GAPSQ: CLOSING THE PRUNING-INDUCED GAP FOR JOINT SPARSIFICATION AND QUANTIZATION OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training sparsification and quantization can be combined to compress large language models, but pruning changes the weight distribution seen by the quantizer in a systematic way. By removing many near-zero weights, it creates a pruning-induced gap between the retained negative and positive branches, leading to under-utilization of the quantization range and available levels. Motivated by this observation, we introduce GapSQ, a training-free method for aggressive joint compression. We primarily study 50% sparsity with 3- or 4-bit quantization, where GapSQ narrows the pruning-induced gap through group-wise inward shifting before quantization. To avoid storing per-weight branch information, GapSQ enforces quantized-space separability, allowing the shift to be reversed using only lightweight group-level metadata. Across Llama and Qwen models, multiple sparsity patterns, and several pruning–quantization backbones, GapSQ consistently improves joint-compression baselines, with larger gains observed in more aggressive low-bit regimes. For example, on Llama 3-8B with 3-bit quantization and 50% sparsity, GapSQ improves average zero-shot accuracy from 57.2% to 62.5% while reducing WikiText-2 perplexity from 30.52 to 11.72 relative to SparseGPT+GPTQ. These results show that explicitly adapting post-pruning weight distributions is an effective way to improve low-bit joint compression.

1 INTRODUCTION

Recent advances in large language models (LLMs) have led to substantial progress in several areas, such as commonsense and mathematical reasoning (Wei et al., 2022; Lewkowycz et al., 2022; Guo et al., 2025), code synthesis (Chen et al., 2021), and complex instruction following (Ouyang et al., 2022; Zhou et al., 2023; Xu et al., 2024). Still, their practical deployment remains severely bottlenecked by memory footprint, storage cost, and inference bandwidth. Post-training compression has therefore become a prevailing tool for practical LLM serving. Among existing approaches, weight sparsification and weight quantization are particularly attractive because they can be applied after pretraining and directly reduce the cost of storing and moving model weights. Sparsification (Frantar & Alistarh, 2023; Sun et al., 2024) removes a subset of weights to induce sparsity, while quantization (Dettmers et al., 2022; Frantar et al., 2023) reduces the numerical precision used to represent model weights. Combining the two offers a natural route toward more aggressive compression than either technique alone.

Yet joint sparsification and quantization is not simply a matter of pruning first and quantizing later. Pruning changes the weight distribution seen by the quantizer: many pruning criteria preferentially remove small-magnitude weights, leaving the retained nonzero weights split into negative and positive branches with a zero-centered empty interval between them. We refer to this interval as a *pruning-induced gap*. Standard quantizers do not explicitly account for this structure: they set the scale based on the full retained range and allocate quantization levels across that range, even though part of it contains no retained weights. Consequently, some quantization levels can be left unused, leading to inefficient use of both the quantization range and the available levels. This inefficiency is especially harmful at low bit-widths such as 3–4 bits, where every level matters.

We propose **GapSQ** to address this pruning–quantization mismatch. As shown in Fig. 1, GapSQ inserts a lightweight group-wise shifting step between pruning and quantization. It shifts the retained

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.

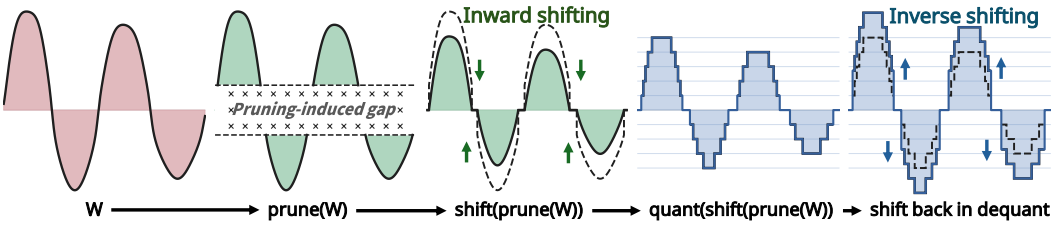


Figure 1: Overview of GapSQ. Pruning removes many near-zero weights and induces a central gap. GapSQ shifts the retained weights inward before quantization, reducing the effective dynamic range seen by the quantizer. During dequantization, the shift is reversed.

positive and negative weights inward by a shared amount, narrowing the pruning-induced gap and reducing the effective dynamic range seen by the quantizer. This allows the available quantization levels to be allocated more effectively to the retained weights.

A key challenge, however, is reversibility: a naive sign-dependent shift would require storing the sign of every retained weight, introducing per-weight overhead. GapSQ avoids this cost with a simple recoverability condition that preserves separability between the negative and positive branches after quantization. The original branch of each retained weight can then be inferred from its quantized level, allowing inverse shifting with only group-level metadata. The shift magnitude is selected through a lightweight calibration-based search that minimizes activation-aware reconstruction error.

Our contributions are threefold. First, we characterize and quantify the pruning-induced gap as an underexplored source of inefficiency in training-free joint compression of LLMs, showing that post-pruning weight distributions can lead to under-utilization of the quantization range and available levels. Second, we propose GapSQ, a simple, training-free, and plug-in joint compression method that addresses this inefficiency through recoverable group-wise inward shifting while requiring only modest group-level metadata. Third, across Llama and Qwen models, multiple bit-widths, and compression backbones, we show that GapSQ consistently improves strong baselines while remaining practical in storage and inference, highlighting the value of accounting for pruning-induced gaps in joint compression.

2 Related Work

Post-training pruning for LLMs. Network pruning reduces model size by removing weights with limited influence on model behavior. Representative post-training pruning methods for LLMs include SparseGPT (Frantar & Alistarh, 2023), which uses approximate second-order information to prune weights while updating the remaining weights to reduce local reconstruction error, and Wanda (Sun et al., 2024), which prunes weights using an activation-aware importance score. Although these methods differ in how they estimate importance, they commonly remove many low-magnitude weights. As a result, pruning reshapes the retained-weight distribution and often leaves an empty region around zero, which is the phenomenon of interest in our work.

Post-training quantization for LLMs. Post-training quantization lowers numerical precision to reduce storage and memory-bandwidth costs. GPTQ (Frantar et al., 2023) reduces quantization error through approximate second-order compensation, while AWQ (Lin et al., 2024) protects salient weights through activation-aware scaling. Other methods, such as QuaRot (Ashkboos et al., 2024) and SpinQuant (Liu et al., 2025), improve low-bit quantization through rotations that flatten weight distribution. These approaches mainly target dense or rotated dense distributions, where the dominant difficulty is often outliers or activation imbalance. In contrast, our focus is the discontinuous distribution produced by pruning, where an empty central interval creates a different form of inefficiency for quantizers.

Joint pruning and quantization. Joint pruning and quantization provides a natural route to more aggressive LLM compression than either technique alone. Frantar & Alistarh (2023) explored combining post-training pruning with quantization and showed that the two techniques can be used together to achieve more aggressive compression. More recently, OBR (Guo et al., 2026) studies joint compression in a rotated domain, where its error-compensation mechanism enables fully quan-

Table 1: **Analysis of pruning-induced gaps.** Statistics are averaged across all quantization groups at 50% sparsity. $\bar{\rho}_{\text{gap}}$: normalized gap width; $\bar{\rho}_{\text{unused}}$: fraction of unused 4-bit RTN levels; $\bar{\eta}_{\text{gain}}$: estimated MSE reduction under hypothetical gap removal. All values are in %.

