PRQUANT: PERMUTATION RESIDUAL QUANTIZATION FOR LOW-OVERHEAD INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-bit quantization of linear layers is often dominated by a small number of outlier channels. Existing smoothing, rotation, and residual-based methods can mitigate this issue, but may shift the quantization bottleneck to weights or introduce costly online operations. To address these limitations, we propose PRQuant (Permutation Residual Quantization), a training-free framework that combines channel permutation with offline weight residual compensation. PRQuant identifies the scaled-weight columns with the largest quantization errors and permutes them into contiguous tail blocks. This structure allows the corresponding weight residuals to be precomputed entirely offline, while replacing scattered activation gathering with simple contiguous access during inference, yielding a single regular MXFP4 GEMM for compensated computation. Experiments show that PRQuant substantially reduces down-projection reconstruction error, with scaling and residual compensation providing the main numerical gains while permutation enables a hardware-friendly contiguous layout. Comprehensive experiment results on Qwen3-4B-Instruct-2507 and Qwen3-30B-A3B-Instruct-2507 illustrate that PRQuant achieves up to averagelly $2.6\times$ and $1.8\times$ operator speedup over BF16 respectively, while preserving near plain MXFP4 end-to-end decoding efficiency. Across five downstream benchmarks, PRQuant achieves the best average accuracy among the quantized methods, improving accuracy over MXFP4 by 1.24 and 0.55, respectively.

1 INTRODUCTION

Large language models (LLMs), including mixture-of-experts (MoE) architectures, have achieved strong performance in language understanding, code generation, and reasoning. However, their deployment is constrained by memory capacity, bandwidth, and the cost of large matrix multiplications (Brown et al., 2020; Fedus et al., 2022; DeepSeek-AI, 2024). Low-bit quantization is therefore a central approach to reducing inference cost and improving throughput.

Recent hardware advances have increased interest in fine-grained block-scaled FP4 formats such as MXFP4 and NVFP4 (Rouhani et al., 2023; Open Compute Project, 2023). These formats share a scale within each small block, which confines an outlier's influence to its local block. Within that block, however, a single large value can inflate the shared scale and coarsen the quantization of all other values. This unique block-wise duality implies that the properties of these formats need to be carefully considered in the design of quantization algorithms.

Existing post-training quantization (PTQ) strategies address outliers from different perspectives, but many established techniques were originally developed for integer quantization or settings without block-wise shared scaling. Rotation-based methods use orthogonal transformations to redistribute activation outliers (Ashkboos et al., 2024). Under MXFP4, global rotations may spread outlier energy across multiple scaling blocks and increase the quantization error of otherwise regular blocks (Shao et al., 2026; Sanjeet et al., 2026; Li et al., 2026). BRQ and BATQuant align their transformations with MXFP4 blocks, but their activation-side transformations still run online during inference (Shao et al., 2026; Li et al., 2026). SmoothQuant migrates quantization difficulty from activations to weights through a mathematically equivalent per-channel scaling transformation (Xiao et al., 2023). However, with weights also quantized to 4 bits, weight quantization can itself become the new bottleneck. Residual compensation can partially recover the accuracy lost under aggressive 4-bit quantization, but

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.

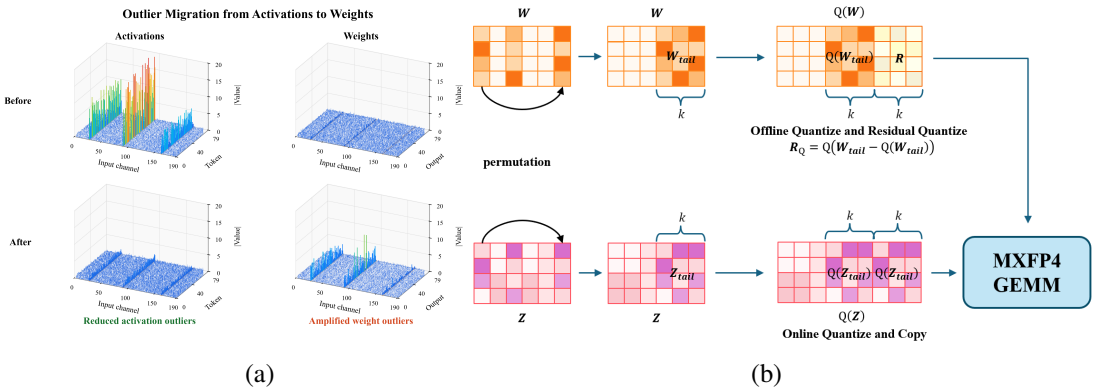


Figure 1: Schematic overview of PRQuant: (a) activation-to-weight quantization difficulty migration under scaling; (b) channel permutation, offline weight residual augmentation, and online activation augmentation.

typically introduces additional computation or memory-access overhead. For example, ARCQuant extends the activation matrix with quantized residual channels to preserve a unified low-precision GEMM path, but it depends on input-dependent residual channels (Meng et al., 2026). SVDQuant absorbs outliers using a low-rank branch, but this additional branch requires specialized kernel fusion to avoid extra data movement (Li et al., 2025).

