# SPARSESLIDERQUANT: SPARSE-QUANTIZED LLM WITH CROSS-BLOCK ERROR COORDINATION

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

## ABSTRACT

Sparse quantization is a promising compression paradigm for Transformer decoders, as it simultaneously reduces numerical precision and arithmetic density. However, aggressive configurations such as W4A4 quantization combined with 2:4 structured sparsity reveal a fundamental tension between local compression and cross-block consistency. Existing methods typically perform sparsification and quantization independently or sequentially within local scopes, making it difficult to effectively coordinate the two sources of error and control their cumulative propagation across blocks. To address this, we propose SparseSliderQuant, a unified rewrite framework that reformulates independent local sparsification and quantization as coordinated cross-block error optimization. Specifically, SparseSliderQuant introduces sliding group alternation, which iteratively updates sparse masks and quantization scales within overlapping block windows, allowing the two compression processes to adapt to each other. Building on this procedure, dual-source mismatch calibration separately accumulates pruning and quantization residuals and feeds them back into unpruned weights, enabling robust error compensation. In addition, cross-block gradient anchoring jointly updates masks and scales according to the output drift of the subsequent block, aligning local reconstruction with downstream behavior throughout the decoder. SparseSliderQuant requires no auxiliary modules and incurs no additional inference-time overhead. Experiments on LLaMA and Qwen demonstrate that, under W4A4 + 2:4 structured sparsity, SparseSliderQuant consistently outperforms sequential and joint methods across language understanding and reasoning benchmarks while achieving a favorable accuracy–efficiency trade-off. Code will be released upon acceptance at `https://github.com/nntk-wb/SparseSliderQuant`.

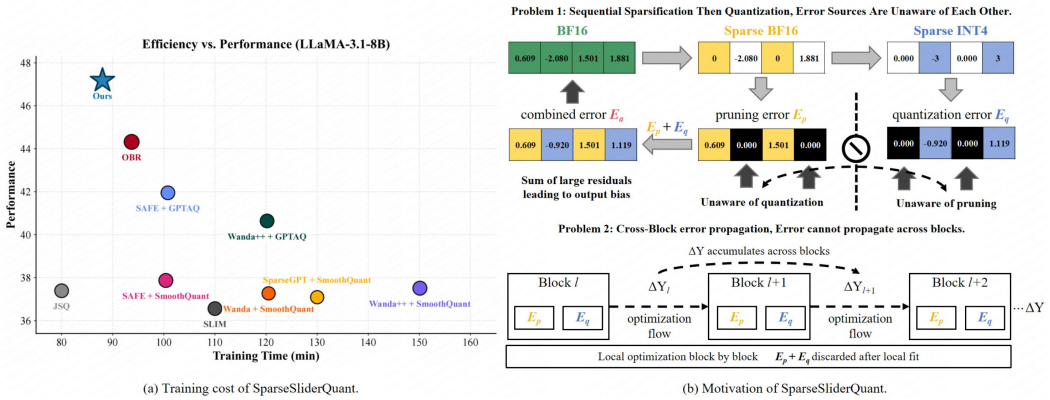
## 1 INTRODUCTION

Transformer decoders dominate modern large language models (LLMs), achieving strong performance in generation, reasoning, and instruction following. Their rapid growth in size incurs substantial memory, computation, and deployment costs, making compression essential for efficient inference. Quantization and structured sparsity reduce these costs along complementary dimensions: quantization lowers bit width for weights and activations, whereas structured sparsity reduces arithmetic density via hardware-friendly patterns. Combining W4A4 with 2:4 structured sparsity is appealing as it reduces both memory traffic and computation. However, joint compression is not simple superposition of independent operations. Pruning changes the weight tensor that quantization processes, while quantization perturbs retained nonzero values. Resulting errors are coupled and propagate through successive blocks, turning local errors into downstream drift.