Pruning method	Llama 3-8B			Qwen2.5-7B		
	$\bar{\rho}_{\text{gap}}$	$\bar{\rho}_{\text{unused}}$	$\bar{\eta}_{\text{gain}}$	$\bar{\rho}_{\text{gap}}$	$\bar{\rho}_{\text{unused}}$	$\bar{\eta}_{\text{gain}}$
Magnitude	26.56	24.95	45.92	25.45	25.20	44.21
SparseGPT	23.10	21.78	40.49	20.40	20.42	36.21
Wanda	20.73	19.70	36.82	18.17	18.46	32.67

tized W/A/KV compression. A recent study on compression order (Kim et al., 2026) further shows that the ordering of pruning and quantization can affect model quality across different compression ratios. Our work is complementary to these efforts: GapSQ focuses on the pruning-induced gap in post-pruning weight distributions and the resulting mismatch with subsequent quantization under aggressive compression. By adapting the post-pruning retained-weight distribution before quantization, GapSQ can be integrated with different pruning and quantization backbones.

3 Characterizing Pruning-Induced Gaps in Joint Compression

Before presenting GapSQ, we empirically characterize the pruning-induced gap with three group-level statistics: (*i*) its normalized width, (*ii*) the fraction of quantization levels left unused after post-pruning quantization, and (*iii*) a coarse estimate of the reduction in quantization error obtainable under hypothetical gap removal. Together, these statistics illustrate how the post-pruning weight distribution can create a systematic mismatch for subsequent quantization.

We perform our analysis at the granularity of weight groups, since modern LLM quantizers typically assign scales and zero-points independently per group. All statistics in this section are reported at 50% sparsity. Let $\mathbf{w} \in \mathbb{R}^g$ denote a pruned weight group containing only retained weights.

Gap width. We define the pruning-induced gap width as the distance between the retained negative weight closest to zero and the retained positive weight closest to zero:

$$\Delta_{\text{gap}}(\mathbf{w}) = \min_{i:\, w_i>0} w_i - \max_{i:\, w_i<0} w_i. \tag{1}$$

We further define the retained range of the group as

$$r(\mathbf{w}) = \max_i w_i - \min_i w_i, \tag{2}$$

and the normalized gap width as

$$\rho_{\text{gap}}(\mathbf{w}) = \frac{\Delta_{\text{gap}}(\mathbf{w})}{r(\mathbf{w})}. \tag{3}$$

This quantity measures the gap width relative to the retained range of the group. As shown in Tab. 1, the effect is large and consistent across both model families and all three pruning methods, with average normalized gap widths ranging from 18.17% to 26.56%; that is, pruning leaves roughly one-fifth to one-quarter of the retained dynamic range empty.

Unused quantization levels. We next quantify how the pruning-induced gap affects level utilization under quantization. For simplicity, we consider standard 4-bit group-wise round-to-nearest (RTN) quantization with $L = 16$ levels. Let $q_i(\mathbf{w}) \in \{0, \ldots, L-1\}$ denote the quantization index assigned to weight w_i. We define the unused-level ratio as

$$\rho_{\text{unused}}(\mathbf{w}) = 1 - \frac{1}{L} \left| \{ q_i(\mathbf{w}) : i = 1, \ldots, g \} \right|. \tag{4}$$

That is, $\rho_{\text{unused}}(\mathbf{w})$ is the fraction of quantization levels that receive no weights in the group. Since the pruning-induced gap contains no retained weights, quantization levels allocated to this region can remain unused. As shown in Tab. 1, the average unused-level ratio ranges from 18.46% to 25.20% across both model families and all three pruning methods, indicating substantial under-utilization of the available levels.

Estimated gain from gap removal. The preceding statistics naturally raise the following question: *how much could quantization error decrease if the pruning-induced gap were hypothetically removed*? For a standard uniform quantizer, the weight reconstruction error typically scales with the square of the quantization step size. Since the step size is proportional to the weight range, we have

$$\mathrm{MSE}(\mathbf{w}) \propto r(\mathbf{w})^2. \tag{5}$$

If the gap were removed entirely while preserving the relative geometry within the retained negative and positive branches, the effective range would contract to

$$r'(\mathbf{w}) = r(\mathbf{w}) - \Delta_{\mathrm{gap}}(\mathbf{w}) = \big(1 - \rho_{\mathrm{gap}}(\mathbf{w})\big) r(\mathbf{w}). \tag{6}$$

This yields the following estimated relative reduction in quantization error:

$$\eta_{\mathrm{gain}}(\mathbf{w}) = 1 - \left(\frac{r'(\mathbf{w})}{r(\mathbf{w})}\right)^2 = 1 - \big(1 - \rho_{\mathrm{gap}}(\mathbf{w})\big)^2. \tag{7}$$

We compute $\bar{\eta}_{\mathrm{gain}}$ by applying Eq. (7) to each quantization group and averaging across groups. Although coarse, this estimate suggests substantial room for improvement: across both model families and all three pruning methods, the estimated gain ranges from *32.67% to 45.92%*. Together, these statistics show that pruning can create a substantial zero-centered gap, leave a nontrivial fraction of quantization levels unused, and introduce a systematic inefficiency for subsequent quantization. This observation motivates explicitly accounting for the pruning-induced gap before quantization, which we do next.

4 GapSQ: Recoverable Shifting of Pruning-Induced Gaps

As shown in Section 3, pruning-induced gaps can lead to inefficient use of the quantization range and available levels. To address this mismatch, we introduce GapSQ, which applies a lightweight recoverable shifting transformation between pruning and quantization. We first present the overall framework, then describe the group-wise inward shifting and its recoverable formulation, and finally explain how the shift is selected in practice.

4.1 GapSQ Pipeline

Let W denote the weight matrix of a linear layer and X denote the calibration activations. A standard prune-then-quantize baseline can be abstractly written as

$$Q = \mathrm{quant}\big(\mathrm{prune}(W; X); X\big).$$

GapSQ is designed as a general framework that builds upon standard pruning and quantization backbones (e.g., SparseGPT for pruning and GPTQ for quantization) by inserting a lightweight shifting transformation between the two stages. The complete GapSQ compression pipeline can be written as

$$Q = \mathrm{GapSQ}(W; X) = \mathrm{quant}\big(\mathrm{shift}(\mathrm{prune}(W; X)); X\big). \tag{8}$$

During inference, the shifted representation is first dequantized and then mapped back to the original sparse-weight domain through a lightweight inverse-shifting rule:

$$\widehat{W} = \mathrm{Dequant}_{\mathrm{GapSQ}}(Q). \tag{9}$$

We next detail these two operations: Section 4.2 introduces the group-wise shifting transformation $\mathrm{shift}(\cdot)$, while Section 4.3 describes how the shift is reversed during dequantization through $\mathrm{Dequant}_{\mathrm{GapSQ}}(\cdot)$.

4.2 Group-Wise Inward Shifting

GapSQ operates at the same granularity as the base quantizer. Consider a single quantization group $\mathbf{w} \in \mathbb{R}^g$ with a retained index set $\mathcal{R} \subseteq \{1, \ldots, g\}$ after pruning. We partition the retained indices into positive and negative subsets:

$$\mathcal{R}_+ = \{i \in \mathcal{R} : w_i > 0\}, \qquad \mathcal{R}_- = \{i \in \mathcal{R} : w_i < 0\}.$$

To reduce the effective range presented to the quantizer, GapSQ applies a shared inward shift to the retained weights. For a given nonnegative shift magnitude s, the shifted weights are

$$w_i^{(s)} = \begin{cases} w_i - s, & i \in \mathcal{R}_+, \\ w_i + s, & i \in \mathcal{R}_-. \end{cases} \tag{10}$$

This transformation narrows the pruning-induced gap by $2s$ and correspondingly contracts the effective dynamic range seen by the subsequent quantizer, allowing the available quantization levels to be concentrated more effectively on the retained weights.

However, naive inward shifting creates a practical challenge. If s is too large, the originally positive and negative branches can cross zero and overlap. Because the original branch must be identified to reverse the shift during dequantization, such overlap would require storing an additional 1-bit sign mask for every retained weight. This per-weight side information would substantially offset the memory and bandwidth benefits of joint compression. We therefore require a constraint that preserves recoverability without introducing per-weight metadata.

