# SPARROTQUANT: PATTERNED SPARSITY-AWARE ROTATION-BASED QUANTIZATION FOR LLM COMPRESSION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Low-bit quantization and semi-structured N:M sparsity are two hardware-friendly techniques for efficient large language model deployment. However, their direct combination suffers from a structural conflict: rotation-based quantization smooths weight distributions to improve low-bit representations, whereas N:M sparsity relies on strong local contrast within each weight group to determine a stable keep/prune pattern. Consequently, applying N:M pruning after rotation produces unstable sparse masks and significant quality degradation. We propose SparRotQuant, a sparsity-aware rotation-based quantization framework that incorporates the N:M sparsity constraint into calibration, aligning the learned sparse support with the rotated low-bit representation used for deployment. To guide this process, we introduce Probability-aware Balanced Group Separation (PBGS), a group-level criterion that combines retention probability with Hessian-aware sensitivity to encourage each weight group to form a reliable sparse pattern before hard projection. With frozen pretrained weights, SparRotQuant learns differentiable sparse masks during calibration while adapting rotations and quantization parameters to the resulting sparse low-bit representation. Experiments across multiple model families show that SparRotQuant maintains strong model quality under W4A4KV16 and W4A4KV4 while preserving the efficiency benefits of sparse low-bit execution.

## 1 INTRODUCTION

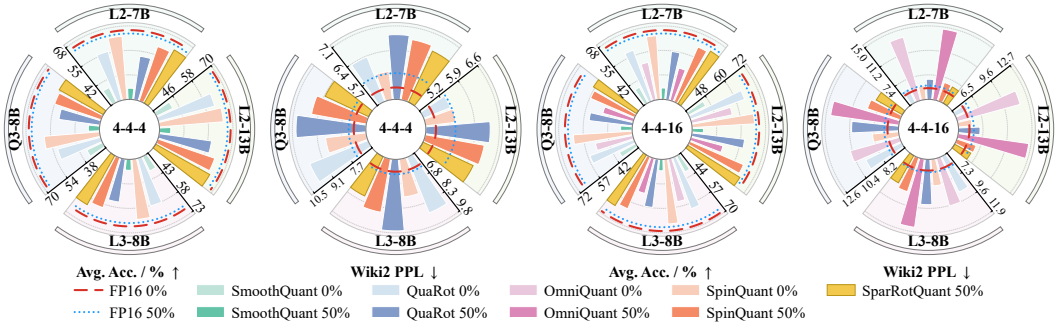



Figure 1: Zero-shot average accuracy (Avg.↑) and WikiText2 perplexity (Wiki2↓) under W4A4KV4 (left) and W4A4KV16 (right). (Sp. denotes the sparsity ratio; 50% refers to 2:4 structured sparsity; L2/L3/Q3 denote LLaMA2, LLaMA3 and Qwen3.)

Deploying large language models under realistic resource constraints requires reducing both memory footprint and computation cost. Low-bit quantization addresses the memory and computation bottlenecks by representing weights and activations with fewer bits (Shao et al., 2023; Lin et al., 2024a; Kim et al., 2023), while semi-structured N:M sparsity reduces effective arithmetic operations through hardware-friendly sparse patterns such as 2:4 sparsity (Mishra et al., 2021). These two directions appear naturally complementary for efficient LLM inference, but sparse low-bit deployment is not a simple composition of quantization and pruning.



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.

Recent rotation-based quantization methods show that equivalent transformations can substantially improve low-bit LLM inference. QuaRot (Ashkboos et al., 2024) applies randomized Hadamard rotations (Tseng et al., 2024) to suppress activation outliers and enable end-to-end 4-bit quantization, while SpinQuant (Liu et al., 2024) learns orthogonal rotations to reduce quantization error under low-bit weights, activations, and KV caches. These methods mainly evaluate rotations by how well they preserve the dense quantized model. Semi-structured pruning follows a different path: SparseGPT (Frantar & Alistarh, 2023) and Wanda (Sun et al., 2023) select masks using reconstruction- or importance-based criteria after the weight representation is fixed, while recent structured-pruning methods also explore differentiable optimization of sparse structures. Together, these works highlight the importance of rotation learning and sparse-structure optimization, but they leave open whether the rotated representation itself is suitable for stable N:M projection.

However, rotation-based sparse quantization exposes a structural conflict: low-bit quantization favors outlier-suppressed weights with smaller dynamic range, whereas pruning must preserve salient weights. Prior joint sparse-quantization methods recognize this misalignment. JSQ observes that pruning may retain outliers harmful to quantization and designs a sparsity metric to reduce this mismatch (Guo et al., 2024), while SLiM compensates for joint quantization and semi-structured sparsity error with a low-rank component (Mozaffari et al., 2024). These methods mainly minimize or compensate for compression error (Zhang et al., 2024), rather than evaluating whether the compressed representation resolves the local quantization–sparsity conflict.

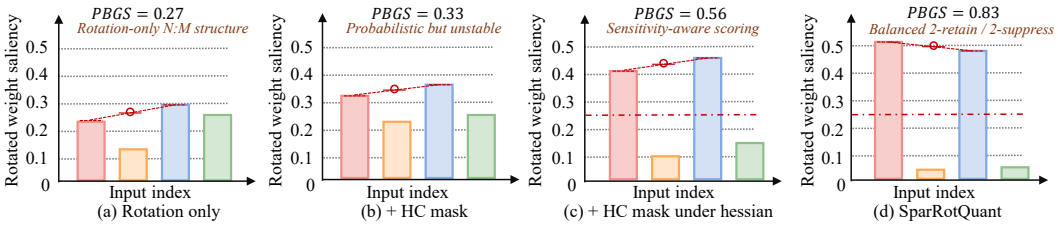



Figure 2: PBGS-guided saliency separation for N:M sparse quantization. Each panel visualizes a representative 4-element group in the rotated weight space. The bars show the relative saliency values $v_i = P_i H_i |\bar{W}_i|^2$ used by PBGS. PBGS evaluates the resulting group-level separation. The red dashed line with the center marker indicates the mean of the two dominant saliency values, summarizing the retained mass assigned to the candidate 2-retain subset. Panels (a)–(c) may leave ambiguous groups, while panel (d) shows the clearer retain/suppress separation promoted by PBGS.

To address this missing criterion, we introduce Probability-aware Balanced Group Separation (PBGS), a group-level metric that evaluates whether rotated weights are suitable for both quantization and sparsity. Rather than measuring only overall compression error, PBGS inspects each sparse group and combines retention probability with Hessian-aware sensitivity under frozen pretrained weights. It assigns higher scores when the important mass is concentrated on the entries to be retained, indicating that the rotated weights are better prepared for exact N:M projection.

Based on PBGS, we propose SparRotQuant, a sparsity-aware rotation-based quantization framework for sparse low-bit LLM deployment. SparRotQuant incorporates PBGS into calibration under Hessian information, where the rotation matrices and soft sparse masks are jointly optimized instead of being learned in separate stages. This allows the rotated weights to evolve toward the structure required by the final sparse deployment, rather than being quantized first and pruned afterward. As shown in Figure 2, the quantization-friendly rotation can flatten the within-group responses and weaken the contrast required by N:M pruning. Although hard-concrete masking makes the sparse pattern differentiable and Hessian-aware scoring improves element-wise importance estimation, neither directly optimizes the group-level separation required by structured sparsity. PBGS fills this gap by turning probability- and sensitivity-weighted element scores into a calibration signal that explicitly promotes a stable N:M sparse pattern.

Guided by PBGS, SparRotQuant integrates sparse-quant compatibility evaluation into rotation calibration, avoiding a sequential pipeline that first learns dense quantized rotations and then applies post-hoc N:M pruning. Our contributions are summarized as follows:

- We introduce Probability-aware Balanced Group Separation (PBGS), a group-level criterion for evaluating sparse-quant compatibility in rotated weight representations. PBGS

combines retention probability, Hessian-aware sensitivity, and rotated weight magnitude to assess whether each sparse group is structurally ready for exact N:M projection.

- We propose SparRotQuant, a sparsity-aware rotation-based quantization framework that embeds N:M sparsity into rotation calibration. SparRotQuant relaxes discrete keep-prune decisions into differentiable mask probabilities and jointly optimizes rotation matrices, sparse masks, and quantization parameters under the guidance of PBGS.
- We conduct systematic experiments under both W4A4KV16 and W4A4KV4 sparse low-bit settings, validating the effectiveness of SparRotQuant in maintaining model quality while preserving the inference-efficiency benefits of exact 2:4 sparse low-bit execution.

## 2 Related work

**Rotation-based quantization.** Rotation-based quantization improves low-bit deployment by applying equivalent transformations to reshape weights and activations without changing the full-precision function. QuaRot uses randomized Hadamard rotations to suppress activation outliers (Ashkboos et al., 2024), while SpinQuant learns orthogonal rotations with Cayley-based updates (Liu et al., 2024). Recent methods further improve this idea with learnable scaling in OSTQuant (Hu et al., 2025) and QR-parameterized distribution calibration in DartQuant (Shao et al., 2025). However, these objectives are mainly designed for dense quantized inference and do not explicitly model the local N:M structural constraints required by hardware-friendly sparse patterns.

**Hessian-guided compression.** Hessian information provides a principled estimate of parameter sensitivity under compression. GPTQ uses approximate Hessian information for layer-wise reconstruction after quantization (Frantar et al., 2022), while QuIP links quantization error to Hessian-based incoherence and improves low-bit quantization with orthogonal preprocessing and adaptive rounding (Chee et al., 2023). BoA further introduces attention-aware Hessian information to capture inter-layer dependency (Kim et al., 2024). These methods show the value of curvature for preserving compressed model quality, but mainly use it after the compression target is fixed.

**Structured N:M sparsity.** Semi-structured N:M sparsity offers a hardware-friendly pruning pattern by retaining a fixed number of nonzero weights in each local group, as supported by NVIDIA 2:4 sparse tensor cores. SparseGPT scales one-shot pruning to LLMs with patterns such as 2:4 and 4:8 (Frantar & Alistarh, 2023), while Wanda removes reconstruction or retraining by using weight magnitude and activation statistics (Sun et al., 2023). Yet these methods mainly select masks in the original weight space and treat pruning as a standalone step, ignoring that rotation-based low-bit deployment changes the local weight distribution and importance structure. Thus, a pre-rotation N:M mask may not match the quantized representation used at inference time.

**Learnable differentiable mask.**

Recent methods learn structured sparse patterns instead of relying on fixed pruning criteria. S2HPruner bridges soft relaxation and hard deployment through mask distillation (Lin et al., 2024b), and FlexHiNM-GP combines multi-level structured sparsity with permutation-aware pruning (Yi et al., 2026). However, these methods mainly optimize sparse patterns in a largely fixed representation, whereas SparRotQuant studies how learnable rotations, Hessian-guided sensitivity, and semi-structured mask formation should interact for low-bit LLM compression.