These limitations motivate a low-overhead W4A4 method that preserves a regular low-precision GEMM path. We propose **PRQuant (Permutation Residual Quantization)**, whose overall pipeline is illustrated in Figure 1. For each candidate α, PRQuant first applies AWQ-style channel scaling and identifies the top-k input channels that are most difficult to quantize in the scaled weights. These channels are permuted to the tail, where a static weight side residual sub-tensor is constructed from their MXFP4 quantization residuals. At inference time, PRQuant does not compute input-dependent activation residuals; it only reuses the corresponding tail activation channels, which can be obtained either by direct copying or through offline expansion of W_{up} and W_{gate}. The main and residual paths can ultimately be merged into a single regular low-precision GEMM.

Our contributions are summarized as follows:

- We propose PRQuant (Permutation Residual Quantization), a training-free PTQ framework that combines AWQ-style scaling, channel permutation, and offline weight residual compensation. The permutation moves the selected difficult channels into a contiguous tail region, so that the corresponding activations can be accessed efficiently during inference without scattered gathering.
- We design an efficient implementation of PRQuant in which the selected weight residuals are precomputed offline and the main computation and residual compensation are executed together in a single augmented MXFP4 GEMM. This avoids online residual construction, irregular activation gathering, and a separate correction branch, while preserving a regular MXFP4 execution path.
- Extensive experiments on both dense and MoE Qwen3 models demonstrate that PRQuant improves downstream accuracy over MXFP4 while modifying only the FFN down-projection layers. Our Ascend 950PR implementation further provides substantial down-projection speedups over BF16 and ARCQuant while retaining end-to-end decoding efficiency close to MXFP4, demonstrating both numerical effectiveness and hardware efficiency across distinct FFN architectures.

2 Related Work

Scaling-based post-training quantization. SmoothQuant suppresses activation outliers by transferring quantization difficulty to weights through equivalent per-channel scaling (Xiao et al., 2023). AWQ protects salient weights using activation-aware scaling (Lin et al., 2024b), while OmniQuant optimizes learnable clipping and equivalent transformations (Shao et al., 2024). These methods are effective across low-bit quantization settings, but under W4A4, scaling may shift the bottleneck to equally constrained 4-bit weights.

Rotation and block-aware FP4 quantization. QuaRot and SpinQuant use randomized or learned rotations to reshape weight and activation distributions, while DartQuant further optimizes rotated activations (Ashkboos et al., 2024; Liu et al., 2025; Shao et al., 2025). However, global rotations may redistribute outliers across MXFP4 blocks and increase block-wise quantization error (Shao et al., 2026; Li et al., 2026). Format-aware methods therefore operate at the microscaling granularity: MR-GPTQ combines GPTQ with block-wise rotations and scale optimization, while BRQ and BATQuant apply MXFP4-aligned transformations (Egiazarian et al., 2026; Shao et al., 2026; Li et al., 2026). These methods still require runtime activation-side transformations or fused-kernel support.

Channel reordering and outlier isolation. RPTQ reduces inter-channel range variation by clustering and reordering activation channels (Yuan et al., 2023), while MixQuant uses permutation to redistribute activation mass across blocks (Sanjeet et al., 2026). These works show that channel layout affects both quantization behavior and execution efficiency, although reordering alone cannot fully recover the accuracy loss of 4-bit quantization.

Mixed-precision and residual compensation. LLM.int8() isolates outlier dimensions into a 16-bit path, while SpQR stores high-error weights in a sparse high-precision component (Dettmers et al., 2022; 2024). OSC similarly combines a 4-bit base GEMM with a 16-bit outlier branch (Zhang et al., 2026). ARCQuant avoids a high-precision branch by augmenting activations with quantized residual channels, but requires online residual construction (Meng et al., 2026); MosaicQuant combines dense and sparse 4-bit components with specialized execution support (Hu et al., 2026). These methods improve quantization accuracy through additional compensation capacity, but require extra runtime processing or execution support.

3 MOTIVATION

In practice, a small fraction of activation channels often contain large outliers (Figure 1(a)), which enlarge the quantization range and degrade 4-bit quantization accuracy. SmoothQuant alleviates this issue by transferring activation magnitudes to the weights through per-channel scaling. Although this transformation improves activation quantization accuracy, it may amplify a small number of scaled-weight input channels and create a new weight-side bottleneck under W4A4 quantization.

Limitations of high precision weight compensation. SVDQuant demonstrates that merely transferring quantization difficulty between activations and weights is insufficient at 4-bit precision. It applies a low-rank compensation approach to decompose the magnified weights into the sum of a 4-bit quantized weight matrix and a full-precision low-rank residual (LoRA) (Li et al., 2025). Although the low-rank component captures the dominant singular directions of the magnified weight and effectively preserves accuracy, it introduces significant overhead. Executing the full-precision residual incurs extra data movement, whose efficient deployment relies on specialized mixed-precision kernel fusion. This makes the approach less compatible with a standard and homogeneous low-precision GEMM path.