4.3 Recoverable Inverse Shifting During Dequantization

The key idea is to ensure that the originally positive and negative branches remain distinguishable in the *quantized* space. To this end, we introduce a group-wise *reference level* together with a strict *separability constraint*.

Let $q_i(s) \in \{0, \dots, 2^b - 1\}$ denote the quantization level assigned to the shifted weight $w_i^{(s)}$. We define the reference level as the lowest level occupied by the positive branch:

$$q_{\mathrm{ref}}(s) = \min_{i \in \mathcal{R}_+} q_i(s). \tag{11}$$

A shift $s > 0$ is considered *recoverable* if the negative branch remains strictly below this level:

$$\max_{i \in \mathcal{R}_-} q_i(s) < q_{\mathrm{ref}}(s). \tag{12}$$

The original branch of each retained weight can then be inferred directly from its quantized level.

Let $\hat{w}_i^{(s)}$ denote the standard dequantized value corresponding to $q_i(s)$. The reconstructed weight is

$$\hat{w}_i(s) = \begin{cases} \hat{w}_i^{(s)} + s, & q_i(s) \geq q_{\mathrm{ref}}(s), \\ \hat{w}_i^{(s)} - s, & q_i(s) < q_{\mathrm{ref}}(s), \end{cases} \qquad i \in \mathcal{R}. \tag{13}$$

Thus, the shift can be reversed without storing per-weight branch information.

Rationale for recoverable shifting. GapSQ creates a temporary representation tailored to quantization: shifting the retained weights inward narrows the pruning-induced gap and lets the available levels cover a tighter effective range. Because the shifting transformation can be strictly reversed during dequantization, this benefit is obtained without changing the underlying sparse structure.

4.4 Feasible Shift Range and Activation-Aware Selection

Having established recoverability, we now formulate how to constrain and select the shift. We first characterize its feasible range:

$$0 \leq s < \frac{1}{2}\Delta_{\mathrm{gap}}(\mathbf{w}), \tag{14}$$

where $\Delta_{\mathrm{gap}}(\mathbf{w})$ is the pruning-induced gap width defined in Eq. (1). Any shift that reaches or exceeds this upper bound causes the retained positive and negative branches to overlap in the shifted space, preventing their original branches from being reliably distinguished after quantization. We therefore restrict the inward shift to this interval.

Within the feasible continuous range, GapSQ performs a lightweight discrete grid search to select a group-wise shift. To guarantee a safe fallback, the candidate set always includes $s = 0$, which recovers the standard quantization baseline and for which the separability condition is waived. For

all other candidates ($s > 0$), we shift the group, quantize it, and check Eq. (12). We define the valid candidate set as

$$\mathcal{S}_{\text{valid}}(\mathbf{w}) = \{0\} \cup \left\{ s \in \mathcal{S}(\mathbf{w}) \setminus \{0\} \;\middle|\; s < \frac{1}{2}\Delta_{\text{gap}}(\mathbf{w}) \text{ and Eq. (12) holds} \right\}, \tag{15}$$

where $\mathcal{S}(\mathbf{w})$ denotes the discrete shift grid considered for the current group.

Among the valid candidates, we select the shift that minimizes the activation-aware local reconstruction error measured by output MSE:

$$s^\star = \arg\min_{s \in \mathcal{S}_{\text{valid}}(\mathbf{w})} \left\| X_g\big(\mathbf{w} - \hat{\mathbf{w}}(s)\big) \right\|_2^2, \tag{16}$$

where X_g denotes the calibration activations corresponding to the current weight group, and $\hat{\mathbf{w}}(s)$ is the reconstructed weight vector obtained via Eq. (13). This output-space objective aligns shift selection more closely with the layer output than criteria based solely on raw weight reconstruction.

The selected shift $s^\star$ and corresponding reference level $q_{\text{ref}}(s^\star)$ are stored per group and used by the inverse-shifting rule during dequantization. The shift is stored as a floating-point scalar, while $q_{\text{ref}}(s^\star)$ is an integer quantization-level index in $\{0, \ldots, 2^b - 1\}$ and can therefore be represented using b bits (e.g., 4 bits for 4-bit quantization). These are the only two additional group-level values required by GapSQ.

5 Experiments

Models and datasets. We evaluate GapSQ on instruction-tuned Llama 3 (3B, 8B) and Qwen2.5 (7B, 72B). Following prior work (Frantar & Alistarh, 2023; Sun et al., 2024; Frantar et al., 2023), we use 128 length-2048 C4 sequences (Raffel et al., 2020) for calibration. We report WikiText-2 perplexity (Merity et al., 2016) and zero-shot accuracy on ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021).

Setup and baselines. We use 50% unstructured sparsity with 4- or 3-bit weight quantization, a group size of 128, and 8 shift candidates per group. Our primary baseline is SparseGPT+GPTQ, a strong joint compression extension explored in SparseGPT (Frantar & Alistarh, 2023). For the 4-bit setting, we additionally include dense 3-bit GPTQ (Frantar et al., 2023) as a matched-bit-width baseline, following prior work (Frantar & Alistarh, 2023). We further evaluate GapSQ with other pruning and quantization backbones. C4 (Raffel et al., 2020) is used for calibration throughout. Unless otherwise specified, the experiments are run on NVIDIA A100 GPUs.

5.1 Experiment Results

Main results. As shown in Tab. 2, GapSQ consistently improves SparseGPT+GPTQ across all evaluated models and both 4- and 3-bit settings at 50% sparsity, in terms of both average zero-shot accuracy and WikiText-2 perplexity. At a matched bit width, the 4-bit sparse GapSQ also outperforms dense 3-bit GPTQ in both metrics across all four models, suggesting that joint sparsification and quantization can offer greater potential than quantization alone under the same storage budget. The gains become more pronounced under more aggressive compression regimes. In particular, on Llama 3-3B with 3-bit quantization and 50% sparsity, GapSQ improves average zero-shot accuracy from **47.6% to 57.8% (+10.2 points)** and reduces WikiText-2 perplexity from **359.47 to 19.48**. Overall, these results highlight the benefit of adapting post-pruning weight distributions for low-bit joint compression, especially as the compression setting becomes more challenging.

Compatibility with other pruning masks and quantization methods. As suggested by the analysis in Sec. 3, GapSQ should improve joint compression quality across different pruning and quantization methods. We verify this in Fig. 2 using two pruning methods, Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023), together with two quantization methods, the classic round-to-nearest (RTN) and GPTQ (Frantar et al., 2023). While pairings other than SparseGPT+GPTQ (Frantar & Alistarh, 2023) are not standard combinations that have been formally studied in prior work,

Table 2: **Evaluation of GapSQ across model families and compression settings.** We report six zero-shot accuracies (%), average accuracy, and WikiText-2 perplexity. The best average accuracy and perplexity within each comparable compression group are shown in **bold**. ΔAvg. and ΔWiki2 denote changes relative to the corresponding SparseGPT+GPTQ baseline.