## 3 Preliminary analysis

**Observation 1: orthogonality-preserving mask placement.**

We first clarify where structured sparsity can be introduced in a rotation-based quantization framework without violating the underlying orthogonal reparameterization.

For a linear layer $Y = XW$, rotation gives the equivalent decomposition $Y = (XR)(R^{-1}W)$. Here, $R$ is the online rotation, while $R^{-1}W$ is the mergeable weight that can be combined offline and executed as a standard linear weight at inference time. If sparsity is imposed before this merge, the masked rotation component would be multiplied by $W$ and the sparse pattern would generally

be mixed into a dense matrix. Therefore, the mask should be applied only after the rotated weight has been formed, so that the exported matrix itself follows the target N:M pattern.

This observation constrains sparsity placement in a precise way. Let $W' = R^{-1}W$ denote the rotated mergeable weight. We impose structured sparsity only on $W'$, yielding $\widehat{W}' = M \odot W'$. The resulting forward pass becomes

$$\widehat{Y} = (XR)\widehat{W}' = (XR)\big(M \odot (R^{-1}W)\big). \tag{1}$$

Under Equation 1, the orthogonal structure of $R$ remains intact, while sparsity is introduced only in the deployable weight representation. This separation preserves the role of $R$ as a geometry-preserving basis transformation and confines structured approximation to the rotated weight space.

This formulation is not merely an implementation detail. Once sparsity is imposed in the rotated basis, quantization-oriented rotation learning and N:M-oriented structural regularization can be expressed in a shared coordinate system. The method developed in the next section builds directly on this formulation.

**Observation 2: coupled interface approximation under $R_{out}$-induced sparsity.**

The structural role of $R_{out}$ differs fundamentally from that of $R_{in}$, while $R_{attn}$ and $R_{ffn}$ are excluded from sparsification because they remain online and cannot be folded into static sparse linear weights (see Figure 3 for notations). While $R_{in}$ defines a shared orthogonal basis for the hidden representation and is calibrated consistently across the corresponding blocks, $R_{out}$ only appears at the local interface between the value projection and the output projection. Consequently, sparsity imposed on the $R_{in}$-induced merged weight yields a single-operator approximation error, whereas sparsity imposed on the $R_{out}$-induced merged weights gives rise to a coupled interface error.

For the $R_{in}$ case, let $Y = (XR_{in})W_1'$ with $W_1' = R_{in}^{-1}W$, and let $\hat{W}_1' = M_1 \odot W_1'$. Defining $E_1 := W_1' - \hat{W}_1'$, we have

$$Y - \hat{Y} = (XR_{in})E_1, \qquad \|Y - \hat{Y}\| \le \|X\|\,\|E_1\|, \tag{2}$$

where the last step follows from the orthogonality of $R_{in}$. The approximation error is therefore localized to a single sparse operator in a shared basis.

By contrast, for the $R_{\text{out}}$-mediated value–output interface, let $U$ denote the activation entering the value–output product, including the preceding input-side rotation and attention mixing. The local computation can be written as $Y = UW_v'W_o'$, where $W_v' = R_{\text{in}}^{-1}W_vR_{\text{out}}$ and $W_o' = R_{\text{out}}^{-1}W_oR_{\text{in}}$. If structured sparsity is applied to both merged weights, yielding $\hat{W}_v' = M_v \odot W_v'$ and $\hat{W}_o' = M_o \odot W_o'$, and we define $E_v := W_v' - \hat{W}_v'$ and $E_o := W_o' - \hat{W}_o'$, then $Y - \hat{Y} = U\big(W_v'W_o' - \hat{W}_v'\hat{W}_o'\big) = U\big(E_v\hat{W}_o' + \hat{W}_v'E_o + E_vE_o\big)$.

Accordingly we have Equation 3, which shows that $R_{out}$-induced sparsity introduces coupled perturbations across both sides of the interface, together with a second-order interaction term.

$$\|Y - \hat{Y}\| \le \|U\| \left(\|E_v\|\,\|\hat{W}_o'\| + \|\hat{W}_v'\|\,\|E_o\| + \|E_v\|\,\|E_o\|\right). \tag{3}$$

Such coupling can make fixed 2:4 approximation more difficult than in the $R_{in}$ case. We therefore avoid sparsifying both sides of the $R_{out}$-mediated interface and impose structured sparsity only on the selected input-side merged weights.

## 4 METHOD

SparRotQuant is designed for hardware-friendly N:M sparse low-bit deployment, using rotation-based quantization as the representation in which sparse patterns are learned. Standard rotation-based quantization improves the conditioning of weights and activations, but it does not control how weights are arranged within each N:M group, making direct pruning unstable when candidate entries have similar magnitudes or sensitivity. SparRotQuant addresses this issue by relaxing the discrete 2:4 constraint into differentiable mask probabilities, so the sparse support can shape the rotated representation during calibration. We further introduce Probability-aware Balanced Group Separation (PBGS), which encourages each 4-element group to concentrate probability-weighted sensitive mass on two dominant entries. After calibration, the soft masks are projected to exact 2:4 patterns, and the sparse weights are quantized for inference without retaining training-time components.

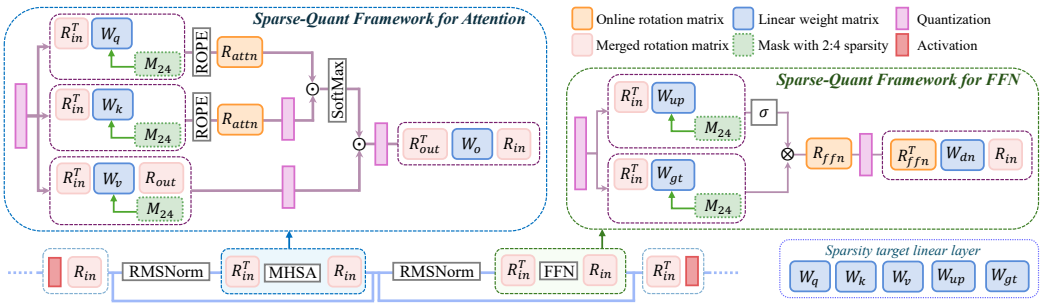



Figure 3: Overview of SparRotQuant.

### 4.1 Rotation-induced 2:4 sparsity

Introducing sparsity into rotation-based quantization requires careful placement of the sparse mask. Rotations are introduced as equivalent reparameterizations, whereas masking is a non-equivalent operation that removes part of the transformed weight. Therefore, applying the mask at an improper location can destroy the role of rotation rather than complement it. Placing sparsity on the output-side merged interface of $R_{out}$ in Figure 3 is also undesirable, since the resulting approximation error is shared by both the value projection and the output projection. As shown by Observation 2 in Section 3, this coupling makes the output-side formulation less suitable for fixed 2:4 approximation. SparRotQuant therefore applies the sparse mask to the input-rotated weight representation, where the approximation error remains localized to the target linear layer.

For a linear layer with input $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and weight $\boldsymbol{W} \in \mathbb{R}^{m\times d}$, the input-side rotation $\boldsymbol{R}_{in} \in \mathbb{R}^{d\times d}$ rewrites the original computation as

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}^\top = (\boldsymbol{X}\boldsymbol{R}_{in})\left(\boldsymbol{R}_{in}^{-1}\boldsymbol{W}^\top\right), \quad \boldsymbol{R}_{in}^\top\boldsymbol{R}_{in} = \boldsymbol{I}. \tag{4}$$

The sparse approximation is then introduced only on the rotated weight term:

$$\widehat{\boldsymbol{Y}} = (\boldsymbol{X}\boldsymbol{R}_{in})Q\left(\boldsymbol{M} \odot \boldsymbol{R}_{in}^{-1}\boldsymbol{W}^\top\right), \quad \boldsymbol{M} \in \{0,1\}^{d\times m}. \tag{5}$$

Here, $\boldsymbol{M}$ is the structured mask and $Q(\cdot)$ denotes the low-bit quantizer. This formulation follows the orthogonality-preserving placement derived in Observation 1 of Section 3: the rotation itself remains intact, and the non-equivalent sparse approximation is confined to the rotated weight.

This placement aligns calibration with the deployed sparse model. By applying the mask before quantization, SparRotQuant optimizes the quantizer on the same sparse support used at inference, avoiding the mismatch between dense rotated calibration and pruned deployment. It also changes the calibration target: beyond producing an outlier-smoothed dense representation, calibration now encourages local groups to become separable for exact N:M projection. As shown in Figure 3, SparRotQuant masks the input-rotated weights of $W_q$, $W_k$, $W_v$, $W_{up}$, and $W_{gt}$, which become the target sparse operators in attention and FFN blocks. After exact 2:4 projection, each target weight is folded with $R_{in}$ into a static sparse low-bit operator, while $R_{out}$ is folded into the subsequent projection weight. $R_{attn}$ and $R_{ffn}$ remain online because they cross attention or nonlinear FFN interfaces and cannot be absorbed into a single matrix.

### 4.2 Differentiable mask relaxation

The sparse support must be learned together with the input-side rotation, but the final 2:4 mask is discrete. A hard top-2 selection within each group blocks gradients and would reduce sparsity to a post-hoc pruning step after rotation learning. SparRotQuant therefore relaxes each binary keep/prune decision into a differentiable gate, so that the sparse support can be optimized during calibration. Throughout this process, the pretrained weights are frozen; only the rotation parameters, gate logits, and quantization parameters are updated.

**Gate parameterization.** Let $W' = \boldsymbol{R}_{in}^{-1}\boldsymbol{W}^\top \in \mathbb{R}^{d\times m}$ denote the input-rotated weight, and let $w'_i \in \mathbb{R}$ be one of its entries. Each $w'_i$ is associated with a learnable gate logit $\alpha_i \in \mathbb{R}$. Using the

hard-concrete relaxation (Maddison et al., 2016; Louizos et al., 2017), we sample a continuous gate $z_i \in [0, 1]$ as

$$u_i \sim \mathcal{U}(0,1), \quad s_i = \sigma\left(\frac{\log u_i - \log(1-u_i) + \alpha_i}{\beta}\right), \tag{6}$$

$$z_i = \min\left(1, \max\left(0, s_i(\zeta - \gamma) + \gamma\right)\right), \quad \beta > 0,\ \gamma < 0,\ \zeta > 1. \tag{7}$$

The sampled gate $z_i$ is differentiable almost everywhere with respect to $\alpha_i$, and serves as a relaxed surrogate of the binary mask entry during calibration. Thus, instead of applying a non-differentiable mask $M_i \in \{0, 1\}$, SparRotQuant uses $z_i$ to softly modulate the rotated weight.