Permutation enables structured weight compensation. After scaling, these dominant error channels are typically scattered across non-contiguous regions (Figure 1(b)), directly compensating for them would require expensive index gatherings. Therefore, we apply channel permutation to reorder these channels into contiguous tail blocks. The permutation yields two benefits. Numerically, it isolates the large-magnitude channels from regular ones, preventing outliers from inflating the shared quantization scales of otherwise well-behaved MXFP4 blocks. Structurally, it transforms the scattered indices into a contiguous data layout, thereby facilitating coalesced memory access and aligning well with hardware vectorized loading during inference.

Static weight residual compensation. Constructing residual compensation on the activation side is input-dependent and therefore requires online computation. In contrast, weight residuals are static and can be precomputed entirely offline. The remaining runtime requirement is only to provide the corresponding activation channels. AWQ-style scaling further shifts the dominant quantization difficulty to a small subset of weight channels, making weight-side residual compensation a natural way to target the remaining error. Meanwhile, permutation consolidates the corresponding channels

into contiguous tail blocks, allowing their activations to be accessed without irregular gathering. Together, scaling and permutation make weight-side residual compensation both numerically targeted and lightweight at runtime, while retaining a regular MXFP4 GEMM path.

4 METHOD

This section presents the PRQuant pipeline for FFN down-projection layers (dense or MoE expert), as illustrated in Figure 1.

Offline scaling and channel reordering. Let X denote the input to a down-projection layer. We use Z to denote the intermediate activations of the gated FFN collected from a calibration set. Given a scaling strength α, we construct the diagonal scaling matrix as $D_\alpha = \mathrm{diag}(s_\alpha)$, where $s_{\alpha,j} = (\max_t |Z_{t,j}|)^\alpha$. We then identify the top-k hardest-to-quantize channels in $W_{\mathrm{down}} D_\alpha$ and move them to contiguous tail blocks using a permutation matrix P_α.

The scaling and permutation can be fused offline into the weights of adjacent linear layers. Specifically,

$$\begin{aligned} \mathrm{MLP}(X) &= \left(XW_{\mathrm{up}}^\top \odot \sigma(XW_{\mathrm{gate}}^\top)\right) W_{\mathrm{down}}^\top \\ &= \left(X(P_\alpha^\top D_\alpha^{-1} W_{\mathrm{up}})^\top \odot \sigma\big(X(P_\alpha^\top W_{\mathrm{gate}})^\top\big)\right) (W_{\mathrm{down}} D_\alpha P_\alpha)^\top \\ &= \widetilde{Z}_\alpha \widetilde{W}_{\mathrm{down},\alpha}^\top. \end{aligned} \tag{1}$$

Here, the last k columns of $\widetilde{Z}_\alpha$ are denoted by $\widetilde{Z}_{\alpha,\mathrm{tail}}$, and the corresponding columns of $\widetilde{W}_{\mathrm{down},\alpha}$ are denoted by $\widetilde{W}_{\mathrm{down},\alpha,\mathrm{tail}}$.

Offline residual augmentation. For the transformed down-projection weight in Eq. (1), we compute the quantization residual of its tail channels as

$$R_{W,\alpha} = \widetilde{W}_{\mathrm{down},\alpha,\mathrm{tail}} - \mathcal{Q}\Big(\widetilde{W}_{\mathrm{down},\alpha,\mathrm{tail}}\Big), \tag{2}$$

where $\mathcal{Q}(\cdot)$ denotes the MXFP4 quantize-dequantize operator.

For notational simplicity, we omit the subscript α below. The residual is appended offline to the transformed down-projection weight, yielding

$$\widehat{Y} = \mathcal{Q}\Big([\widetilde{Z}, \widetilde{Z}_{\mathrm{tail}}]\Big)\, \mathcal{Q}\Big([\widetilde{W}_{\mathrm{down}}, R_W]\Big)^\top. \tag{3}$$

Thus, the main computation and residual compensation are expressed as a single augmented MXFP4 GEMM whose reduction dimension increases from C to $C + k$, where C denotes the input dimension of the down-projection layer. For different candidate values of α, we select the one that minimizes the normalized reconstruction error between $\widehat{Y}$ and $ZW_{\mathrm{down}}^\top$.

Activation tail construction. For the activation side of Eq. (3), we consider two implementations: online copy and offline copy.

Online copy. Since the permutation is fused into the adjacent projection weights, the selected channels already form a contiguous tail in $\widetilde{Z}$, i.e., $\widetilde{Z}_{\mathrm{tail}} = \widetilde{Z}[:, C-k:C]$. This contiguous layout avoids scattered gathering. The tail slice is then appended to the original activation, and $[\widetilde{Z}, \widetilde{Z}_{\mathrm{tail}}]$ is quantized as the input to the augmented GEMM.

Offline copy. Alternatively, activation duplication can be encoded in the producer weights during offline model preparation. Following Eq. (1), the inverse channel scaling is absorbed into the up-projection weights, while the channel permutation is applied to both the up- and gate-projection weights. We then duplicate the last k output channels of both transformed projections, so that the gated FFN directly produces $[\widetilde{Z}, \widetilde{Z}_{\mathrm{tail}}]$. This removes the runtime tail-copy operation at the cost of expanding the producer output dimension from C to $C + k$.