Model	Method	#Bits	Sparsity ratio	Arc-C (↑)	BoolQ (↑)	HellaS. (↑)	MMLU (↑)	PIQA (↑)	WinoG. (↑)	Avg. (↑)	Δ*Avg.* (↑)	Wiki2 (↓)	Δ*Wiki2* (↓)
Llama 3-3B	Original	16	0%	46.2	78.5	70.5	62.3	75.4	67.3	66.7	-	11.05	-
	GPTQ (quant-only)	3	0%	24.8	50.6	31.3	44.0	58.7	58.1	44.6	-	713.47	-
	SparseGPT+GPTQ	4	50%	34.4	76.3	59.0	47.1	71.2	60.9	58.2	-	20.36	-
	GapSQ (Ours)	4	50%	36.0	76.1	60.3	49.5	70.1	64.3	**59.4**	*+1.2*	**18.06**	*-2.30*
	SparseGPT+GPTQ	3	50%	26.6	58.4	40.6	37.6	64.2	58.0	47.6	-	359.47	-
	GapSQ (Ours)	3	50%	34.7	75.6	58.9	46.1	70.3	61.3	**57.8**	*+10.2*	**19.48**	*-339.99*
Llama 3-8B	Original	16	0%	55.1	84.1	79.3	68.3	80.7	74.4	73.6	-	7.22	-
	GPTQ (quant-only)	3	0%	47.4	77.3	73.7	50.7	75.6	67.6	65.4	-	28.97	-
	SparseGPT+GPTQ	4	50%	45.2	79.2	71.1	53.5	76.9	69.8	66.0	-	12.24	-
	GapSQ (Ours)	4	50%	47.2	78.6	72.0	56.0	77.0	71.6	**67.1**	*+1.1*	**10.58**	*-1.66*
	SparseGPT+GPTQ	3	50%	35.4	63.8	65.4	46.4	69.3	63.0	57.2	-	30.52	-
	GapSQ (Ours)	3	50%	39.2	75.2	69.7	50.4	71.8	68.8	**62.5**	*+5.3*	**11.72**	*-18.80*
Qwen 2.5-7B	Original	16	0%	56.4	86.7	74.9	72.9	77.6	67.8	72.7	-	8.61	-
	GPTQ (quant-only)	3	0%	47.3	81.7	75.7	62.3	76.0	64.0	67.8	-	10.63	-
	SparseGPT+GPTQ	4	50%	48.1	83.4	72.7	62.4	76.8	69.5	68.8	-	10.85	-
	GapSQ (Ours)	4	50%	50.6	85.1	73.9	64.9	78.1	69.4	**70.3**	*+1.5*	**10.34**	*-0.51*
	SparseGPT+GPTQ	3	50%	44.4	78.9	69.9	59.9	76.4	65.6	65.8	-	12.69	-
	GapSQ (Ours)	3	50%	46.4	79.1	72.0	59.1	76.7	68.5	**67.0**	*+1.2*	**11.81**	*-0.88*
Qwen 2.5-72B	Original	16	0%	62.7	89.1	86.1	83.4	83.5	77.7	80.4	-	3.88	-
	GPTQ (quant-only)	3	0%	61.3	89.5	85.7	81.3	82.5	74.0	79.0	-	6.44	-
	SparseGPT+GPTQ	4	50%	61.4	88.3	83.5	77.1	82.1	77.4	78.3	-	6.85	-
	GapSQ (Ours)	4	50%	61.6	90.1	85.4	80.0	83.0	77.0	**79.5**	*+1.2*	**5.91**	*-0.94*
	SparseGPT+GPTQ	3	50%	59.8	86.4	77.9	76.3	81.4	75.4	76.2	-	8.51	-
	GapSQ (Ours)	3	50%	60.5	90.5	84.7	79.1	82.6	76.1	**78.9**	*+2.7*	**7.37**	*-1.14*

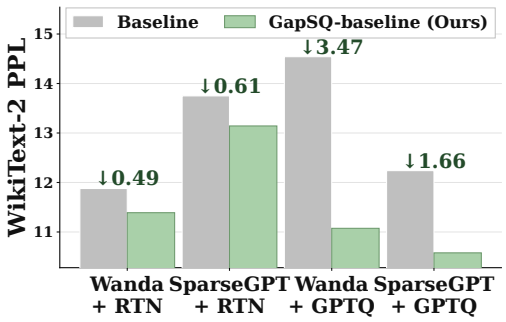


Figure 2: **Compatibility of GapSQ with diverse pruning–quantization backbones.** Consistent gains are observed throughout.

Table 3: GapSQ achieves consistent improvements across **different sparsity patterns.**

Methods	Sparsity	0-shot ↑	Wiki2 ↓
SparseGPT+GPTQ	2:4	55.2	22.72
GapSQ (Ours)	2:4	**56.4**	**18.66**
SparseGPT+GPTQ	4:8	61.5	17.04
GapSQ (Ours)	4:8	**62.1**	**14.00**

Table 4: **Compression time** for 4-bit quantization with 50% sparsity; The compression cost of GapSQ remains practical.

Methods	Llama 3-3B	Llama 3-8B
SparseGPT+GPTQ	10m17s	19m11s
GapSQ (Ours)	20m24s	35m57s

they nevertheless serve as reasonable baselines. Across all four combinations, GapSQ consistently reduces the perplexity, showing that its gain is robust across different choices of pruning mask and quantization backend. The strongest result is obtained with GapSQ instantiated with SparseGPT+GPTQ (our default), which achieves the best final perplexity of **10.58** among all tested combinations. Additional results with AWQ (Lin et al., 2024) and OmniQuant (Shao et al., 2024) in Appendix C further show that the benefit of GapSQ extends to other quantization methods.

Compatibility with other sparsity patterns. We further evaluate GapSQ under hardware-friendly semi-structured $N:M$ sparsity, considering both 2:4 and 4:8 patterns. In particular, 2:4 sparsity is natively supported by NVIDIA Sparse Tensor Cores (Mishra et al., 2021). As shown in Tab. 3, GapSQ consistently improves SparseGPT+GPTQ (Frantar & Alistarh, 2023) in both settings. Under 2:4 sparsity, it improves the zero-shot average from 55.2% to 56.4% and reduces WikiText-2 perplexity from 22.72 to 18.66; under 4:8 sparsity, it improves the average from 61.5% to 62.1%

Table 5: **Inference efficiency of GapSQ relative to dense FP16 inference.** Left: standalone kernel speedup with 4-bit 2:4 sparsity on NVIDIA A100 GPUs. Right: vLLM serving latency (ms) for Llama 3-8B, shown as **Dense → GapSQ**, with 64 input tokens, 64 output tokens, and QPS=1.

Batch size	**1**	**8**	**64**	**128**
Llama 3-8B	2.63×	2.67×	1.48×	1.17×
Qwen2.5-72B	3.48×	3.39×	1.81×	1.14×

Hardware	**TTFT ↓**	**TPOT ↓**	**E2EL ↓**
GH200	14.02→11.33	5.64→3.11	369.17→207.49
A100	18.44→15.36	13.17→11.45	848.39→737.02

Table 6: **Effect of the number of shift candidates**. Performance largely saturates after 4–8 candidates, and thus we use 8 candidates in all experiments.

Baseline Wiki2 perplexity: 12.239					
#Candidates	2	4	8	16	32
Wiki2 ↓	11.087	10.596	10.578	10.577	10.581

Table 7: **GapSQ with different shift selection criteria**. Output MSE performs best.

Criteria	**0-shot ↑**	**Wiki2 ↓**
Weight MSE	66.4	10.73
Output MAE	66.8	10.62
Output MSE (Ours)	**67.1**	**10.58**

and reduces perplexity from 17.04 to 14.00. These results suggest that the gains of GapSQ extend beyond unstructured sparsity to practical semi-structured patterns.

5.2 Practical Efficiency

Compression overhead. We report the wall-clock compression time of GapSQ in Tab. 4. On Llama 3-3B, compression time increases from `10m17s` for SparseGPT+GPTQ to `20m24s` for GapSQ, while on Llama 3-8B it increases from `19m11s` to `35m57s`. The additional cost mainly comes from the group-wise shift search. Since candidate grids can be precomputed and multiple groups can be evaluated in batch, the overall compression cost remains practical.