At the matrix level, the relaxed sparse forward pass is

$$\widehat{\boldsymbol{Y}} = (\boldsymbol{X}\boldsymbol{R}_{in})Q\left(\boldsymbol{Z} \odot \boldsymbol{R}_{in}^{-1}\boldsymbol{W}^\top\right), \quad \boldsymbol{Z} \in [0,1]^{d\times m}. \tag{8}$$

This formulation creates a differentiable path from the calibration loss to the gate logits through $\boldsymbol{Z}$, and to the rotation through $W' = \boldsymbol{R}_{in}^{-1}\boldsymbol{W}^\top$. As a result, the sparse support and the rotated representation are optimized in the same calibration loop, while the original weights remain unchanged.

**Retention probability.** In addition to the sampled gate $z_i$, the hard-concrete relaxation provides a deterministic retention probability:

$$p_i = P(z_i > 0) = \sigma\left(\alpha_i - \beta \log \frac{-\gamma}{\zeta}\right), \quad p_i \in [0, 1]. \tag{9}$$

The gate $z_i$ is used to form the relaxed sparse forward pass, while $p_i$ describes the probability that the corresponding weight will survive the mask. Unlike a single gate sample, $p_i$ is noise-free and differentiable with respect to $\alpha_i$, making it more suitable for measuring the state of the soft sparse support.

### 4.3 Probability-aware group separation

Hard-concrete relaxation makes the mask searchable, but does not enforce a clear structure within each 2:4 group. Gate probabilities may remain diffuse or favor positions with limited output sensitivity, making the final top-2 projection fragile to quantization and rotation updates. We therefore introduce Probability-aware Balanced Group Separation (PBGS), a group-level objective that guides the relaxed mask toward a stable two-retained/two-pruned pattern before hard projection.

**Sensitivity-aware Hessian gated mass.** During calibration, the pretrained weights are frozen. SparRotQuant optimizes the rotation, gate logits, and quantization parameters, rather than directly updating the original weights. Let $\bar{W} = (W')^\top \in \mathbb{R}^{m\times d}$ denote the output-by-input view of the rotated weight, and use the same layout for $Z$, $P$, and $H$ below. For an entry $\bar{w}_{r,c}$ with gate $z_{r,c}$, the effective masked weight is $z_{r,c}\bar{w}_{r,c}$, where $r$ indexes the output channel and $c$ indexes the input channel. Suppressing this entry corresponds to closing the gate, and the induced local perturbation can be measured by a diagonal Hessian approximation (Kurtic et al., 2022):

$$\Delta\mathcal{L}_{r,c}(z) \approx \frac{1}{2}h_{r,c}\big((1 - z_{r,c})\bar{w}_{r,c}\big)^2, \quad z_{r,c} \in [0,1],\ h_{r,c} \in \mathbb{R}_+. \tag{10}$$

Here, $h_{r,c}$ is estimated from calibration data and measures the sensitivity of the reconstruction objective to suppressing this gated entry. This sensitivity prevents the mask search from selecting a visually valid but loss-unaware 2:4 pattern.

Given $\bar{W} \in \mathbb{R}^{m\times d}$, the retention probability $P \in [0,1]^{m\times d}$, and the non-negative sensitivity estimate $H \in \mathbb{R}_+^{m\times d}$, we define the probability-aware saliency score as

$$V = P \odot H \odot A, \qquad A_{ij} = |\bar{W}_{ij}|^2. \tag{11}$$

Each entry of $V$ measures how much sensitivity-weighted mass is assigned to the retained side of the relaxed mask. Large values correspond to positions that are likely to remain active and costly to suppress, as the same score is used for the final 2:4 projection after calibration.

**Group-level separation.** PBGS is computed over consecutive 4-channel groups, matching the N:M sparsity layout. For each output channel $r$ and group index $g$, we collect the local mass values as

$$V_{r,g} = V_{r,\,4g:4g+4} \quad \in \mathbb{R}_+^4. \tag{12}$$

Let $\mathrm{Top}_2(V_{r,g})$ and $\mathrm{Bot}_2(V_{r,g})$ denote the two largest and two smallest entries in this group. A stable relaxed group should satisfy two properties: its sensitive mass should concentrate on two positions, and the boundary between retained and pruned candidates should be clear.

We encode these properties with a dominance term and a separation term:

$$d_{r,g} = \frac{\sum \mathrm{Top}_2(V_{r,g})}{\sum V_{r,g} + \epsilon} \in [0,1], \quad s_{r,g} = \sigma\left(\frac{\min \mathrm{Top}_2(V_{r,g}) - \max \mathrm{Bot}_2(V_{r,g})}{T(\mathrm{mean}(V_{r,g}) + \epsilon)}\right) \in [0,1]. \quad (13)$$

The dominance term measures whether the group mass is concentrated on two candidates. The separation term measures whether the weakest retained candidate remains distinguishable from the strongest pruned candidate. Together, they indicate whether the evolving gate probabilities have formed a reliable 2:4 structure before discretization.

**PBGS objective.** PBGS averages the group scores over all target 2:4 groups:

$$\mathrm{PBGS} = \frac{1}{|\mathcal{G}|} \sum_{(r,g)\in\mathcal{G}} d_{r,g} s_{r,g}. \quad (14)$$

where $\mathcal{G}$ denotes the set of consecutive 4-channel groups in the target linear weights. A high PBGS value indicates that the relaxed support has assigned most probability-aware sensitive mass to two dominant positions, leaving the other two positions safely suppressible. This aligns the mask search with the final 2:4 deployment pattern, where the two entries with the largest $V$ in each group are retained.

The sparsity loss is defined as

$$\mathcal{L}_p = -\log(\mathrm{PBGS} + \epsilon). \quad (15)$$

Minimizing $\mathcal{L}_p$ does not learn the sparsity ratio; the ratio is fixed by the final 2:4 projection. Its role is to shape the differentiable mask search so that each local group becomes stable before projection. Thus, hard-concrete gates provide searchable mask variables $z_i$, while PBGS imposes the structural preference on those variables. The complete SparRotQuant pipeline is summarized in Algorithm A.1.

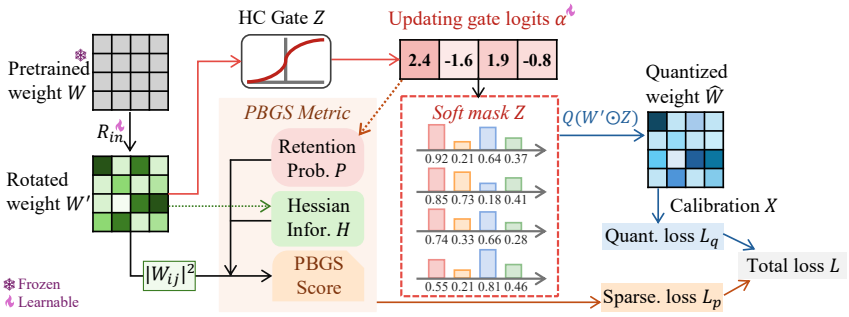



Figure 4: SparRotQuant calibration and deployment workflow.

### 4.4 Sparse Projection and Deployment

SparRotQuant only optimizes the rotation parameters, the gate logits $\alpha$, and the quantization parameters. The hard-concrete gates, denoted by $HC(\alpha)$ as defined in Section 4.2, provide two quantities: the relaxed mask $Z = HC(\alpha)$ for the sparse forward pass, and the retention probability $P = P(Z > 0)$ for PBGS. With $W' = R_{in}^{-1} W^\top$, the sparse-quantized weight used during calibration is

$$\widehat{W} = Q(Z \odot W'). \quad (16)$$

SparRotQuant uses a compact two-term objective:

$$\mathcal{L} = \mathcal{L}_q + \lambda_p \mathcal{L}_p, \qquad \mathcal{L}_p = -\log(\mathrm{PBGS}(P, H, \bar{W}) + \epsilon). \quad (17)$$

Here, $\mathcal{L}_q$ preserves the block output under the current sparse-quantized forward pass, while $\mathcal{L}_p$ guides differentiable mask search toward a stable two-retained/two-pruned pattern. The orthogonal rotations are updated with the Cayley-style optimization (Liu et al., 2024) used in rotation-based quantization, and PBGS couples this calibration process with sparse-structure learning. Hard-concrete relaxation does not introduce an additional sparsity loss, and the final 2:4 ratio is enforced

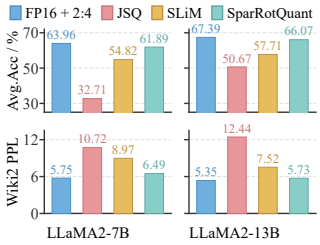



Figure 5: Comparison with joint sparse-quant. methods.

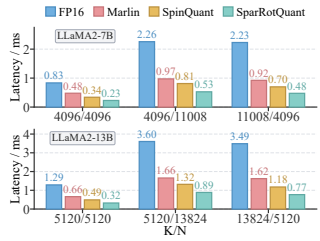



Figure 6: Prefill kernel latency (ms, $M$=4096).

Table 1: End-to-end prefill efficiency vs. FP16 (seq. 256–8192).

| Model | Method | Speedup↑ Avg. | Speedup↑ Range | Mem. Saving↑ |
|---|---|---|---|---|
| L2-7B | FP16 + 2:4 | 1.67× | 1.55–1.76× | 1.55× |
| | SpinQuant | 2.02× | 1.89–2.12× | 2.41× |
| | **SparRotQuant** | **2.56×** | **2.42–2.71×** | **3.60×** |
| L2-13B | FP16 + 2:4 | 1.79× | 1.66–1.88× | 1.60× |
| | SpinQuant | 2.24× | 2.10–2.35× | 2.44× |
| | **SparRotQuant** | **3.02×** | **2.78–3.27×** | **3.68×** |

by hard projection rather than an $L_0$-style penalty (Louizos et al., 2017). Figure 4 summarizes the SparRotQuant calibration deployment workflow, in which PBGS-guided differentiable mask learning is optimized together with the quantization loss to obtain the final sparse low-bit model. After exact 2:4 projection, we perform a short recovery calibration with the pretrained weights, learned rotations, and projected masks fixed, updating only the quantization parameters under $L_q$. The same recovery protocol is used for all compared sparse variants.

## 5 Experiments

We evaluate `SparRotQuant` on LLaMA2-7B and LLaMA2-13B (Touvron et al., 2023), LLaMA3-8B (Grattafiori et al., 2024), and Qwen3-8B (Yang et al., 2025) under sparse low-bit compression settings. On NVIDIA RTX 4090 GPUs, we report zero-shot accuracy on eight commonsense reasoning benchmarks: ARC-c, ARC-e (Clark et al., 2018), BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), and WinoGrande (Sakaguchi et al., 2021), together with WikiText2 perplexity (Merity et al., 2016). For efficiency, we report inference latency under identical runtime settings and averages over multiple runs.

