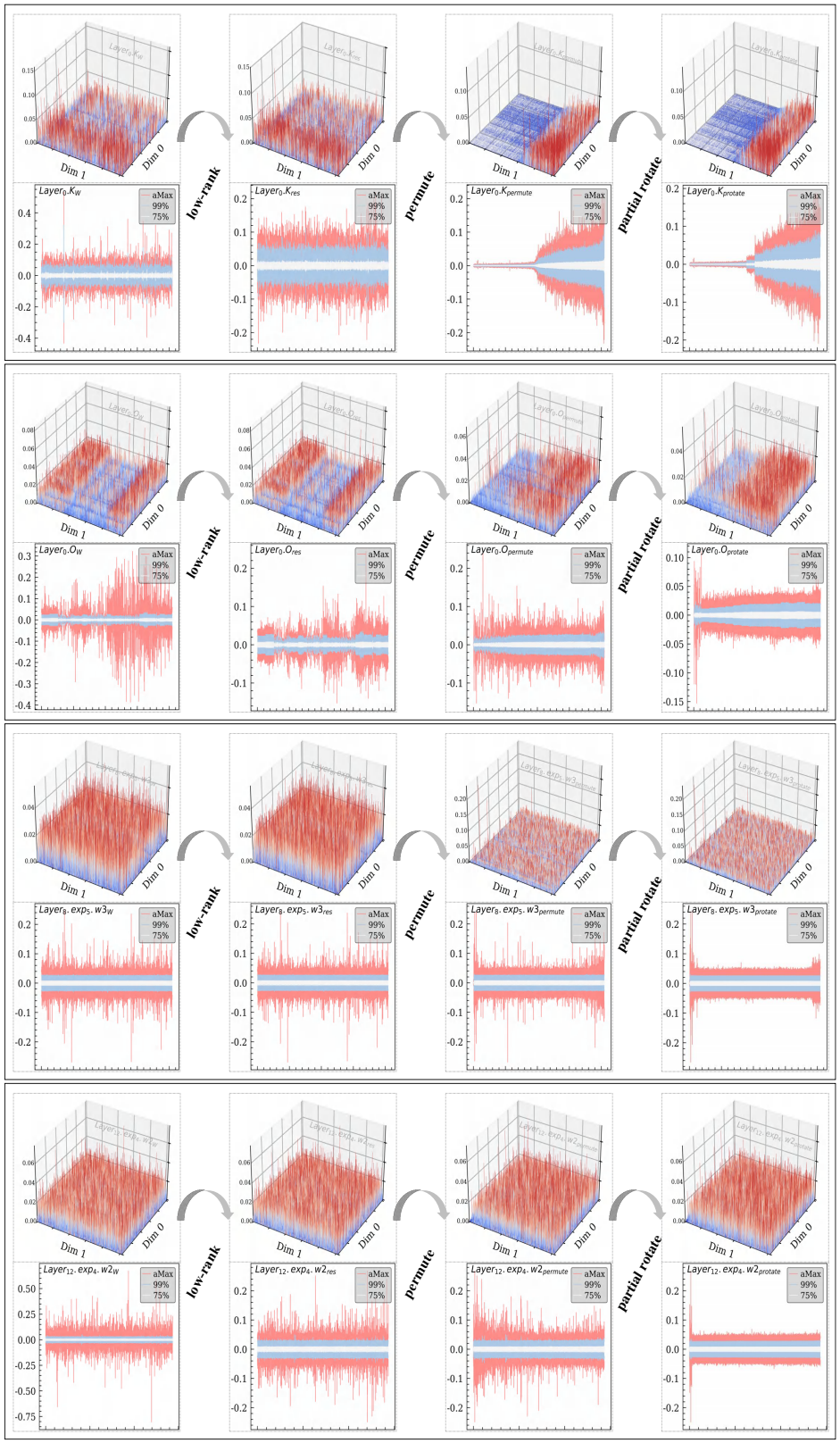

Figure 9: Visualization of layers in Mixtral-8x7B