Algorithm 1 Calibration and Export Pipeline of PRQuant

Require: Down-projection weight W_{down}, calibration activations Z_{cal}, MXFP4 quantizer $\mathcal{Q}$, candidate set $\mathcal{A}$, tail width k

Compute the per-channel maxima of Z_{cal} for scaling.
for each $\alpha \in \mathcal{A}$ **do**
  Construct D_α and the scaled weight $W_{\mathrm{down}} D_\alpha$.
  Select the top-k channels with the largest column-wise MXFP4 quantization errors.
  Construct P_α to move the selected channels to a contiguous tail block.
  Obtain $\widetilde{Z}_\alpha$ and $\widetilde{W}_{\mathrm{down},\alpha}$ using Equation (1).
  Compute $R_{W,\alpha} = \widetilde{W}_{\mathrm{down},\alpha,\mathrm{tail}} - \mathcal{Q}(\widetilde{W}_{\mathrm{down},\alpha,\mathrm{tail}})$.
  Construct $[\widetilde{Z}_\alpha, \widetilde{Z}_{\alpha,\mathrm{tail}}]$ and $[\widetilde{W}_{\mathrm{down},\alpha}, R_{W,\alpha}]$, and evaluate Equation (3).
end for
Select $\alpha^\star$ with the minimum reconstruction error and fuse the corresponding $(P_{\alpha^*}, D_{\alpha^*})$ into the up-, gate-, and down-projection weights.
Export the augmented down-projection weight $[\widetilde{W}_{\mathrm{down}}, R_W]$ in MXFP4.
For online copy, quantize $\widetilde{Z}$ once and reuse its quantized contiguous tail; alternatively, duplicate the corresponding producer channels offline.
Execute one MXFP4 GEMM with the augmented activation and weight.

Kernel design for efficient inference. We implement PRQuant on Ascend 950PR, following the online-copy formulation above. The quantized augmented weight matrix $W^q_{\mathrm{aug}} = \mathcal{Q}\Big([\widetilde{W}_{\mathrm{down}}, R_W]\Big)$. are prepared offline. At runtime, the kernel quantizes the reordered activation $\widetilde{Z}$ to MXFP4 once, then appends a copy of its quantized tail together with the corresponding block scales. This reuses the quantized representation of $\widetilde{Z}_{\mathrm{tail}}$ without repeating quantization. Provided that the tail and concatenation boundaries are aligned with the MXFP4 quantization blocks and the copied blocks retain their original scales, this procedure is equivalent to quantizing $[\widetilde{Z}, \widetilde{Z}_{\mathrm{tail}}]$ in Eq. (3). A single MXFP4 GEMM then combines the augmented activation with W^q_{aug}, extending the reduction dimension from C to $C + k$ without online activation-residual construction or a separate correction GEMM.

5 Experiments

5.1 Experimental Setup

Model and quantization configuration. We use Qwen3-4B-Instruct-2507 and Qwen3-30B-A3B-Instruct-2507 (Yang et al., 2025) as our primary evaluation models, denoted as Qwen3-4B and Qwen3-30B hereafter. Unless otherwise specified, PRQuant is applied only to the FFN down-projection layers: the dense FFN down-projections for Qwen3-4B and the expert down-projections for Qwen3-30B. All compared methods use the same model checkpoint and evaluation pipeline. We evaluate a mixed MXFP quantization configuration, where the attention modules are quantized in MXFP8 and the FFN/MoE modules are quantized in MXFP4.

Baseline methods. We benchmark PRQuant against full-precision BF16, default MXFP4, and five representative low-bit quantization methods. Specifically, BF16 provides the full-precision reference, whereas default MXFP4 directly quantizes both activations and weights without additional quantization-aware transformations or compensation. The low-bit baselines are: ARCQuant, which compensates for activation quantization error; BATQuant, which learns block-wise affine transformations and clipping (Li et al., 2026); FlatQuant, which learns affine transformations to reshape weight and activation distributions (Sun et al., 2025); MR-GPTQ, which combines Hessian-based weight quantization with MXFP-aligned rotations and scale optimization (Egiazarian et al., 2026); and QuaRot, for which we apply random Hadamard rotations to the down-projection layers (Ashkboos et al., 2024). All quantized methods follow the same mixed-precision setting, with MXFP8 for the attention modules and MXFP4 for the FFN/MoE modules.

Table 1: End-to-end accuracy (↑) on Qwen3-30B and Qwen3-4B using zero-shot evaluation. For quantized configurations, the attention modules use MXFP8 and the FFN/MoE modules use MXFP4. Bold values denote the best result among the quantized configurations for each model. The average is the unweighted mean over the five benchmarks.