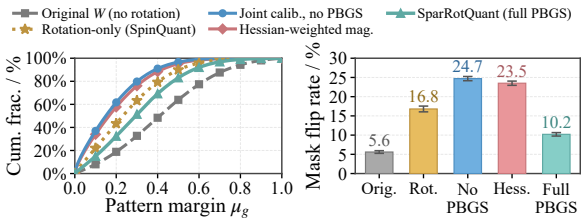
### 5.1 Accuracy Results

Figure 1 reports average zero-shot accuracy and WikiText2 perplexity across LLaMA2-7B/13B, LLaMA3-8B, and Qwen3-8B under W4A4KV16 and W4A4KV4. We compare `SparRotQuant` with `SmoothQuant` (Xiao et al., 2023), `QuaRot` (Ashkboos et al., 2024), `OmniQuant` (Shao et al., 2023), and `SpinQuant` (Liu et al., 2024), with and without 50% 2:4 sparsity. Across model families and quantization settings, `SparRotQuant` consistently improves over sparse `SpinQuant` in both average accuracy and WikiText2 perplexity, showing that sparsity-aware rotation calibration reduces the quality loss caused by exact 2:4 projection. Full task-level results are provided in Tables B.1–B.4, and C4-calibration robustness in Table B.6.

Figure 5 further compares `SparRotQuant` with joint sparse-quantization methods, including `JSQ` (Guo et al., 2024) and `SLiM` (Mozaffari et al., 2024), under the matched W4A4KV16 and exact 2:4 setting. `SparRotQuant` consistently outperforms both methods on the evaluated LLaMA2 models, demonstrating the empirical advantage of explicitly shaping the rotated representation for structured sparsity under joint quantization and exact 2:4 sparsity. Detailed task-level results are reported in Table B.5.

### 5.2 Inference Efficiency

Figure 6 first evaluates kernel-level efficiency on representative linear projections from LLaMA2-7B and LLaMA2-13B. This comparison isolates the complementary contributions of low-bit quantization and 2:4 structured sparsity: `SparRotQuant` reduces kernel latency by 72.7–78.7% compared with dense FP16 and by 45.8–52.6% compared with `Marlin-2:4` (Frantar et al., 2025), while also achieving a consistent 31.9–35.1% latency reduction over dense 4-bit `SpinQuant`. These results demonstrate that structured sparsity provides additional operator-level acceleration on top of low-bit quantization, with the same benefit extending to decode-shaped workloads. Detailed sparse




(a) Pattern separability (b) Optimal pattern flip rate

Figure 7: Mechanism analysis of 2:4 pattern formation. (a) Distribution of the pattern margin $\mu_g$. (b) Minimum-cost pattern flip rate across disjoint calibration subsets.

Table 2: Ablation on the components of the saliency $V_i$ on LLaMA2-7B under W4A4KV16.

| Variant | Saliency $V_i$ | Avg. | Wiki2 ↓ |
|---|---|---|---|
| No PBGS | – | 55.73 | 6.97 |
| Probability only | $p_i$ | 58.08 | 6.68 |
| Sensitivity only | $h_i$ | 59.47 | 6.60 |
| Magnitude only | $\lvert\bar{w}_i\rvert^2$ | 58.83 | 6.64 |
| Hessian-weighted mag. | $h_i\lvert\bar{w}_i\rvert^2$ | 60.52 | 6.53 |
| Wanda-as-$V$ | $\lvert\bar{w}_i\rvert\,\lVert x_i\rVert_2$ | 60.11 | 6.56 |
| **Full PBGS** | $\boldsymbol{p_i h_i \lvert\bar{w}_i\rvert^2}$ | **61.89** | **6.49** |

tensor core prefill and decode-shaped kernel results are provided in Table C.1 and Appendix C.3, respectively.

Table 1 summarizes end-to-end prefill efficiency from sequence lengths 256 to 8192, comparing `SparRotQuant` with FP16 2:4 sparsity and dense 4-bit `SpinQuant`. Across both model sizes, `SparRotQuant` achieves the highest speedup and lowest runtime memory through combined structured sparsity and low-bit storage. Detailed latency and memory results are reported in Tables C.2 and C.4.

### 5.3 Ablation Study

**Effect of sparsity placement.** We first evaluate where sparsity-aware rotation learning should be applied by comparing $R_{in}$-only, $R_{out}$-only, and joint $R_{in} + R_{out}$ optimization under W4A4KV16 with 50% structured sparsity. As shown in Table D.1, using only $R_{in}$ or $R_{out}$ yields similar model quality, while $R_{in}$ gives a smaller runtime memory footprint because sparsity covers a larger fraction of the transformed projection weights. Jointly sparsifying both rotation spaces leads to a clear quality degradation, suggesting a less favorable approximation tradeoff when sparsity is extended across both transformed spaces.

**Effect of PBGS components.** Table 2 shows that all partial PBGS variants improve over removing PBGS, but remain below sparse `SpinQuant`, while the full $p_i h_i |\bar{w}_i|^2$ formulation achieves the best accuracy and perplexity. Figure 7 further shows that full PBGS substantially recovers the pattern separability degraded by rotation and reduces mask instability. Together with the Wanda-style comparison, these results support the motivation of PBGS: rotation increases ambiguity near the 2:4 projection boundary, while PBGS restores a substantially clearer and more reproducible sparse pattern structure.

**Efficiency and calibration analysis.** Figure D.1 shows that structured pruning and low-bit quantization provide complementary efficiency gains, while `SparRotQuant` achieves the strongest combined speedup and memory reduction on both LLaMA2-7B and LLaMA2-13B. Figure D.2 further shows that `SparRotQuant` substantially reduces the degradation caused by exact 2:4 projection. A budget-matched comparison confirms that longer calibration alone does not explain this gain (Tables D.2–D.3).

## 6 Conclusion

We present SparRotQuant, a framework for N:M sparse low-bit LLM compression that integrates differentiable mask learning into rotation calibration and guides sparse structure formation with PBGS. By optimizing sparsity on the $R_{in}$ side, SparRotQuant aligns the rotated representation with the final exact N:M projection while avoiding coupled perturbations across rotation spaces. Experiments across multiple models validate that SparRotQuant maintains strong model quality under W4A4KV16 and W4A4KV4 while preserving the efficiency benefits of sparse low-bit execution.

### AI USE STATEMENT

We used generative AI tools only for language polishing and improving the clarity and readability of the manuscript. All technical content, experimental design, implementation, analysis, and conclusions were produced and verified by the authors. The authors take full responsibility for the final content of the paper.

### REPRODUCIBILITY STATEMENT

We provide the implementation details, model and dataset configurations, optimization settings, and evaluation protocols necessary to reproduce our main results in the paper and appendix. Additional experimental settings and ablation details are also reported in the supplementary material. We will release the code and supporting scripts to facilitate reproducibility.

## 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*, volume 34, pp. 7432–7439, 2020.

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

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 the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)*, 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.

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 Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Elias Frantar, Roberto L Castro, Jiale Chen, Torsten Hoefler, and Dan Alistarh. Marlin: Mixed-precision auto-regressive parallel inference on large language models. In *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, pp. 239–251, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jinyang Guo, Jianyu Wu, Zining Wang, Jiaheng Liu, Ge Yang, Yifu Ding, Ruihao Gong, Haotong Qin, and Xianglong Liu. Compressing large language models by joint sparsification and quantization. In *Forty-first International Conference on Machine Learning*, 2024.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *arXiv preprint arXiv:2501.13987*, 2025.

Junhan Kim, Ho-young Kim, Eulrang Cho, Chungman Lee, Joonyoung Kim, and Yongkweon Jeon. Boa: Attention-aware post-training quantization without backpropagation. *arXiv preprint arXiv:2406.13474*, 2024.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 4163–4181, 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, 2024a.

Weihao Lin, Shengji Tang, Chong Yu, Peng Ye, and Tao Chen. S2HPruner: Soft-to-hard distillation bridges the discretization gap in pruning. *Advances in Neural Information Processing Systems*, 37:116547–116574, 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. *arXiv preprint arXiv:2405.16406*, 2024.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through $l_0$ regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

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.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. Slim: One-shot quantization and sparsity with low-rank approximation for llm weight compression. *arXiv preprint arXiv:2410.09615*, 2024.

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

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

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

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. *arXiv preprint arXiv:2511.04063*, 2025.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

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

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

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 *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Xiaodie Yi, Hayun Lee, and Dongkun Shin. Flexhinm-gp: Flexible hierarchical pruning via region allocation and channel permutation. In *International Conference on Learning Representations*, volume 2026, pp. 97333–97352, 2026.

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.

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

# APPENDIX

## A DETAILS OF THE SPARROTQUANT PIPELINE

Algorithm A.1 summarizes the calibration and export procedure used by `SparRotQuant`. During calibration, the pretrained weights remain frozen, and only the rotation parameters, hard-concrete gate logits, and quantization parameters are updated. The Hessian-aware sensitivity is estimated from calibration activations for the same target linear layers on which sparse masks are learned. The shared $R_{in}$ is jointly calibrated using block-level reconstruction supervision, while the gate logits and quantization parameters remain specific to their corresponding target layers.

**Algorithm A.1** SparRotQuant pipeline

**Require:** Pretrained model $f$ w/ frozen weights $W$, calibration steps $T$, calibration set $\mathcal{D}$, target layers $\mathcal{T}$, quantizer $Q$

**Ensure:** Sparse-quantized model $\widehat{f}$

1: Initialize rotations $R$, gate logits $\alpha$, and quantization parameters;
2: **for** $t = 1, \ldots, T$ **do**
3: Sample $X \sim \mathcal{D}$;
4: Compute $W' \leftarrow R_{in}^{-1} W^\top$ for each $W \in \mathcal{T}$;
5: Generate $Z \leftarrow \mathrm{HC}(\alpha)$ and $P \leftarrow P(Z > 0)$;
6: Form $\widehat{W} \leftarrow Q(Z \odot W')$;
7: Estimate $H$ from calibration statistics;
8: Compute $\mathcal{L}_q$ and$\mathcal{L}_p \leftarrow -\log(\mathrm{PBGS}(P, H, (W')^\top) + \epsilon)$;
9: Update $R, \alpha$, and quantization parameters by minimizing $\mathcal{L} = \mathcal{L}_q + \lambda_p \mathcal{L}_p$;
10: **end for**
11: Project $V$ to exact 2:4 masks $M$ within each 4-channel group;
12: Export $\widehat{W} \leftarrow Q(M \odot W')$ and construct $\widehat{f}$;
13: Return $\widehat{f}$

After calibration, the probability-aware saliency scores are projected into exact 2:4 masks by retaining the two largest entries of $V$ within each consecutive 4-element group. After hard projection, we apply a short recovery calibration with the projected sparse support and learned rotations fixed, following the same protocol for all compared sparse variants. The reported final results use the recovered checkpoints.