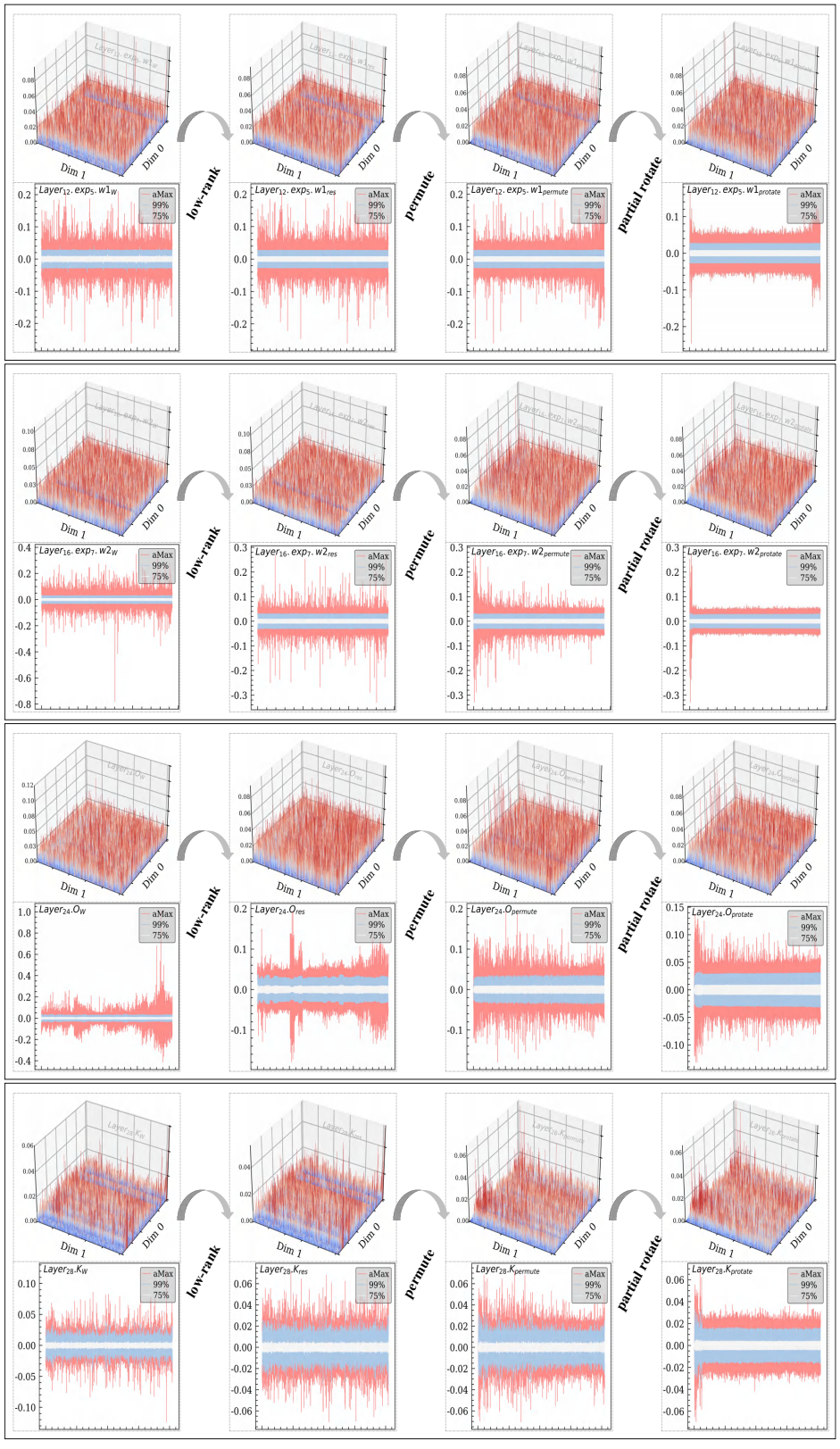

Figure 10: Visualization of layers in Mixtral-8x7B