Kernel-level speedup. We implement GapSQ based on Sparse-Marlin (Frantar et al., 2024) to evaluate its kernel-level speedup. As shown in Tab. 5 left, GapSQ consistently accelerates inference over dense FP16 on both Llama 3-8B and Qwen2.5-72B with 4-bit 2:4 sparsity on NVIDIA A100 GPUs. The speedup reaches up to **2.67**× on Llama 3-8B and **3.48**× on Qwen2.5-72B.

vLLM serving efficiency. We further integrate GapSQ into the vLLM serving stack and evaluate Llama 3-8B with 64 input tokens, 64 output tokens, and QPS=1. We report time to first token (TTFT), time per output token (TPOT), and end-to-end latency (E2EL). As shown in Tab. 5 right, GapSQ consistently reduces all three latency metrics relative to dense FP16 on both NVIDIA GH200 and A100. In particular, on GH200, E2EL is reduced **from 369.17 ms to 207.49 ms**. These results show that the inference speedup of GapSQ also carries over to an end-to-end serving environment. We provide additional overhead analysis in Appendix E.

5.3 Ablation Studies

Effect of shift search budget. We study the sensitivity of GapSQ to the number of shift candidates (Tab. 6). Even with only 2 candidates, WikiText-2 perplexity decreases from 12.239 to 11.087, a 9.4% reduction over the baseline. Increasing the budget to 4 candidates further reduces perplexity to 10.596, while 8 candidates achieve 10.578; additional candidates provide negligible improvement. We therefore use 8 candidates throughout, which reduces perplexity by 13.6% relative to the baseline while keeping the search budget relatively small.

Choice of shift selection criterion. We study three reconstruction-based criteria for shift selection in GapSQ: weight MSE ($\|W - \hat{W}\|_2^2$), output MAE ($\|XW - X\hat{W}\|_1$), and output MSE ($\|XW - X\hat{W}\|_2^2$). As shown in Tab. 7, the results indicate that output-space criteria outperform weight-space reconstruction, and that output MSE is slightly better than output MAE, achieving the best performance of 67.1% zero-shot average accuracy and 10.58 WikiText-2 perplexity. We therefore use output MSE as the default shift selection criterion in all settings.

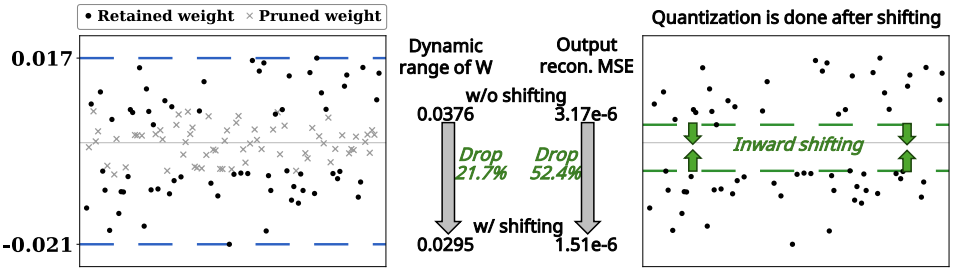


Figure 3: **Effect of GapSQ** on a single weight group. GapSQ shifts the weights inward, contracting the effective dynamic range by *21.7%* and reducing the output reconstruction MSE by *52.4%*.

Table 8: **Pushing joint compression to 2 bits** on Llama 3-8B. GapSQ † extends the shift search to include outward shifts, while GapSQ ‡ additionally uses an unrounded floating-point zero-point.

Method	Sparsity ratio	Arc-C (↑)	BoolQ (↑)	HellaS. (↑)	MMLU (↑)	PIQA (↑)	WinoG. (↑)	Avg. (↑)	Δ*Avg.* (↑)	Wiki2 (↓)	Δ*Wiki2* (↓)
GPTQ (quant-only)	0%	26.6	44.1	29.4	24.3	53.0	49.9	37.9	-	419.13	-
SparseGPT+GPTQ	50%	22.9	55.6	34.9	24.6	55.9	50.8	40.8	-	121.36	-
GapSQ † (Ours)	50%	23.5	61.4	41.5	25.2	55.1	53.4	43.3	*+2.5*	41.66	*-79.70*
GapSQ ‡ (Ours)	50%	24.4	62.4	53.1	26.4	56.0	54.9	**46.2**	*+5.4*	**24.27**	*-97.09*

5.4 Analysis & Discussion

Visualizing the shifting mechanism. To illustrate the effect of GapSQ, Fig. 3 shows a representative weight group (more in Appendix J). By shifting the retained weights inward, GapSQ contracts the effective dynamic range by **21.7%** and reduces the output MSE by **52.4%**, showing that the improved quantization geometry translates into substantial output-space reconstruction gains.

Toward extreme 2-bit joint compression. We further study GapSQ on Llama 3-8B under the extreme 2-bit, 50% sparsity setting (Tab. 8). With only four quantization levels, adjacent levels of a uniform 2-bit quantizer are separated by roughly one third of the represented range. A pruning-induced gap of around 20% can therefore be smaller than a single level spacing and need not manifest as an explicitly unused level. Instead, adapting the post-pruning distribution remains useful for improving how the extremely limited quantization levels are allocated over the retained weights.

To provide more flexibility in this extreme regime, we modify only the compression-time search. Specifically, we extend the shift space from nonnegative inward shifts to signed shifts, allowing outward shifts by replacing the lower bound in Eq. (14) with $-\max_i |w_i|$, and increase the search budget from 8 to 16 candidates. This variant, GapSQ †, leaves the deployed representation, stored metadata, and inverse-shifting procedure unchanged. It improves average zero-shot accuracy from **40.8% to 43.3%** and reduces WikiText-2 perplexity from **121.36 to 41.66** relative to SparseGPT+GPTQ. Allowing an unrounded floating-point zero-point further improves the results to **46.2%** average accuracy and **24.27** perplexity. Overall, these results show that accounting for the pruning-induced gap remains beneficial even under extreme 2-bit joint compression.

6 Conclusion

We characterized the pruning-induced gap in post-pruning weight distributions as an important source of inefficiency in joint sparsification and quantization of LLMs, showing how it can lead to under-utilization of the available quantization range and levels. Building on this observation, we introduced GapSQ, a training-free framework that adapts post-pruning weight distributions through recoverable shifting. Across Llama and Qwen models, 3- and 4-bit quantization, multiple sparsity patterns, and different compression backbones, GapSQ consistently improves strong joint-compression baselines, with particularly pronounced gains under more aggressive low-bit settings. Our evaluation is currently limited to LLMs; extending the underlying idea to broader architectures and modalities, such as CNNs, ViTs, and multimodal models, is an interesting direction for future work.

AI USE STATEMENT

We used generative AI tools as auxiliary aids for language polishing and wording refinement, as well as for helping improve the consistency and clarity of mathematical notation. We also used them for a limited amount of routine and repetitive coding, such as generating batch job scripts with different arguments for existing pipelines and extracting specified results from experiment logs.

Generative AI was not used to develop the core research idea, analysis, or methodology; formulate the central mathematical results; implement the core algorithms; or determine the substantive structure and scientific narrative of the paper. All AI-assisted text was reviewed and revised by the authors. AI-assisted scripts were manually inspected and tested, and extracted experimental results were checked against the original logs. The authors take full responsibility for all content, claims, code, and reported results in this work.

ETHICS STATEMENT

To the best of our knowledge, this work raises no major ethical concerns. It does not involve human subjects or sensitive private data, and no new datasets are collected or released. As with other LLM compression methods, aggressive compression may affect model reliability, and compressed models should therefore be evaluated carefully before deployment in safety-sensitive applications.

REPRODUCIBILITY STATEMENT

We have documented the experimental settings, datasets, compression configurations, and hardware environments used throughout the paper. Upon acceptance, we will release the code required to reproduce our results, together with detailed instructions on dependencies, environment setup, and evaluation procedures.

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. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

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

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of NAACL-HLT*, pp. 2924–2936, 2019.

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.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan-Adrian Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *11th International Conference on Learning Representations*, 2023.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. *arXiv preprint arXiv:2408.11743*, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Hang Guo, Yawei Li, and Luca Benini. Optimal brain restoration for joint quantization and sparsification of llms. In *14th International Conference on Learning Representations*, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations*, 2021.