## B SUPPLEMENTAL ACCURACY EVALUATION

### B.1 FULL TASK-LEVEL RESULTS

We provide the complete task-level results corresponding to the averaged model-quality results in Figure 1. Tables B.1–B.4 report results on LLaMA2-7B, LLaMA2-13B, LLaMA3-8B, and Qwen3-8B under W4A4KV16 and W4A4KV4. All methods follow the same evaluation protocol within each model and quantization setting. Across the four model configurations, applying exact 2:4 sparsity after dense low-bit calibration consistently degrades model quality, while `SparRotQuant` reduces this loss relative to sparse `SpinQuant`. Throughout this appendix, 50% sparsity denotes exact 2:4 structured sparsity applied to the selected target linear layers $W_q$, $W_k$, $W_v$, $W_{\mathrm{up}}$, and $W_{\mathrm{gt}}$. Although the improvement varies across individual tasks, the average accuracy and WikiText2 trends remain consistent across model families and quantization settings. These results support the main observation that incorporating the target sparse structure during rotation calibration improves robustness to the final exact 2:4 projection.

### B.2 COMPARISON WITH JOINT SPARSE-QUANTIZATION METHODS

We further compare `SparRotQuant` with methods that jointly address sparsity and quantization under a matched W4A4KV16 and exact 2:4 setting. We reimplement `JSQ` and `SLiM` following their reported configurations and use the same calibration and evaluation protocol under the

Table B.1: Zero-shot accuracy and WikiText2 perplexity of LLaMA2-7B.

| Bits | Method | Sparsity | ARC-c | ARC-e | BoolQ | PIQA | SIQA | HellaS. | OBQA | WinoG. | Avg. | Wiki2 ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16-16-16 | Full Precision | 0% | 49.34 | 73.19 | 76.58 | 78.12 | 46.63 | 75.27 | 57.40 | 69.48 | 65.75 | 5.34 |
| 16-16-16 | Full Precision | 50% | 47.86 | 71.95 | 74.81 | 76.91 | 46.09 | 72.94 | 54.27 | 66.87 | 63.96 | 5.75 |
| 4-4-16 | SmoothQuant | 0% | 26.74 | 35.06 | 50.13 | 49.56 | 38.22 | 29.25 | 28.60 | 50.43 | 38.50 | – |
| | QuaRot | 0% | 44.02 | 66.31 | 68.14 | 71.48 | 41.82 | 67.76 | 40.80 | 62.37 | 57.84 | 6.05 |
| | OmniQuant | 0% | 36.20 | 59.35 | 60.22 | 62.18 | 40.84 | 53.68 | 31.13 | 51.58 | 49.40 | 13.78 |
| | SpinQuant | 0% | 46.28 | 70.84 | 73.21 | 75.93 | 46.05 | 72.01 | 53.07 | 65.51 | 62.86 | 5.76 |
| | SmoothQuant | 50% | 24.47 | 29.62 | 48.31 | 45.87 | 33.08 | 25.11 | 26.73 | 47.09 | 35.04 | – |
| | QuaRot | 50% | 42.28 | 64.63 | 65.82 | 68.88 | 38.12 | 64.12 | 33.87 | 60.02 | 54.72 | 6.72 |
| | OmniQuant | 50% | 33.14 | 56.42 | 57.91 | 59.76 | 38.92 | 51.27 | 28.67 | 48.37 | 46.81 | 14.67 |
| | SpinQuant | 50% | 44.60 | 68.12 | 71.12 | 74.02 | 44.42 | 69.42 | 51.13 | 63.25 | 60.76 | 6.62 |
| | **SparRotQuant** | **50%** | **45.58** | **70.21** | **71.92** | **75.12** | **45.31** | **70.82** | **51.53** | **64.62** | **61.89** | **6.49** |
| 4-4-4 | SmoothQuant | 0% | 25.91 | 30.86 | 50.73 | 53.42 | 33.67 | 28.44 | 30.60 | 48.57 | 37.78 | – |
| | QuaRot | 0% | 41.09 | 65.56 | 67.87 | 70.49 | 40.88 | 66.79 | 37.93 | 57.44 | 56.01 | 6.37 |
| | SpinQuant | 0% | 46.07 | 71.22 | 72.68 | 75.79 | 45.96 | 71.86 | 52.80 | 64.97 | 62.67 | 5.81 |
| | SmoothQuant | 50% | 24.36 | 29.42 | 45.61 | 49.76 | 29.54 | 25.79 | 27.87 | 46.92 | 34.91 | – |
| | QuaRot | 50% | 36.05 | 62.10 | 63.98 | 68.05 | 38.55 | 63.45 | 31.87 | 54.34 | 52.30 | 6.85 |
| | SpinQuant | 50% | 42.35 | 67.55 | 69.45 | 73.40 | 42.10 | 68.95 | 49.40 | 62.85 | 59.51 | 6.75 |
| | **SparRotQuant** | **50%** | **45.22** | **69.74** | **71.41** | **74.73** | **44.98** | **70.36** | **51.00** | **64.21** | **61.46** | **6.61** |

Table B.2: Zero-shot accuracy and WikiText2 perplexity of LLaMA2-13B.

| Bits | Method | Sparsity | ARC-c | ARC-e | BoolQ | PIQA | SIQA | HellaS. | OBQA | WinoG. | Avg. | Wiki2 ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16-16-16 | Full Precision | 0% | 53.41 | 77.62 | 79.85 | 80.68 | 50.20 | 79.29 | 57.27 | 72.40 | 68.84 | 4.79 |
| 16-16-16 | Full Precision | 50% | 52.05 | 75.63 | 78.32 | 79.16 | 49.44 | 77.91 | 55.73 | 70.88 | 67.39 | 5.35 |
| 4-4-16 | SmoothQuant | 0% | 29.64 | 48.29 | 55.92 | 61.48 | 37.56 | 43.18 | 32.87 | 54.86 | 45.48 | – |
| | QuaRot | 0% | 48.41 | 73.06 | 76.54 | 78.18 | 47.30 | 75.19 | 47.20 | 68.51 | 64.30 | 5.18 |
| | OmniQuant | 0% | 41.55 | 66.32 | 67.48 | 70.64 | 43.77 | 63.29 | 37.53 | 60.46 | 56.38 | 11.62 |
| | SpinQuant | 0% | 50.77 | 75.85 | 78.07 | 79.49 | 49.41 | 77.46 | 51.47 | 70.77 | 66.65 | 5.13 |
| | SmoothQuant | 50% | 27.18 | 43.92 | 50.84 | 56.71 | 34.92 | 37.86 | 29.53 | 51.06 | 41.50 | – |
| | QuaRot | 50% | 46.84 | 71.24 | 73.95 | 76.44 | 43.88 | 72.86 | 44.13 | 66.27 | 61.95 | 6.12 |
| | OmniQuant | 50% | 37.94 | 62.41 | 62.77 | 66.83 | 40.62 | 56.31 | 33.80 | 56.72 | 52.18 | 12.41 |
| | SpinQuant | 50% | 48.55 | 74.16 | 76.24 | 77.85 | 47.53 | 76.14 | 48.87 | 68.64 | 64.75 | 5.96 |
| | **SparRotQuant** | **50%** | **50.28** | **75.28** | **77.64** | **79.04** | **48.77** | **77.26** | **50.13** | **70.16** | **66.07** | **5.73** |
| 4-4-4 | SmoothQuant | 0% | 27.83 | 44.61 | 52.38 | 58.79 | 35.82 | 39.74 | 31.07 | 52.64 | 42.86 | – |
| | QuaRot | 0% | 46.41 | 71.83 | 74.66 | 75.92 | 45.64 | 73.52 | 45.47 | 66.84 | 62.54 | 5.32 |
| | SpinQuant | 0% | 49.23 | 74.47 | 77.34 | 78.64 | 47.82 | 76.83 | 49.20 | 69.44 | 65.37 | 5.37 |
| | SmoothQuant | 50% | 25.96 | 41.32 | 48.47 | 54.63 | 32.94 | 35.28 | 28.47 | 48.95 | 39.50 | – |
| | QuaRot | 50% | 44.37 | 68.92 | 71.58 | 74.76 | 43.16 | 70.28 | 41.53 | 64.53 | 59.89 | 6.34 |
| | SpinQuant | 50% | 47.01 | 72.84 | 74.87 | 77.28 | 46.04 | 74.86 | 46.87 | 67.38 | 63.39 | 6.21 |
| | **SparRotQuant** | **50%** | **48.64** | **73.64** | **75.96** | **78.04** | **47.38** | **75.74** | **47.07** | **68.74** | **64.40** | **6.08** |

matched W4A4KV16 and exact 2:4 setting. The comparison includes `JSQ` (Guo et al., 2024) and `SLiM` (Mozaffari et al., 2024) on LLaMA2-7B and LLaMA2-13B. The same calibration and evaluation protocol is used for the matched comparisons, and results are reported over three independent runs. FP16 with 2:4 sparsity is included as a non-quantized reference.

Table B.5 provides the complete task-level results corresponding to Figure 5. `SparRotQuant` achieves the strongest average accuracy and WikiText2 perplexity among the compared compressed methods on both model sizes. The improvement is also larger than the observed run-to-run variation, supporting the stability of the measured advantage.

### B.3 Calibration corpus robustness

The main model-quality results use WikiText2 training data for calibration and evaluate perplexity on the held-out WikiText2 test set. To examine whether the relative improvement depends on the calibration corpus, we additionally calibrate sparse `SpinQuant` and `SparRotQuant` on C4 while keeping the remaining evaluation setting unchanged. In both cases, perplexity is evaluated on the same WikiText2 test set.