Method	ARC-C	ARC-E	HellaSwag	PIQA	WinoGrande	Average
Qwen3-30B						
BF16	95.93	98.77	90.19	92.27	75.92	90.62
MXFP4	94.57	98.58	88.74	90.81	74.03	89.35
ARCQuant	**95.25**	**98.77**	87.23	91.57	73.01	89.17
BATQuant	94.24	98.59	88.60	91.62	74.59	89.53
FlatQuant	94.92	**98.77**	88.73	91.51	75.14	89.81
MR-GPTQ	94.23	97.88	**89.21**	91.46	75.30	89.62
QuaRot	94.92	97.88	87.70	91.51	72.38	88.88
PRQuant	94.58	**98.77**	88.77	**91.95**	**75.45**	**89.90**
Qwen3-4B						
BF16	93.22	97.71	80.56	85.26	63.61	84.07
MXFP4	90.85	96.65	73.83	83.08	**60.38**	80.96
ARCQuant	92.20	96.65	75.28	83.95	58.56	81.33
BATQuant	92.20	96.47	77.30	**84.44**	59.51	81.98
FlatQuant	90.17	95.23	**77.96**	83.79	59.98	81.43
MR-GPTQ	89.83	**97.18**	76.41	83.30	59.27	81.20
QuaRot	90.85	97.00	75.32	83.35	59.59	81.22
PRQuant	**92.54**	**97.18**	77.51	83.84	59.91	**82.20**

Calibration and benchmarking datasets. All quantization methods use the same calibration set, `mix_calib.jsonl` [1]. We evaluate end-to-end accuracy across five standard benchmarks: ARC-Challenge (ARC-C), ARC-Easy (ARC-E), HellaSwag, PIQA, and WinoGrande (Clark et al., 2018; Zellers et al., 2019; Bisk et al., 2020; Sakaguchi et al., 2021).

Implementation and environmental setup. For the end-to-end accuracy comparison, all methods are implemented and evaluated under identical environmental conditions on the Ascend 950PR. Specifically, all model checkpoints (including full-precision and quantized results) are deployed using the `vllm-ascend (0.20.2 RC1)` serving framework, and end-to-end evaluation is conducted uniformly via OpenCompass (OpenCompass Contributors, 2023).

5.2 End-to-End Accuracy

We report the end-to-end accuracy in Table 1. For Qwen3-30B, default MXFP4 reduces the average accuracy from 90.62 to 89.35, corresponding to a 1.27 degradation from BF16. By modifying only the expert down-projection computation, PRQuant improves the average accuracy to 89.90, yielding a 0.55 gain over MXFP4 and recovering approximately 43.3% of the accuracy gap between MXFP4 and BF16.

The improvement over MXFP4 is observed across all five benchmarks. In particular, PRQuant improves WinoGrande and PIQA by 1.42 and 1.14, respectively, while providing smaller gains of 0.19, 0.03, and 0.01 on ARC-E, HellaSwag, and ARC-C. Compared with the BF16 reference, PRQuant fully preserves the ARC-E accuracy and limits the PIQA degradation to only 0.32. These empirical results demonstrate that mitigating the reconstruction error in sensitive expert down-projections improves end-to-end accuracy.

Although their average accuracies are slightly lower than that of PRQuant, BATQuant, FlatQuant, and MR-GPTQ achieve comparable results on this model. We therefore compare their quantization

[1] `http://gitcode.com/Ascend/msmodelslim/blob/master/lab_calib/mix_calib.jsonl`

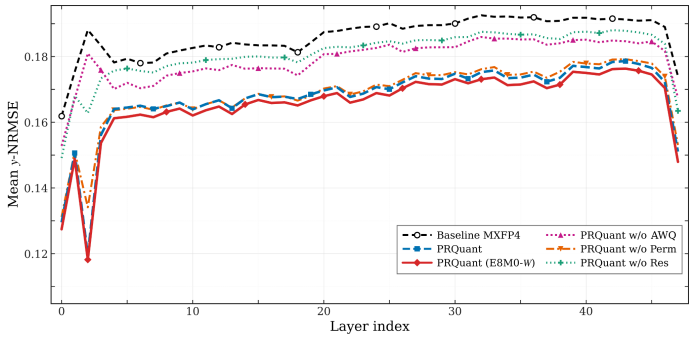


Figure 2: Ablation results on the down-projection layers of routed experts of Qwen3-30B. Layer-wise mean y-NRMSE across experts.

cost in Table 2. The training-based BATQuant and FlatQuant and the Hessian-based MR-GPTQ require 7.4–9.1 hours, whereas the training-free PRQuant completes quantization in 0.4 hours and uses less memory. These results make PRQuant the most economical choice among the compared methods for quantizing Qwen3-30B, combining comparable or better accuracy with the lowest time and memory costs.

For the dense Qwen3-4B, the quantization gap is substantially larger: default MXFP4 reduces the average accuracy from 84.07 to 80.96, corresponding to a 3.11 degradation from BF16. Applying PRQuant to the dense FFN down-projections improves the average accuracy to 82.20, yielding a 1.24 improvement over MXFP4 and recovering approximately 39.9% of the average accuracy gap between MXFP4 and BF16. PRQuant again achieves the highest average accuracy among the evaluated quantized methods.