Minjun Kim, Jaehyeon Choi, Hyunwoo Yang, Jongjin Kim, Jinho Song, and U Kang. Prune-then-quantize or quantize-then-prune? understanding the impact of compression order in joint model compression. In *14th International Conference on Learning Representations*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

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, 2024.

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 *13th International Conference on Learning Representations*, 2025.

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

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

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.

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, Gao Peng, Yu Qiao, and Ping Luo. Omniquant: Omnidirectionally calibrated quantization for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45472–45496, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *12th International Conference on Learning Representations*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *12th International Conference on Learning Representations*, 2024.

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, 2019.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Table 9: **Evaluation on the Llama 3-8B Base model.** We report six zero-shot accuracies, average accuracy, and WikiText-2 perplexity. The best average accuracy and perplexity within each comparable compression group are shown in **bold**. ΔAvg. and ΔWiki2 denote changes relative to the corresponding SparseGPT+GPTQ baseline.

Model	Method	#Bits	Sparsity ratio	Arc-C (↑)	BoolQ (↑)	HellaS. (↑)	MMLU (↑)	PIQA (↑)	WinoG. (↑)	Avg. (↑)	Δ*Avg.* (↑)	Wiki2 (↓)	Δ*Wiki2* (↓)
Llama 3-8B (Base)	Original	16	0%	53.5	82.1	78.9	63.1	81.2	74.4	72.2	-	6.24	-
	GPTQ (quant-only)	3	0%	37.2	68.7	67.9	54.0	71.9	69.4	61.5	-	27.82	-
	SparseGPT+GPTQ	4	50%	40.2	74.9	69.5	47.7	75.1	70.4	63.0	-	10.81	-
	GapSQ (Ours)	4	50%	42.8	77.5	70.1	48.2	76.1	71.0	**64.3**	*+1.3*	**9.67**	*-1.14*
	SparseGPT+GPTQ	3	50%	37.5	72.7	60.8	41.9	74.4	67.6	59.1	-	42.07	-
	GapSQ (Ours)	3	50%	38.2	79.4	68.2	45.3	75.2	68.4	**62.5**	*+3.4*	**10.81**	*-31.26*

Table 10: **Evaluation across different sparsity ratios on Llama 3-8B.** We use 4-bit quantization and report six zero-shot accuracies, average accuracy, and WikiText-2 perplexity. The best average accuracy and perplexity within each sparsity setting are shown in **bold**. ΔAvg. and ΔWiki2 denote changes relative to the corresponding SparseGPT+GPTQ baseline.

Method	#Bits	Sparsity ratio	Arc-C (↑)	BoolQ (↑)	HellaS. (↑)	MMLU (↑)	PIQA (↑)	WinoG. (↑)	Avg. (↑)	Δ*Avg.* (↑)	Wiki2 (↓)	Δ*Wiki2* (↓)
Original	16	0%	55.1	84.1	79.3	68.3	80.7	74.4	73.6	-	7.22	-
SparseGPT+GPTQ	4	20%	52.30	83.46	78.54	64.82	81.07	74.11	72.38	-	8.67	-
GapSQ (Ours)	4	20%	53.69	83.21	78.48	65.26	80.79	74.29	**72.62**	*+0.24*	**7.62**	*-1.05*
SparseGPT+GPTQ	4	30%	52.02	81.66	77.20	64.09	80.20	74.11	71.55	-	9.02	-
GapSQ (Ours)	4	30%	52.22	82.11	77.78	65.24	79.76	74.27	**71.90**	*+0.35*	**7.91**	*-1.11*
SparseGPT+GPTQ	4	40%	49.23	81.22	75.44	61.61	79.92	73.56	70.16	-	9.60	-
GapSQ (Ours)	4	40%	50.43	82.75	75.78	61.78	79.27	74.51	**70.75**	*+0.59*	**8.60**	*-1.00*

A Evaluation on the Llama 3-8B Base Model

Our main experiments use instruction-tuned models. We further evaluate GapSQ on the Llama 3-8B base model under the same 50% sparsity and quantization settings. As shown in Tab. 9, GapSQ consistently improves SparseGPT+GPTQ at both 4 and 3 bits. At 4 bits, GapSQ improves average zero-shot accuracy from 63.0% to 64.3% and reduces WikiText-2 perplexity from 10.81 to 9.67. The gains become more pronounced at 3 bits, where average accuracy improves from 59.1% to 62.5% and perplexity decreases substantially from 42.07 to 10.81. These results show that adapting post-pruning weight distributions also benefits base models, particularly under more challenging low-bit compression.

B Evaluation Across Different Sparsity Ratios

We further evaluate GapSQ across different sparsity ratios on Llama 3-8B with 4-bit quantization. As shown in Tab. 10, GapSQ consistently improves SparseGPT+GPTQ at 20%, 30%, and 40% sparsity in terms of both average zero-shot accuracy and WikiText-2 perplexity. In particular, WikiText-2 perplexity is reduced from 8.67 to 7.62, 9.02 to 7.91, and 9.60 to 8.60 at 20%, 30%, and 40% sparsity, respectively, while average accuracy also improves consistently. These results show that the benefit of adapting post-pruning weight distributions is not tied to a specific sparsity ratio.

C Joint Compression with Other Quantization Methods

We further evaluate whether GapSQ generalizes to quantization methods beyond GPTQ by combining SparseGPT with AWQ (Lin et al., 2024) and OmniQuant (Shao et al., 2024) on Llama 3-8B. We consider three configurations with the same effective 3-bit weight-storage budget: dense 3-bit quantization, 4-bit quantization with 50% sparsity, and 3-bit quantization with 33.3% sparsity. The latter provides a complementary setting with less aggressive pruning. In particular, SparseGPT alone at 50% sparsity gives a WikiText-2 perplexity of 10.50, already worse than dense 3-bit OmniQuant (9.80) and AWQ (9.55), indicating that pruning error can already be substantial before quantization. Since GapSQ is designed to reduce the quantization error arising from the mismatch between

Table 11: **Joint compression with AWQ and OmniQuant on Llama 3-8B.** All configurations have the same effective 3-bit weight-storage budget. We report six zero-shot accuracies, their average, and WikiText-2 perplexity. ΔAvg. and ΔWiki2 denote changes relative to the corresponding joint-compression baseline without GapSQ. The best average accuracy and perplexity within each quantizer group are shown in **bold**.

Method	#Bits	Sparsity ratio	Arc-C (↑)	BoolQ (↑)	HellaS. (↑)	MMLU (↑)	PIQA (↑)	WinoG. (↑)	Avg. (↑)	Δ*Avg.* (↑)	Wiki2 (↓)	Δ*Wiki2* (↓)
OmniQuant												
OmniQuant (quant-only)	3	0%	45.3	79.1	73.8	57.4	76.7	67.9	66.7	-	9.80	-
SparseGPT+OmniQuant	4	50%	46.2	75.1	70.0	54.3	76.7	70.9	65.5	-	11.04	-
SparseGPT+OmniQuant	3	33.3%	48.1	79.7	72.8	56.4	77.7	71.4	67.7	-	9.89	-
GapSQ-SparseGPT+OmniQuant	4	50%	46.7	77.4	70.8	55.4	76.3	71.7	66.4	*+0.9*	10.78	*-0.26*
GapSQ-SparseGPT+OmniQuant	3	33.3%	48.9	80.1	73.9	56.2	78.0	71.7	**68.1**	*+0.4*	**9.62**	*-0.27*
AWQ												
AWQ (quant-only)	3	0%	48.2	80.6	72.9	58.2	78.1	70.7	68.1	-	9.55	-
SparseGPT+AWQ	4	50%	43.3	79.8	69.8	54.9	76.1	70.0	65.7	-	11.07	-
SparseGPT+AWQ	3	33.3%	47.3	80.6	73.6	57.3	78.0	71.3	68.0	-	9.57	-
GapSQ-SparseGPT+AWQ	4	50%	45.2	80.3	70.3	55.7	76.8	70.7	66.5	*+0.8*	10.83	*-0.24*
GapSQ-SparseGPT+AWQ	3	33.3%	48.1	80.5	74.1	58.1	77.5	72.0	**68.4**	*+0.4*	**9.42**	*-0.15*

Table 12: **Additional semi-structured $N:M$ sparsity results** at 4-bit. We report perplexity on WikiText-2 and C4 for 2:4 and 4:8 sparsity patterns. GapSQ consistently improves the baseline across different models and structured sparsity settings.