Table B.3: Zero-shot accuracy and WikiText2 perplexity of LLaMA3-8B.

| Bits | Method | Sparsity | ARC-c | ARC-e | BoolQ | PIQA | SIQA | HellaS. | OBQA | WinoG. | Avg. | Wiki2 ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16-16-16 | Full Precision | 0% | 57.73 | 77.79 | 83.38 | 80.89 | 48.15 | 79.42 | 51.20 | 73.67 | 69.03 | 6.05 |
| 16-16-16 | Full Precision | 50% | 56.05 | 75.31 | 79.35 | 78.72 | 46.84 | 77.48 | 50.27 | 71.87 | 66.99 | 6.21 |
| 4-4-16 | SmoothQuant | 0% | 26.08 | 36.48 | 52.41 | 53.74 | 40.39 | 31.79 | 30.27 | 54.36 | 40.69 | – |
| | QuaRot | 0% | 42.24 | 64.99 | 73.62 | 75.17 | 43.60 | 72.63 | 41.40 | 65.27 | 59.87 | 8.73 |
| | OmniQuant | 0% | 38.05 | 58.93 | 65.47 | 70.36 | 41.37 | 62.88 | 36.53 | 60.68 | 54.28 | 9.63 |
| | SpinQuant | 0% | 51.92 | 74.33 | 78.13 | 77.32 | 46.59 | 76.03 | 49.80 | 68.93 | 65.38 | 7.51 |
| | SmoothQuant | 50% | 23.57 | 32.75 | 46.97 | 50.24 | 38.49 | 27.85 | 26.93 | 49.41 | 37.03 | – |
| | QuaRot | 50% | 39.22 | 62.66 | 69.55 | 72.74 | 41.48 | 68.89 | 38.20 | 61.74 | 56.81 | 9.25 |
| | OmniQuant | 50% | 34.91 | 54.34 | 61.66 | 63.39 | 39.51 | 57.04 | 33.47 | 57.89 | 50.28 | 11.47 |
| | SpinQuant | 50% | 46.76 | 72.54 | 73.79 | 72.61 | 42.66 | 71.41 | 45.60 | 62.53 | 60.99 | 8.36 |
| | **SparRotQuant** | **50%** | **49.02** | **71.69** | **75.56** | **75.45** | **44.13** | **73.18** | **47.33** | **64.34** | **62.59** | **7.94** |
| 4-4-4 | SmoothQuant | 0% | 24.17 | 33.78 | 49.66 | 49.84 | 39.56 | 28.45 | 29.13 | 49.11 | 37.96 | – |
| | QuaRot | 0% | 41.55 | 63.15 | 71.88 | 73.60 | 42.61 | 69.23 | 40.27 | 63.59 | 58.24 | 8.97 |
| | SpinQuant | 0% | 50.89 | 73.35 | 77.16 | 78.53 | 46.92 | 75.39 | 49.93 | 67.39 | 64.95 | 7.63 |
| | SmoothQuant | 50% | 22.53 | 29.15 | 44.45 | 46.96 | 37.90 | 24.38 | 26.87 | 44.99 | 34.65 | – |
| | QuaRot | 50% | 38.48 | 60.78 | 67.64 | 68.68 | 40.38 | 65.49 | 37.07 | 58.25 | 54.60 | 9.68 |
| | SpinQuant | 50% | 45.92 | 70.67 | 72.81 | 72.93 | 42.84 | 70.57 | 45.87 | 60.14 | 60.22 | 8.62 |
| | **SparRotQuant** | **50%** | **48.83** | **71.58** | **75.42** | **75.36** | **43.97** | **73.11** | **47.27** | **64.28** | **62.48** | **7.96** |

Table B.4: Zero-shot accuracy and WikiText2 perplexity of Qwen3-8B.

| Bits | Method | Sparsity | ARC-c | ARC-e | BoolQ | PIQA | SIQA | HellaS. | OBQA | WinoG. | Avg. | Wiki2 ↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16-16-16 | Full Precision | 0% | 62.11 | 81.38 | 85.65 | 73.75 | 52.35 | 61.45 | 55.13 | 73.62 | 68.18 | 6.99 |
| 16-16-16 | Full Precision | 50% | 58.81 | 79.56 | 81.99 | 73.72 | 51.75 | 59.62 | 53.20 | 71.74 | 66.30 | 7.28 |
| 4-4-16 | SmoothQuant | 0% | 29.98 | 32.36 | 48.07 | 47.88 | 37.99 | 27.19 | 30.80 | 52.59 | 38.36 | – |
| | QuaRot | 0% | 40.17 | 66.74 | 71.67 | 68.29 | 42.76 | 49.09 | 41.47 | 65.23 | 55.68 | 9.57 |
| | OmniQuant | 0% | 35.25 | 57.06 | 64.96 | 62.04 | 42.70 | 43.46 | 36.67 | 60.90 | 50.38 | 10.58 |
| | SpinQuant | 0% | 46.70 | 77.43 | 80.66 | 70.83 | 50.61 | 55.28 | 54.53 | 69.06 | 63.14 | 8.35 |
| | SmoothQuant | 50% | 25.05 | 27.89 | 44.01 | 44.46 | 33.96 | 19.68 | 27.73 | 49.74 | 34.07 | – |
| | QuaRot | 50% | 36.11 | 64.03 | 70.56 | 65.71 | 42.73 | 49.06 | 38.27 | 62.85 | 53.67 | 10.21 |
| | OmniQuant | 50% | 32.70 | 57.03 | 58.47 | 57.70 | 37.96 | 34.45 | 33.40 | 58.54 | 46.28 | 12.18 |
| | SpinQuant | 50% | 40.65 | 68.76 | 73.48 | 67.72 | 42.79 | 51.24 | 43.60 | 63.18 | 56.43 | 9.10 |
| | **SparRotQuant** | **50%** | **43.87** | **71.04** | **73.54** | **67.75** | **42.85** | **55.25** | **46.27** | **63.82** | **58.05** | **8.68** |
| 4-4-4 | SmoothQuant | 0% | 20.26 | 27.86 | 45.92 | 39.94 | 36.22 | 22.43 | 27.00 | 47.44 | 33.38 | – |
| | QuaRot | 0% | 36.14 | 57.34 | 67.95 | 63.83 | 38.05 | 45.36 | 38.07 | 61.93 | 51.08 | 9.88 |
| | SpinQuant | 0% | 46.33 | 68.93 | 73.51 | 68.26 | 42.82 | 53.22 | 49.80 | 63.59 | 58.31 | 8.63 |
| | SmoothQuant | 50% | 17.61 | 24.59 | 37.74 | 37.76 | 32.61 | 16.46 | 24.47 | 43.16 | 29.30 | – |
| | QuaRot | 50% | 32.88 | 57.09 | 64.99 | 63.80 | 33.99 | 45.33 | 35.13 | 58.14 | 48.92 | 10.29 |
| | SpinQuant | 50% | 40.20 | 66.77 | 70.56 | 65.06 | 38.02 | 49.03 | 43.27 | 59.41 | 54.04 | 9.48 |
| | **SparRotQuant** | **50%** | **43.61** | **67.81** | **70.59** | **64.10** | **38.08** | **50.97** | **43.33** | **61.74** | **55.03** | **9.04** |

Table B.6 shows that changing the calibration corpus affects absolute perplexity, but preserves the relative trend between the two methods. `SparRotQuant` consistently improves over sparse `SpinQuant` across all four models under both W4A4KV16 and W4A4KV4. This indicates that the benefit of sparsity-aware rotation calibration is not specific to the WikiText2 calibration distribution.

Table B.5: Comparison with joint sparse-quantization methods under W4A4KV16 with exact 2:4 structured sparsity. Results are reported as mean ± std over three independent runs. FP16 (2:4) denotes the 2:4-sparse model without quantization and is included as a non-quantized reference.

| Model | Method | ARC-c | ARC-e | BoolQ | PIQA | SIQA | HellaS. | OBQA | WinoG. | Avg.↑ | Wiki2↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L2-7B | FP16 (2:4) | 47.86 | 71.95 | 74.81 | 76.91 | 46.09 | 72.94 | 54.27 | 66.87 | 63.96 | 5.75 |
| | JSQ | 21.90±0.57 | 26.07±0.61 | 37.91±0.09 | 52.05±0.60 | 34.89±0.24 | 25.73±0.07 | 13.40±0.92 | 49.75±1.48 | 32.71±0.23 | 10.72±0.20 |
| | SLiM | 36.01±0.48 | **70.66**±0.55 | 71.11±0.52 | 74.25±0.23 | 43.47±0.69 | 49.11±0.20 | 26.27±1.29 | **67.72**±0.36 | 54.82±0.24 | 8.97±0.03 |
| | SparRotQuant | **45.58**±0.26 | 70.21±0.35 | **71.92**±0.11 | **75.12**±0.61 | **45.31**±0.34 | **70.82**±0.06 | **51.53**±0.69 | 64.62±0.33 | **61.89**±0.12 | **6.49**±0.04 |
| L2-13B | FP16 (2:4) | 52.05 | 75.63 | 78.32 | 79.16 | 49.44 | 77.91 | 55.73 | 70.88 | 67.39 | 5.35 |
| | JSQ | 30.26±0.55 | 62.39±0.66 | 71.45±0.30 | 69.88±0.19 | 40.45±0.16 | 42.77±0.14 | 24.00±0.20 | 64.14±0.66 | 50.67±0.29 | 12.44±0.29 |
| | SLiM | 38.62±0.94 | 72.10±0.32 | **80.50**±0.15 | 75.03±0.42 | 43.43±0.29 | 50.99±0.15 | 29.67±0.61 | **71.35**±0.28 | 57.71±0.10 | 7.52±0.04 |
| | SparRotQuant | **50.28**±0.21 | **75.28**±0.53 | 77.64±0.28 | **79.04**±0.46 | **48.77**±0.15 | **77.26**±0.18 | **50.13**±0.22 | 70.16±0.14 | **66.07**±0.19 | **5.73**±0.03 |

Table B.6: WikiText2 perplexity (↓) of exact 2:4 sparse models calibrated on WikiText2 training data (WT2) or C4.

| Bits | Method | Calib. | L2-7B | L2-13B | L3-8B | Q3-8B |
|---|---|---|---|---|---|---|
| 4-4-4 | SpinQuant | WT2 | 6.75 | 6.21 | 8.62 | 9.48 |
| | SparRotQuant | WT2 | 6.61 | 6.08 | 7.96 | 9.04 |
| | SpinQuant | C4 | 7.95 | 7.63 | 9.11 | 9.96 |
| | SparRotQuant | C4 | 7.69 | 7.27 | 8.34 | 9.46 |
| 4-4-16 | SpinQuant | WT2 | 6.62 | 5.96 | 8.36 | 9.10 |
| | SparRotQuant | WT2 | 6.49 | 5.73 | 7.94 | 8.68 |
| | SpinQuant | C4 | 7.03 | 7.14 | 8.68 | 9.47 |
| | SparRotQuant | C4 | 6.76 | 6.89 | 8.23 | 8.95 |

Table B.7: Hyperparameters used for SparRotQuant calibration.

| Hyperparameter | Value |
|---|---|
| Calibration samples | 800 |
| Sequence length | 2048 |
| Batch size | 8 |
| Rotation LR | 1.5 |
| Gate LR | 0.1 |
| Quantizer LR | $1 \times 10^{-3}$ |
| Recovery steps | 20 |
| $\lambda_p$ | 0.18 |
| PBGS separation temperature $T$ | 1.0 |
| HC $(\beta, \gamma, \zeta)$ | $(2/3, -0.1, 1.1)$ |

### B.4 Calibration hyperparameters

We provide the calibration hyperparameters used throughout the main `SparRotQuant` experiments in Table B.7. Unless otherwise specified, calibration uses 800 samples with a sequence length of 2048 and a batch size of 8, corresponding to 100 optimization steps per block. The rotation matrices, sparse gate parameters, and quantizers are optimized with separate learning rates to account for their different optimization dynamics. For sparse mask learning, we use a PBGS regularization weight $\lambda_p = 0.18$ and set the PBGS separation temperature $T$ to 1.0, with $(\beta, \gamma, \zeta) = (2/3, -0.1, 1.1)$. After exact 2:4 projection, we perform 20 recovery steps using the projected sparse structure. These settings are kept fixed across the main model and quantization configurations unless otherwise stated.