Compared with MXFP4, PRQuant improves four of the five benchmarks, with gains of 1.69 on ARC-C, 0.53 on ARC-E, 3.68 on HellaSwag, and 0.76 on PIQA. The largest improvement is observed on HellaSwag, suggesting that the dense model is particularly sensitive to quantization errors in the FFN down-projections. Although individual baselines remain competitive on specific tasks, PRQuant provides the strongest overall accuracy across the five benchmarks. These results show that reducing reconstruction error in the targeted down-projections translates into end-to-end accuracy gains for both MoE expert FFNs and dense FFNs, while requiring no modification to the attention computation.

5.3 Ablation Studies

We conduct layer-wise ablations on the expert down-projection layers of Qwen3-30B. We report the mean normalized reconstruction error (denoted as y-NRMSE) across experts in each layer. We compare the complete PRQuant with naive MXFP4 quantization and three ablated variants that remove AWQ scaling (w/o AWQ), residual compensation (w/o Res), and channel permutation (w/o Perm), respectively. We additionally evaluate PRQuant with weight-side E8M0 block-scale search, denoted as PRQuant (E8M0-W). Figure 2 presents the averaged y-NRMSE across all evaluated layers and the layer-wise results.

PRQuant reduces the average y-NRMSE from 0.1861(naive MXFP4) to 0.1678. Removing AWQ scaling or residual compensation causes the largest degradation, showing that they provide the main numerical gains. Removing permutation results in only a modest error increase, while its primary benefit is structural: it reorganizes scattered hard channels into a contiguous tail block, enabling residual compensation through a regular augmented GEMM without dynamic gathering. Finally, introducing a local block-scale search PRQuant (E8M0-W) further suppresses the error to 0.1658, demonstrating that optimizing weight scaling factors can achieve even higher reconstruction accuracy under MXFP4 quantization.

5.4 Efficiency and Speed

We evaluate the runtime efficiency of PRQuant on Ascend 950PR at both the operator and end-to-end levels. We compare BF16, plain MXFP4, ARCQuant, and PRQuant under the same inference

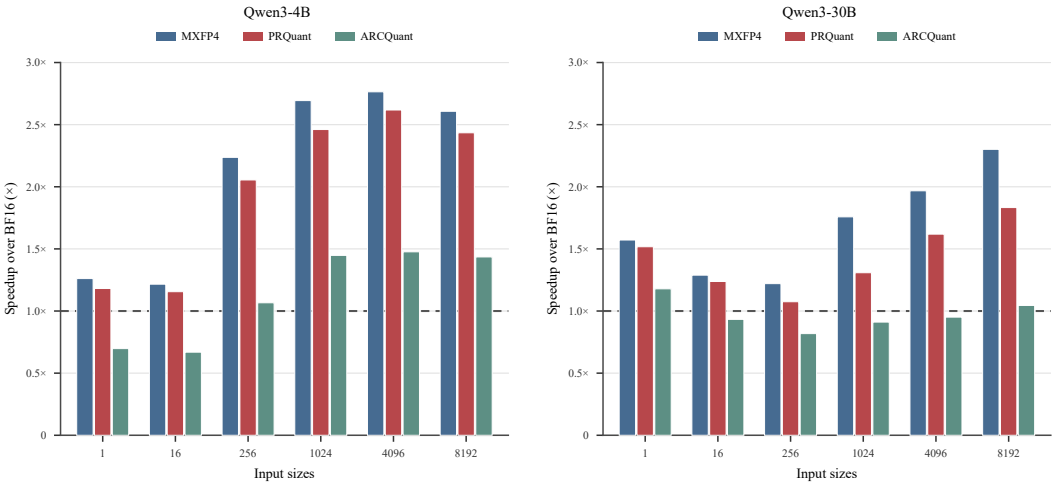


Figure 3: Operator-level Speedup (↑) over BF16 on Ascend 950PR for the down-projection kernels of Qwen3-4B and Qwen3-30B across representative input sizes M. The dashed horizontal line denotes the BF16 baseline ($1\times$).

Table 2: Quantization cost (↓) of Qwen3-30B. Both BATQuant and FlatQuant are implemented using the `ModelSlim`[2] framework.

Method	Quantization Latency (h)	Memory (GiB)
BATQuant	8.5	36.1
FlatQuant	7.4	22.9
MR-GPTQ	9.1	59.2
PRQuant	0.4	4.4

configuration. For end-to-end measurements, we use identical output lengths, KV-cache precision, and KV-cache capacity across all methods.

Operator efficiency. Figure 3 compares the speedups of the down-projection kernels relative to BF16 across input sizes M. On Qwen3-4B and Qwen3-30B, PRQuant achieves up to approximately $2.6\times$ and $1.8\times$ speedup over BF16, respectively, while being about $1.8\times$ and $1.6\times$ faster than ARCQuant across the representative workloads. The additional advantage over ARCQuant mainly comes from the lightweight online activation handling. Compared with plain MXFP4, PRQuant introduces additional computation from the expanded GEMM reduction dimension and the tail copy operation, and therefore naturally incurs some latency overhead. Nevertheless, PRQuant incurs only about 5-10% additional latency on Qwen3-4B, while the gap is larger on Qwen3-30B. The larger gap on Qwen3-30B is consistent with its larger relative reduction dimension expansion: $k/C = 16.7\%$ ($C = 768$, $k = 128$), compared with 10.5% ($C = 9728$, $k = 1024$) on Qwen3-4B. Moreover, the smaller expert projections in Qwen3-30B take less time to compute, so tail copying and associated activation-processing overhead account for a larger fraction of the total kernel latency.