Model	Sparsity	Baseline			Ours		
		Method	Wiki2 ↓	C4 ↓	Method	Wiki2 ↓	C4 ↓
Llama 3-3B	2:4	SparseGPT+GPTQ	44.51	48.40	GapSQ	37.23 (***-7.28***)	43.20 (***-5.20***)
	4:8	SparseGPT+GPTQ	36.24	38.65	GapSQ	23.99 (***-12.25***)	30.99 (***-7.66***)
Qwen 2.5-7B	2:4	SparseGPT+GPTQ	15.74	23.86	GapSQ	14.73 (***-1.01***)	23.07 (***-0.79***)
	4:8	SparseGPT+GPTQ	12.77	20.80	GapSQ	12.45 (***-0.32***)	20.39 (***-0.41***)

post-pruning weight distributions and subsequent quantization, the 33.3%-sparse setting allows us to evaluate the method when pruning error is less dominant.

As shown in Tab. 11, GapSQ consistently improves the corresponding joint-compression baseline for both AWQ and OmniQuant under both sparsity/bit allocations. The 3-bit, 33.3%-sparse configuration further shows that these gains can translate into an advantage over quantization alone under the same effective storage budget. With OmniQuant, GapSQ improves the joint-compression baseline from 67.7% to **68.1%** average accuracy and from 9.89 to **9.62** WikiText-2 perplexity, surpassing dense 3-bit OmniQuant (66.7%, 9.80). Similarly, with AWQ, GapSQ improves the corresponding baseline from 68.0% to **68.4%** and from 9.57 to **9.42**, also outperforming dense 3-bit AWQ (68.1%, 9.55). These results show that the benefit of adapting post-pruning weight distributions extends beyond RTN and GPTQ, while an appropriate balance between sparsity and quantization precision can make joint compression competitive with, and even better than, quantization alone under the same storage budget.

D Additional Semi-Structured N:M Sparsity Results

Tab. 12 reports additional 4-bit results under $N:M$ sparsity, with perplexity evaluated on WikiText-2 and the C4 validation set. Across all tested models and both sparsity patterns, GapSQ consistently improves the SparseGPT+GPTQ (Frantar & Alistarh, 2023) baseline. These results further support that the benefit of GapSQ extends beyond unstructured sparsity to practical, more hardware-friendly semi-structured sparsity patterns.

E Additional vLLM Serving Analysis

We further analyze the serving overhead introduced by GapSQ relative to the underlying Sparse-Marlin (Frantar et al., 2024) implementation. In our experiments, Dense Marlin uses dense 4-bit

Table 13: **Detailed vLLM serving latency on Llama 3-8B.** All values are in milliseconds with 64 input tokens, 64 output tokens, QPS=1, and TP=1. Dense Marlin uses dense 4-bit GPTQ-quantized weights, while Sparse-Marlin additionally uses 2:4 sparsity.

NVIDIA GH200

Method	TTFT	TPOT	E2EL
Dense FP16	14.02	5.64	369.17
Dense Marlin	11.62	3.32	220.72
Sparse-Marlin	11.30	3.01	200.95
GapSQ	11.33	3.11	207.49

NVIDIA A100

Method	TTFT	TPOT	E2EL
Dense FP16	18.44	13.17	848.39
Dense Marlin	15.92	11.86	763.48
Sparse-Marlin	15.33	11.14	725.89
GapSQ	15.36	11.45	737.02

Table 14: **Effect of compression order** using the SparseGPT+GPTQ pipeline on Llama 3-8B with 50% sparsity and 4-bit quantization.

Compression order	0-shot ↑	Wiki2 ↓
Prune → Quantize	**66.0**	**12.24**
Quantize → Prune	64.3	30.16

GPTQ-quantized weights, while Sparse-Marlin combines 4-bit GPTQ quantization with 2:4 sparsity and serves as the kernel basis for GapSQ. *The only additional inference-time computation of GapSQ is the inverse-shifting operation in Eq. (13).*

We evaluate Llama 3-8B with 64 input tokens, 64 output tokens, QPS=1, and tensor parallelism of 1 on both NVIDIA GH200 and A100. Tab. 13 reports time to first token (TTFT), time per output token (TPOT), and end-to-end latency (E2EL).

Relative to Sparse-Marlin, GapSQ introduces only a small serving overhead: 0.3%/3.3%/3.3% in TTFT/TPOT/E2EL on GH200 and 0.2%/2.8%/1.5% on A100. Despite the additional inverse-shifting operation, GapSQ remains faster than both Dense Marlin and dense FP16 across all three metrics on both platforms, indicating that its inference overhead remains practical. We also observe that, in the current serving stack, adding sparsity on top of an already optimized dense low-bit kernel provides relatively modest additional gains. Further improving end-to-end sparse low-precision serving efficiency remains an important direction for the broader community.

F Discussion on Compression Order

Our main experiments follow the prune-then-quantize order. This is consistent with prior joint-compression methods: SparseGPT+GPTQ (Frantar & Alistarh, 2023; Frantar et al., 2023) and OBR (Guo et al., 2026) both adopt this order. The recent compression-order study (Kim et al., 2026) also reports prune-then-quantize as the better choice in the low-bit regime relevant to our setting. Our results on Llama 3-8B at 50% sparsity and 4-bit quantization show the same trend (Tab. 14).

GapSQ is designed for post-pruning weights, so the current formulation does not directly apply to quantize-then-prune pipelines. Extending this idea to the reverse order is left for future work.

G Testing with Different Calibration Sets

Tab. 15 evaluates GapSQ with different calibration sets on Llama 3-8B under 4-bit quantization with 50% sparsity. We consider three commonly used choices, WikiText2 (Merity et al., 2016), Pile (Gao et al., 2020), and C4 (Raffel et al., 2020), and report perplexity on both WikiText2 and C4. Across all three calibration sets, GapSQ consistently improves over SparseGPT+GPTQ (Frantar & Alistarh, 2023), indicating that its gains are not tied to a particular calibration source.

H Storage Accounting and Metadata-Matched Comparison

We first make the storage accounting of GapSQ explicit. For 4-bit quantization with 50% sparsity and group size 128, the retained quantized weights require 2 bits per original weight on average,

Table 15: **Testing with different calibration sets** on Llama 3-8B at 4 bits with 50% sparsity. We report WikiText2 and C4 perplexities when calibrating on WikiText2, Pile, or C4. GapSQ consistently improves over SparseGPT+GPTQ across all three calibration sources.

Calib. Set	Baseline			Ours		
	Method	Wiki2 ↓	C4 ↓	Method	Wiki2 ↓	C4 ↓
WikiText2	SparseGPT+GPTQ	9.82	17.92	GapSQ	9.41 (***-0.41***)	17.39 (***-0.53***)
Pile	SparseGPT+GPTQ	11.77	18.29	GapSQ	11.05 (***-0.72***)	17.73 (***-0.56***)
C4	SparseGPT+GPTQ	12.24	17.87	GapSQ	10.58 (***-1.66***)	16.37 (***-1.50***)

Table 16: **Comparison across quantization group sizes** on Llama 3-8B with 4-bit quantization and 50% sparsity. Because GapSQ doubles the group-level metadata, *its group sizes 128 and 64 are metadata-matched to baseline group sizes 64 and 32, respectively*.