## C Supplemental inference evaluation

### C.1 Kernel-level efficiency

**Kernel and runtime setting.** All efficiency experiments use custom CUTLASS-based 2:4 sparse GEMM kernels on a single NVIDIA RTX 4090 GPU. For each four-weight group, the sparse representation stores two INT4 values, 4-bit position metadata, and one FP16 scaling factor per output row, resulting in 5.33× compression for sparsified layers. Since sparse `SpinQuant` and `SparRotQuant` share the same W4A4+2:4 deployment format, they exhibit comparable inference efficiency under the same sparse kernel. `SparRotQuant` therefore focuses on improving model quality while retaining the efficiency benefits of sparse low-bit execution.

**Sparse Tensor Core execution.** The kernels use NVIDIA sparse Tensor Core instructions with hardware metadata rather than dense masking. All deployed 2:4 patterns are validated against an integer reference implementation and produce bit-exact outputs. Because `m16n8k128.s4` only supports pair-granular sparsity, packed INT4 weights and activations are sign-extended to INT8 registers and executed through `m16n8k64.s8`. The numerical values remain within the signed

4-bit range, accumulate in INT32, and use fused per-row weight scaling and per-token activation scaling, preserving W4A4 semantics while retaining packed-INT4 storage.

Table C.1 reports the kernel latency underlying Figure 6 for representative linear projection shapes from LLaMA2-7B and LLaMA2-13B. We compare dense FP16 execution, `Marlin`-based sparse execution, dense 4-bit `SpinQuant`, and the sparse low-bit kernel used by `SparRotQuant` under the same matrix shapes. This experiment isolates the operator-level contribution of combining low-bit quantization with exact 2:4 sparsity before considering the remaining Transformer overhead. Across all evaluated shapes, `SparRotQuant` reduces kernel latency by 72.7–78.7% compared with dense FP16, 45.8–52.6% compared with `Marlin-2:4`, and 31.9–35.1% compared with dense `SpinQuant`. The consistent latency reduction over both sparse `Marlin` execution and dense 4-bit execution demonstrates the complementary kernel-level benefits of structured sparsity and low-bit quantization.

Table C.1: Prefill kernel latency (ms, $M = 4096$) and latency reduction of `SparRotQuant` over the evaluated baselines.

| Model | K/N | Latency (ms)↓ | | | | Latency Reduction vs. (%)↑ | | |
|---|---|---|---|---|---|---|---|---|
| | | FP16 | Marlin | SpinQuant | SparRotQuant | FP16 | Marlin | SpinQuant |
| L2-7B | 4096/4096 | 0.834 | 0.478 | 0.339 | 0.228 | 72.7 | 52.3 | 32.7 |
| | 4096/11008 | 2.257 | 0.973 | 0.812 | 0.527 | 76.7 | 45.8 | 35.1 |
| | 11008/4096 | 2.229 | 0.923 | 0.698 | 0.475 | 78.7 | 48.5 | 31.9 |
| L2-13B | 5120/5120 | 1.287 | 0.661 | 0.488 | 0.322 | 75.0 | 51.3 | 34.0 |
| | 5120/13824 | 3.603 | 1.656 | 1.318 | 0.886 | 75.4 | 46.5 | 32.8 |
| | 13824/5120 | 3.486 | 1.623 | 1.176 | 0.769 | 77.9 | 52.6 | 34.6 |

### C.2 End-to-end prefill efficiency

Table C.2 provides the full end-to-end prefill latency results over sequence lengths from 256 to 8192. All speedups are measured relative to the dense FP16 baseline under the same model and sequence length. FP16+S isolates the benefit of exact 2:4 sparsity without low-bit quantization, while dense `SpinQuant` isolates the benefit of 4-bit rotation-based quantization without sparsity.

`SparRotQuant` consistently achieves the lowest prefill latency across all evaluated sequence lengths on both LLaMA2-7B and LLaMA2-13B. The improvement over both FP16+S and dense `SpinQuant` shows that the complementary kernel-level gains observed in Table C.1 translate to end-to-end inference. The achievable end-to-end speedup is smaller than the isolated kernel speedup because attention, activation movement, and other non-sparsified Transformer components remain unchanged.

### C.3 Decode-shaped kernel efficiency

To examine whether the kernel-level benefit extends beyond prefill-shaped workloads, we further evaluate decode-shaped GEMMs on LLaMA2-7B with $T = 32$ concurrently processed token positions. We follow the same projection convention as Section C.1, where $K/N$ denotes the input and output dimensions of each linear projection. Sparse W4A4 uses our exact 2:4 sparse kernel, while Dense W4A4 uses the dense INT4 GEMM implementation evaluated with `SpinQuant` in Table C.1. Dense FP16 uses the same CUTLASS-based FP16 GEMM baseline as in our kernel-level evaluation.

We use a cold-cache protocol for all configurations. Each timed invocation is preceded by writing a 256 MB flush buffer, followed by device synchronization. We perform 20 warmup iterations before measurement and report the median latency over 100 timed runs.

As shown in Table C.3, Sparse W4A4 achieves 4.74× to 5.15× speedup over dense FP16 across the evaluated projection shapes. Compared with dense W4A4, exact 2:4 sparsity provides an additional 1.26× to 1.36× kernel-level acceleration. These results show that structured sparsity provides additional efficiency beyond low-bit quantization under decode-shaped workloads.

Table C.2: Prefill latency and speedup under different sequence lengths. Speedup is computed relative to the dense FP16 baseline. FP16+S denotes FP16 inference with 50% structured sparsity.

| Model | SeqLen | Prefill Latency (ms) ↓ | | | | Speedup ↑ | | |
|---|---|---|---|---|---|---|---|---|
| | | FP16 | FP16+S | SpinQuant | SparRotQuant | FP16+S | SpinQuant | SparRotQuant |
| L2-7B | 256 | 8.694 | 5.609 | 4.600 | 3.520 | 1.55× | 1.89× | 2.47× |
| | 512 | 15.947 | 9.905 | 8.136 | 6.276 | 1.61× | 1.96× | 2.54× |
| | 1024 | 30.514 | 18.160 | 14.882 | 11.446 | 1.68× | 2.05× | 2.67× |
| | 2048 | 58.618 | 33.883 | 27.650 | 21.602 | 1.73× | 2.12× | 2.71× |
| | 4096 | 118.968 | 67.593 | 57.194 | 46.878 | 1.76× | 2.08× | 2.54× |
| | 8192 | 261.802 | 154.001 | 130.901 | 108.182 | 1.70× | 2.00× | 2.42× |
| L2-13B | 256 | 12.479 | 7.517 | 5.942 | 4.395 | 1.66× | 2.10× | 2.84× |
| | 512 | 22.894 | 13.309 | 10.453 | 7.684 | 1.72× | 2.19× | 2.98× |
| | 1024 | 44.675 | 24.958 | 19.594 | 14.184 | 1.79× | 2.28× | 3.15× |
| | 2048 | 88.522 | 48.110 | 37.669 | 27.094 | 1.84× | 2.35× | 3.27× |
| | 4096 | 210.989 | 112.228 | 91.337 | 69.180 | 1.88× | 2.31× | 3.05× |
| | 8192 | 388.162 | 213.276 | 174.848 | 139.878 | 1.82× | 2.22× | 2.78× |

Table C.3: Kernel latency for decode-shaped workloads.

| $K/N$ | Sparse W4A4 2:4 (ms) | Dense W4A4 (ms) | Dense FP16 (ms) | vs. FP16 | vs. W4A4 |
|---|---|---|---|---|---|
| 4096/4096 | 0.0095 | 0.0129 | 0.0450 | 4.74× | 1.36× |
| 4096/11008 | 0.0216 | 0.0273 | 0.1102 | 5.10× | 1.26× |
| 11008/4096 | 0.0214 | 0.0278 | 0.1103 | 5.15× | 1.30× |

### C.4 Runtime memory

Table C.4 reports per-layer runtime memory usage during prefill over sequence lengths 256–8192; the "Mem. Saving" in Table 1 is the corresponding average saving relative to dense FP16. Dense `SpinQuant` reduces the weight footprint through 4-bit storage, while `SparRotQuant` further compresses the selected projection weights using exact 2:4 sparse storage. The relative memory saving is largest at shorter sequence lengths, where model weights constitute a larger fraction of the runtime footprint. As sequence length increases, activations, attention buffers, KV-cache-related states, sparse metadata, and workspace account for a larger fraction of memory. Consequently, the measured saving remains below the ideal weight-only compression ratio, while `SparRotQuant` consistently retains an additional memory advantage over dense low-bit inference.

Table C.4: Prefill memory usage and memory saving under different sequence lengths. Memory saving is computed relative to the dense FP16 baseline. FP16+S denotes FP16 inference with 50% structured sparsity.

| Model | SeqLen | Memory Usage (GB) ↓ | | | | Memory Saving ↑ | | |
|---|---|---|---|---|---|---|---|---|
| | | FP16 | FP16+S | SpinQuant | SparRotQuant | FP16+S | SpinQuant | SparRotQuant |
| L2-7B | 256 | 0.442 | 0.266 | 0.157 | 0.104 | 1.66× | 2.82× | 4.27× |
| | 512 | 0.469 | 0.288 | 0.174 | 0.116 | 1.63× | 2.69× | 4.05× |
| | 1024 | 0.523 | 0.329 | 0.207 | 0.139 | 1.59× | 2.53× | 3.77× |
| | 2048 | 0.629 | 0.408 | 0.268 | 0.179 | 1.54× | 2.35× | 3.52× |
| | 4096 | 0.827 | 0.559 | 0.390 | 0.258 | 1.48× | 2.12× | 3.21× |
| | 8192 | 1.221 | 0.878 | 0.636 | 0.439 | 1.39× | 1.92× | 2.78× |
| L2-13B | 256 | 0.681 | 0.396 | 0.236 | 0.154 | 1.72× | 2.88× | 4.42× |
| | 512 | 0.716 | 0.424 | 0.263 | 0.172 | 1.69× | 2.72× | 4.16× |
| | 1024 | 0.784 | 0.475 | 0.307 | 0.205 | 1.65× | 2.55× | 3.82× |
| | 2048 | 0.913 | 0.571 | 0.385 | 0.257 | 1.60× | 2.37× | 3.55× |
| | 4096 | 1.168 | 0.763 | 0.546 | 0.363 | 1.53× | 2.14× | 3.22× |
| | 8192 | 1.672 | 1.169 | 0.853 | 0.577 | 1.43× | 1.96× | 2.90× |

# D Details of ablation analysis

## D.1 Rotation placement

Table D.1 provides the task-level results for the rotation-placement ablation summarized in Section 5.3. The experiment compares sparsity-aware optimization on $R_{in}$, on $R_{out}$, and jointly on both rotation spaces under the same W4A4KV16 and 50% exact 2:4 setting. Using only $R_{in}$ or $R_{out}$ yields similar model quality, while $R_{in}$ gives a smaller runtime memory footprint because sparsity covers a larger fraction of the transformed projection weights. Joint optimization over $R_{in}$ and $R_{out}$ substantially degrades accuracy and perplexity, consistent with the coupled perturbation terms identified in Section 3. These results support the use of input-side sparsification in the final design. The Avg. column is computed over the five tasks reported in Table D.1: ARC-e, PIQA, SIQA, HellaSwag, and WinoGrande.