End-to-end efficiency. Tables 3 and 4 report the end-to-end TTFT and TPOT results. PRQuant consistently reduces both TTFT and TPOT relative to BF16 across all tested configurations. During decoding, its latency remains within approximately 6% of plain MXFP4 on Qwen3-4B and 4% on Qwen3-30B. Compared with ARCQuant, PRQuant reduces TTFT by approximately 4.5-38.8% and TPOT by 1.8-6.8% on Qwen3-4B. On Qwen3-30B, PRQuant achieves up to 14.9% lower TTFT than ARCQuant while preserving decode efficiency, with the TPOT difference remaining within 1.6%. Together with the accuracy results, these results show that PRQuant preserves most of the decoding efficiency of plain MXFP4 while providing improved quantization accuracy.

[2] `https://gitcode.com/Ascend/msmodelslim`

Table 3: Prefill latency TTFT (ms, ↓) of Qwen3-4B and Qwen3-30B on Ascend 950PR under different batch sizes and input sequence lengths (ISL).

Batch	ISL	Qwen3-4B				Qwen3-30B			
		BF16	MXFP4	PRQuant	ARCQuant	BF16	MXFP4	PRQuant	ARCQuant
1	1024	41.2	30.8	31.8	52.0	65.5	52.8	55.7	62.9
	2048	72.2	47.0	46.1	54.1	92.5	65.8	86.8	85.1
	4096	147.8	87.4	88.0	93.2	186.2	129.8	161.7	166.5
8	1024	224.9	132.4	132.8	171.2	250.4	174.6	234.6	248.7
	2048	346.4	189.6	192.5	222.5	458.8	296.5	411.3	447.9
	4096	640.6	390.7	395.1	424.1	917.8	610.2	749.5	847.3
16	1024	346.2	198.9	202.0	237.3	430.0	277.6	383.2	401.7
	2048	628.2	352.0	352.6	398.2	826.4	529.8	716.8	757.8
	4096	1228.0	776.8	782.8	833.2	1746.8	1168.8	1334.1	1561.5
32	1024	595.4	338.8	344.0	385.5	767.5	489.6	658.7	696.3
	2048	1179.1	667.3	684.7	757.3	1560.6	1004.3	1265.8	1389.6
	4096	2427.7	1599.9	1610.9	1686.9	3411.0	2314.5	2545.4	2992.5

Table 4: Decode latency TPOT (ms/token, ↓) and speedup (↑) over BF16 of Qwen3-4B and Qwen3-30B on Ascend 950PR with ISL $= 2048$ and OSL $= 128$.

Model	Batch	TPOT (ms/token)				Speedup		
		BF16	MXFP4	PRQuant	ARCQuant	MXFP4	PRQuant	ARCQuant
Qwen3-4B	1	7.4	4.9	5.2	5.3	1.51×	1.42×	1.40×
	8	11.0	7.7	7.7	8.2	1.43×	1.43×	1.34×
	16	15.3	11.0	11.0	11.8	1.39×	1.39×	1.30×
	32	23.9	17.9	17.9	19.0	1.34×	1.34×	1.26×
Qwen3-30B	1	8.2	6.6	6.8	6.9	1.24×	1.21×	1.19×
	8	24.7	13.4	13.7	13.7	1.84×	1.80×	1.80×
	16	25.1	15.2	15.6	15.4	1.65×	1.61×	1.63×
	32	35.5	22.9	23.6	23.9	1.55×	1.50×	1.49×

6 Conclusion

In this work, we presented PRQuant, a low-overhead framework for MXFP4 quantization. Our approach addresses the weight quantization errors amplified by activation scaling, which concentrate in a small set of input channels of the weight matrix. PRQuant combines AWQ-style scaling with a permutation that groups these channels into contiguous tail blocks and can be fused into the preceding projections offline. We precompute and quantize the corresponding weight residuals offline. At inference time, we copy the quantized tail activations together with their block scales, or obtain the augmented activations through offline producer expansion. This allows the main computation and residual compensation to share a single MXFP4 GEMM without online residual construction, irregular gathers, or specialized mixed-precision kernels. Experiments on both MoE and dense models show that applying PRQuant only to FFN down-projections improves average downstream accuracy over default MXFP4. Meanwhile, PRQuant substantially accelerates the down-projection over BF16 and ARCQuant, while retaining end-to-end decoding efficiency close to that of plain MXFP4. These results demonstrate the benefits of structuring residual compensation around standard low-precision computation. Future work will explore extending PRQuant beyond FFN down-projections to other Transformer layers, such as attention output and FFN up/gate projections, through specific offline fusion strategies.

Ethics Statement

This work focuses on model compression and inference efficiency and does not involve new data collection or additional risks to user privacy. Low-bit quantization may alter model output behavior; therefore, task-level and safety regression tests should be conducted before deployment.