Group Size	Baseline			Ours		
	Method	Wiki2 ↓	C4 ↓	Method	Wiki2 ↓	C4 ↓
128	SparseGPT+GPTQ	12.24	17.87	GapSQ	10.58 (***-1.66***)	16.37 (***-1.50***)
64	SparseGPT+GPTQ	11.70	16.98	GapSQ	10.51 (***-1.19***)	16.32 (***-0.66***)
32	SparseGPT+GPTQ	11.33	16.74	GapSQ	10.50 (***-0.83***)	16.19 (***-0.55***)

Table 17: **Compatibility with the fully quantized OBR framework** on Llama 3-8B. We compare GapSQ-OBR with OBR following the official OBR setting. The best average accuracy and WikiText-2 perplexity are shown in **bold**. ΔAvg. and ΔWiki2 denote changes relative to OBR.

Method	W/A/K/V	Sparsity	Arc-C	BoolQ	HellaS.	MMLU	PIQA	WinoG.	Avg.	Δ*Avg.*	Wiki2	Δ*Wiki2*
QuaRot	3/4/4/4	0%	24.2	42.5	37.2	23.0	54.6	51.2	38.8	-	103.35	-
OBR	4/4/4/4	50%	31.3	70.6	57.8	34.3	68.8	58.9	53.6	-	19.41	-
GapSQ-OBR (Ours)	4/4/4/4	50%	34.9	70.3	58.8	36.8	71.0	57.2	**54.8**	*+1.2*	**18.24**	*-1.17*

while the pruning mask typically requires another 1 bit per weight. Following our implementation, SparseGPT+GPTQ stores an FP16 scale (16 bits) and a 4-bit integer zero-point per group. Under this accounting, the baseline requires

$$2 + 1 + \frac{16 + 4}{128} = 3.15625 \tag{17}$$

bits per original weight, corresponding to 19.7% of dense FP16 weight storage.

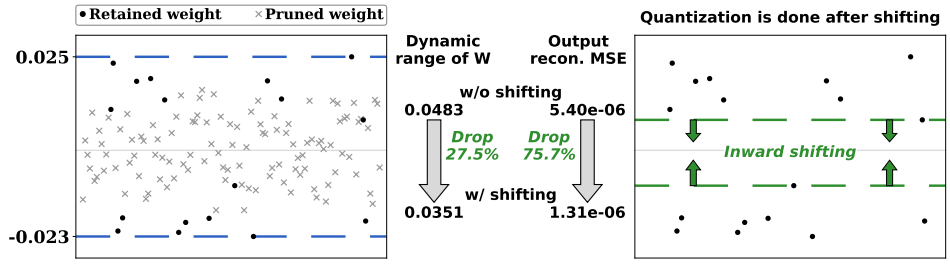
GapSQ additionally stores an FP16 shift (16 bits) and a 4-bit integer reference level per group, giving

$$2 + 1 + \frac{16 + 4 + 16 + 4}{128} = 3.3125 \tag{18}$$

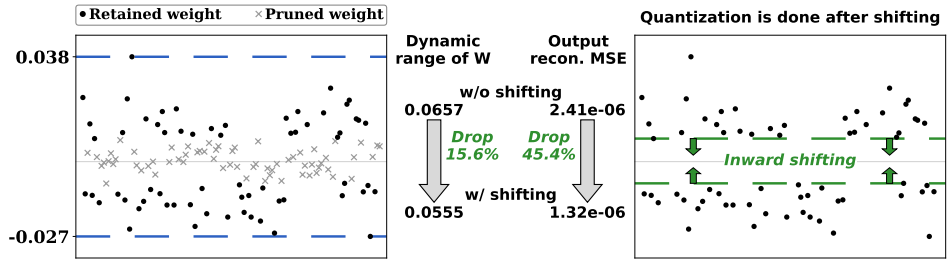
bits per original weight, corresponding to **20.7%** of dense FP16 weight storage. Thus, although GapSQ doubles the group-level metadata from 20 to 40 bits per group, the overall storage increases by only 1.0 percentage point, from 19.7% to 20.7% of dense FP16 weight storage.

We further examine whether the improvement can be explained simply by this additional metadata. Since GapSQ uses 40 bits of group-level metadata per group compared with 20 bits for SparseGPT+GPTQ, GapSQ with group size 128 has the same metadata budget per weight as SparseGPT+GPTQ with group size 64. Similarly, GapSQ with group size 64 is metadata-matched to the baseline with group size 32.

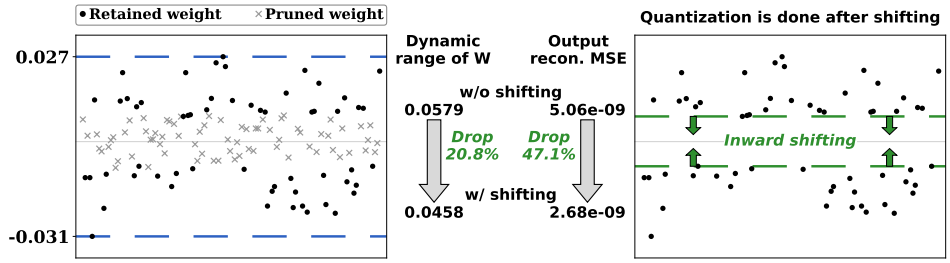
As shown in Tab. 16, GapSQ with group size 128 still achieves lower WikiText-2/C4 perplexity than the metadata-matched group-size-64 baseline (10.58/16.37 vs. 11.70/16.98). It also outperforms the group-size-32 baseline (11.33/16.74), despite the latter using twice as much group-level metadata per weight as GapSQ with group size 128. Similarly, at the second metadata-matched operating point, GapSQ with group size 64 achieves 10.51/16.32 perplexity compared with 11.33/16.74 for the group-size-32 baseline. These results indicate that ***the gains of GapSQ cannot be explained by additional metadata alone***.



(a) Representative group from the `k_proj` layer of Llama 3-8B. GapSQ contracts the dynamic range by 27.5% and reduces the output MSE by 75.7%.



(b) Representative group from the `up_proj` layer of Llama 3-8B. GapSQ contracts the dynamic range by 15.6% and reduces the output MSE by 45.4%.



(c) Representative group from the `down_proj` layer of Llama 3-8B. GapSQ contracts the dynamic range by 20.8% and reduces the output MSE by 47.1%.

Figure 4: **Additional group-level visualizations of GapSQ.**

I Compatibility with Fully Quantized Joint Compression

We further evaluate whether GapSQ can be integrated with OBR (Guo et al., 2026), which performs joint sparsification and quantization in a rotated domain and supports low-bit weight, activation, and KV-cache quantization. Following the OBR setting, we evaluate Llama 3-8B with randomized Hadamard transformations and compare dense W3/A4/K4/V4 QuaRot against OBR and GapSQ-OBR under 50% sparsity with W4/A4/K4/V4 quantization.

As shown in Tab. 17, GapSQ consistently improves OBR under this fully quantized setting, increasing average zero-shot accuracy from 53.6% to **54.8%** and reducing WikiText-2 perplexity from 19.41 to **18.24**. These results further show that the benefit of adapting post-pruning weight distributions is compatible with rotated joint-compression pipelines and extends beyond weight-only quantization.

J Additional Group-Level Visualization Results

To complement the single-group illustration in the main paper, Fig. 4 provides three additional examples from different layers of Llama 3-8B. In each case, we visualize the retained and pruned weights in one quantization group, the change in effective dynamic range before and after shifting,

and the corresponding output reconstruction MSE. Across these examples, inward shifting consistently contracts the effective range seen by the quantizer and reduces the local output reconstruction error. These visualizations further suggest that GapSQ is broadly effective across different layers and weight groups.