Table D.1: Comparison of sparsity-aware rotation placement on LLaMA2-7B under W4A4KV16 with 50% structured sparsity.

| Method | Mem.(GB) | ARC-e | PIQA | SIQA | HellaS. | WinoG. | Avg. | Wiki2 ↓ |
|---|---|---|---|---|---|---|---|---|
| FP16 | 13.68 | 73.19 | 78.12 | 46.63 | 75.27 | 69.48 | 68.54 | 5.34 |
| FP16 + 50% Sparsity | 7.72 | 71.95 | 76.91 | 46.09 | 72.94 | 66.87 | 66.95 | 5.75 |
| $R_{in}$-only | 3.39 | 70.21 | 75.12 | 45.31 | 70.82 | 64.62 | 65.22 | 6.49 |
| $R_{out}$-only | 4.17 | 70.68 | 74.81 | 45.02 | 70.27 | 65.08 | 65.17 | 6.52 |
| $R_{in} + R_{out}$ | 2.91 | 64.17 | 71.36 | 40.71 | 65.58 | 59.74 | 60.31 | 7.14 |

## D.2 Efficiency contribution of quantization and sparsity

Figure D.1 decomposes the efficiency improvement into pruning-only and quantization-only variants. The pruning-only variant (Variation1) benefits from reduced arithmetic through the regular 2:4 pattern, while the dense 4-bit variant (Variation2) primarily reduces weight storage and low-bit computation. `SparRotQuant` improves both prefill speedup and memory saving by combining exact 2:4 sparsity with the low-bit rotated representation. The consistent shift toward the upper-right region for both LLaMA2-7B and LLaMA2-13B confirms that the two compression mechanisms remain complementary at the end-to-end level.

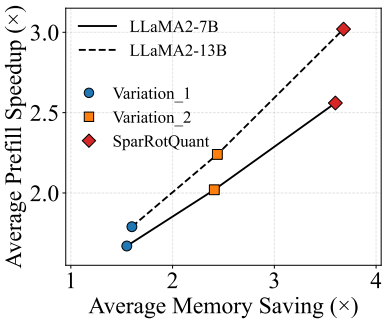



Figure D.1: Efficiency trade-off.

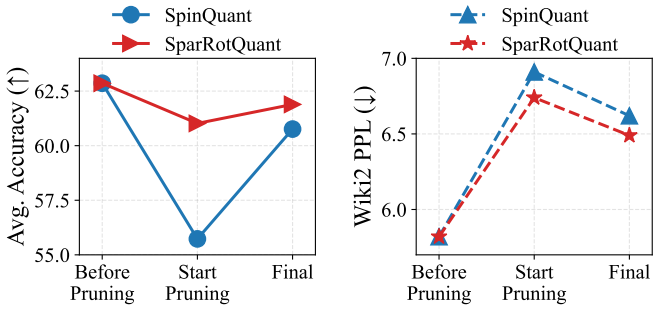



Figure D.2: Sparsification stages.

## D.3 Sparsification-stage analysis

Figure D.2 traces `SpinQuant` and `SparRotQuant` across three stages: before pruning, immediately after exact 2:4 projection, and after recovery. The "Before Pruning" point corresponds to the common dense rotated checkpoint used only for initialization. Starting from this checkpoint, `SpinQuant` is directly subjected to exact 2:4 projection, whereas `SparRotQuant` continues calibration by jointly refining the rotations, gate logits, and quantization parameters under sparsity-aware mask learning before exact projection.

Both methods start from dense low-bit rotated models, but applying the exact mask causes a substantially larger accuracy drop and perplexity increase for SpinQuant. In contrast, SparRotQuant undergoes a smaller transition because the target sparse structure has already influenced the rotated representation during calibration. During recovery, the learned rotations and exact 2:4 masks remain fixed, and the same recovery protocol is applied to both methods. The remaining advantage of SparRotQuant therefore indicates that its gain is primarily established before recovery by reducing the mismatch between rotation calibration and hard sparse projection.

### D.4 Validation of PBGS-guided mask formation

Figure 7 directly examines whether rotation weakens 2:4 pattern separability and whether PBGS restores it. We evaluate all 2:4 groups in $W_q$, $W_k$, $W_v$, $W_{\mathrm{up}}$, and $W_{\mathrm{gt}}$ of LLaMA2-7B under W4A4KV16. For each 4-channel group $g$, we enumerate the six possible 2:4 patterns and measure the reconstruction cost of pruning pattern $k$ as

$$c_k = w_{P_k}^{\top} G_{g,P_k P_k} w_{P_k}, \qquad G_g = X_g^{\top} X_g, \tag{D.1}$$

where $P_k$ denotes the two pruned positions. After sorting the six costs as $c_{(1)} \le c_{(2)} \le \cdots$, we define the normalized pattern margin

$$\mu_g = \frac{c_{(2)} - c_{(1)}}{c_{(2)} + c_{(1)} + \epsilon}. \tag{D.2}$$

A larger $\mu_g$ indicates that the preferred 2:4 pattern is more clearly separated from its closest alternative. To measure support stability independently, we recompute the minimum-cost pattern using a disjoint calibration subset and report the fraction of groups whose selected pattern changes.

For the *No PBGS* variant, gate logits are optimized only by $\mathcal{L}_q$, and the final 2:4 mask retains the two largest $p_i$ values in each group. Figure 7 (a) shows that rotation shifts the margin distribution toward smaller values; joint calibration alone does not recover this structure, while full SparRotQuant shifts it back toward larger margins. The same trend appears in Figure 7 (b): the mask flip rate rises from $5.6\%$ for the original weights to $16.8\%$ after rotation and $24.7\%$ without PBGS, whereas full SparRotQuant reduces it to $10.2\%$.

These results directly validate the central motivation of SparRotQuant: rotation degrades local sparse-pattern separability, and PBGS reshapes the rotated representation toward a more decisive and reproducible 2:4 support before hard projection. Rather than treating PBGS as a uniquely optimal separation objective, we design it as a projection-aligned surrogate that explicitly promotes separation between the two entries retained by exact 2:4 projection and the two entries removed. Its saliency $v_i = p_i h_i |\bar{w}_i|^2$ further couples retention probability with sensitivity and rotated weight magnitude, while the component ablations in Table 2 show that each signal alone is insufficient to recover the full performance. Together, the component ablations and mask-stability analysis support PBGS as a structured objective tailored to stable 2:4 projection rather than an arbitrary auxiliary regularizer.

### D.5 Calibration cost and budget-matched comparison

SparRotQuant introduces additional offline calibration cost relative to SpinQuant. Starting from the shared dense rotated checkpoint, the joint mask-learning phase adds element-wise gate optimization, diagonal Hessian statistics, and PBGS computation, without full-Hessian construction. As shown in Table D.2, this increases total calibration time by 27–33%. The cost is incurred once, offline. Since the deployed model uses the same W4A4 and exact 2:4 format as sparse SpinQuant, no inference-time latency or storage overhead is introduced.

To examine whether the accuracy gain stems from this additional budget rather than from sparsity-aware mask learning, we construct a budget-matched baseline on LLaMA2-7B under W4A4KV16. Starting from the same projected 2:4 mask, sparse SpinQuant continues to calibrate its rotations and quantization parameters, with the mask fixed in the rotated coordinate system, and then applies the same recovery protocol. Its total calibration time is matched to that of SparRotQuant (2.8 h). As shown in Table D.3, the extra calibration improves sparse SpinQuant only marginally, from 60.76 to 61.07 Avg. and from 6.62 to 6.59 Wiki2 PPL. Under the same budget, SparRotQuant reaches 61.89 and 6.49. Across multi-runs, the standard deviation of Avg. is at most 0.18 for these

Table D.2: Total calibration time hours (h) of SpinQuant and SparRotQuant. Additional time and relative overhead are computed from unrounded measurements.

| Model | SpinQuant | SparRotQuant | Additional | Relative Overhead |
|---|---|---|---|---|
| LLaMA2-7B | 2.2 | 2.8 | 0.6 | 27% |
| LLaMA2-13B | 4.5 | 5.7 | 1.2 | 27% |
| LLaMA3-8B | 2.6 | 3.4 | 0.8 | 31% |
| Qwen3-8B | 2.3 | 3.0 | 0.8 | 33% |

Table D.3: Budget-matched comparison on LLaMA2-7B (W4A4KV16, exact 2:4). Results are averaged over multiple runs, except No PBGS, which is taken from Table 2.

| Method | Calib. time (h) | Avg. ↑ | Wiki2 ↓ |
|---|---|---|---|
| Sparse SpinQuant | 2.2 | 60.76 | 6.62 |
| Sparse SpinQuant + extra calib. | 2.8 | 61.07 | 6.59 |
| No PBGS (joint calib.) | 2.8 | 55.73 | 6.97 |
| SparRotQuant | 2.8 | **61.89** | **6.49** |

three methods. The $+0.31$ gain from extra calibration therefore lies within run-to-run variation, whereas the remaining margin of 0.82 in favor of `SparRotQuant` is well beyond it, and its perplexity improvement is consistent across runs.

In addition, the No-PBGS variant in Table 2 undergoes the same joint calibration phase with a comparable budget (2.8 h), yet reaches only 55.73 Avg. and 6.97 Wiki2 PPL. Additional calibration time is therefore neither sufficient for, nor the primary source of, the improvement. With a fixed support, further optimization can only adapt the rotation to a predetermined mask. `SparRotQuant` instead shapes the rotated representation and the sparse support jointly under PBGS guidance before exact projection.