AI Use Statement

We used AI solely to assist with language editing, including grammar correction and improvements to the clarity and readability of the manuscript. All AI-assisted edits were reviewed and revised by the authors to ensure that they accurately reflected the intended meaning. The authors take full responsibility for the final content of the paper.

References

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. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100213–100240, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Tom B. Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332, 2022.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, 2024.

Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. In *International Conference on Learning Representations*, 2026.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Yangjia Hu, Haodong Wang, Zicong Hong, Qianli Liu, Quanxin Shou, Jian Lin, Song Guo, Xiaowei Shen, Xiangjun Huang, Dian Wang, and Jian Yang. MosaicQuant: Inlier–outlier disaggregation for unified 4-bit LLM quantization. *arXiv preprint arXiv:2606.15652*, 2026.

Ji-Fu Li, Manyi Zhang, Xiaobo Xia, Han Bao, Haoli Bai, Zhenhua Dong, and Xianzhi Yu. BATQuant: Outlier-resilient MXFP4 quantization via learnable block-wise optimization. *arXiv preprint arXiv:2603.16590*, 2026.

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. In *International Conference on Learning Representations*, 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 *Advances in Neural Information Processing Systems*, volume 37, pp. 87766–87800, 2024a.

Haokun Lin, Xinle Jia, Haobo Xu, Bingchen Yao, Xianglong Guo, Yichen Wu, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. DuQuant++: Fine-grained rotation enhances microscaling FP4 quantization. *arXiv preprint arXiv:2604.17789*, 2026.

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. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 87–100, 2024b.

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 *International Conference on Learning Representations*, 2025.

Haoqian Meng, Yilun Luo, Yafei Zhao, Wenyuan Liu, Peng Zhang, and Xindian Ma. ARCQuant: Boosting NVFP4 quantization with augmented residual channels for LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8609–8623. Association for Computational Linguistics, 2026.

Open Compute Project. OCP microscaling formats (MX) specification, version 1.0. Technical report, Open Compute Project, September 2023. Final specification.

OpenCompass Contributors. OpenCompass: A universal evaluation platform for foundation models. GitHub repository, 2023. `https://github.com/open-compass/opencompass`.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

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.

Sai Sanjeet, Ian Colbert, Pablo Monteagudo-Lago, Giuseppe Franco, Yaman Umuroglu, and Nicholas J. Fraser. MixQuant: Pushing the limits of block rotations in post-training quantization. *arXiv preprint arXiv:2601.22347*, 2026.

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. In *International Conference on Learning Representations*, 2024.

Yuantian Shao, Yuanteng Chen, Peisong Wang, Jianlin Yu, Jing Lin, Yiwu Yao, Zhihui Wei, and Jian Cheng. DartQuant: Efficient rotational distribution calibration for LLM quantization. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yuantian Shao, Peisong Wang, Yuanteng Chen, Chang Xu, Zhihui Wei, and Jian Cheng. Block rotation is all you need for MXFP4 quantization. In *International Conference on Learning Representations*, 2026.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, Xin Jiang, Wulong Liu, and Jun Yao. Flatquant: Flatness matters for llm quantization. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 57587–57613, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 2023.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. RPTQ: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Zhiyuan Zhang, Yanzhao Li, Zhiqiang Zou, Bai Du, Yupeng Sun, Hui Dong, and Hui Wang. OSC: Hardware efficient W4A4 quantization via outlier separation in channel dimension. *arXiv preprint arXiv:2604.12782*, 2026.

A Applicability and future work

Although this work uses permutation as the primary example, the underlying principle is more general. Any channel-side operation acting on a hidden dimension shared by adjacent modules can support a low-overhead compensation mechanism, provided that it can be absorbed offline into the producer or consumer weights, or that it exposes the required activation sub-tensor through regular access. Because different layers provide different fusion opportunities, the applicable operations need not be restricted to permutation. They may also include diagonal scaling, block permutation, channel duplication, structured augmentation, or structured pruning with residual correction.

This principle applies most naturally to the second projection of an FFN, including the down-projection of a dense FFN and the down-projection of an MoE expert. For these layers, channel reordering, scaling, and tail duplication can be implemented by modifying adjacent weights offline. Similar ideas may also be extended to the attention output projection.

We select the MoE expert down-projection as the primary experimental setting for two reasons. First, this layer follows the gated activation, and prior work has observed token-local massive outliers at FFN down-projection inputs. SmoothQuant may also migrate activation-side difficulty at this position into pronounced weight-side outliers (Lin et al., 2024a). Second, existing MXFP4 methods typically rely on block-aware rotations, affine transformations, fallback paths, or residual/low-rank compensation to recover accuracy. These mechanisms may still introduce online activation transformations, dynamic residual construction, additional high-precision branches, or specialized kernel fusion (Shao et al., 2026; Lin et al., 2026; Li et al., 2026; Zhang et al., 2026; Meng et al., 2026; Li et al., 2025). The MoE expert down-projection therefore combines substantial quantization difficulty with a suitable channel structure, making it a representative layer for evaluating the proposed method.