Considerable progress has been made in quantization and sparsification when they are studied separately. Post-training quantization methods such as GPTQ Frantar et al. (2022), AWQ Lin et al. (2024), and SmoothQuant Xiao et al. (2023) mitigate low-bit degradation through Hessian-aware compensation, activation-aware scaling, and outlier migration, respectively. Rotation-based approaches, including QuaRot Ashkboos et al. (2024), SpinQuant Liu et al. (2025), and FlatQuant Sun et al. (2024b), further improve quantization robustness by reshaping weight and activation distributions. In parallel, pruning methods such as SparseGPT Frantar & Alistarh (2023), Wanda Sun et al. (2024a), Wanda++ Yang et al. (2025), and SAFE Lee et al. (2025) improve sparse reconstruction using second-order information, activation-aware saliency, regional gradients, or flatness-aware



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.




(a) Training cost of SparseSliderQuant.

(b) Motivation of SparseSliderQuant.

Figure 1: Training cost and motivation. (a) On LLaMA-3.1-8B (W4A4, 2:4), SparseSliderQuant outperforms OBR in accuracy (47.18% vs. 44.32%) and speed (88.0 vs. 93.7 min). (b) Motivation: coupled pruning and quantization residuals are discarded after local fit, accumulating across blocks.

criteria. Although effective along their respective compression dimensions, these methods do not explicitly model the interaction between structured pruning and low-bit quantization. Recent sparse-quantization methods have begun to address this interaction. JSQ Guo et al. (2024) introduces a sparsity-aware reconstruction metric to mitigate conflicts between pruning and quantization. SLIM Mozaffari et al. (2024) organizes quantization, pruning, and error compensation into a sequential pipeline. OBR Guo et al. (2025) uses Hessian-based weight restoration to compensate for joint sparsification and quantization errors. Despite these advances, most methods still compress layers or blocks using local statistics and reconstruction objectives. Consequently, the effect of a compressed block on subsequent blocks is captured only indirectly, leaving a discrepancy between the local objective and cross-block behavior that determines inference quality.

We attribute this discrepancy to two limitations, as illustrated in Figure 1. First, pruning and quantization produce residuals from different sources that remain coupled: pruning removes weight entries while quantization rounds retained values. Existing methods discard these residuals after local reconstruction, leaving subsequent blocks to process distorted activations without recovery. Moreover, a small local error does not guarantee cross-block consistency—a slight output deviation shifts the next block's input distribution, causing errors to accumulate with depth. Second, static optimization separates gradient signals for two compression modes: permanently assigning groups to either mask or scale optimization prevents each from observing the other's distortion, while simultaneous updates cause unstable interactions. Effective joint compression therefore requires controlled alternating optimization and explicit compensation for errors propagating across blocks.

Motivated by these observations, we propose SparseSliderQuant, which reformulates joint sparsification and quantization from independent local compression into coordinated cross-block error optimization. Within overlapping block windows, sliding group alternation updates sparse masks and quantization scales in turn, exposing each group to both compression modes without simultaneous-update instability. Dual-source mismatch calibration then decomposes compression error into pruning and quantization residuals, accumulates their coupled mismatch in a buffer, and feeds it back into unpruned weights. Cross-block gradient anchoring constrains subsequent block output drift and backpropagates this to both masks and scales. Together, they align local reconstruction with downstream behavior without auxiliary modules. Our contributions are summarized as follows:

- We propose SparseSliderQuant, a unified rewrite framework that reformulates joint sparsification and quantization from independent local compression into effective coordinated cross-block error optimization for aggressive low-bit structured-sparse LLM compression.
- We introduce sliding group alternation and dual-source mismatch calibration, alternately optimizing sparse masks and quantization scales while accumulating pruning and quantization residuals and feeding them back into unpruned weights for robust error compensation.
- We introduce cross-block gradient anchoring, which updates sparse masks and quantization scales according to next-block output drift, aligning local reconstruction objectives with consistent downstream behavior throughout compressed Transformer decoders.
- We evaluate SparseSliderQuant on LLaMA and Qwen models under W4A4 quantization with 2:4 structured sparsity, demonstrating improvements over sequential and joint sparse-quant methods across benchmarks while maintaining an accuracy–efficiency trade-off.

# 2 Related Work

## 2.1 Post-Training Quantization

Post-training quantization (PTQ) reduces LLM costs by converting weights and activations to low-bit representations without retraining. LLM.int8() Dettmers et al. (2022) isolates activation outliers for higher-precision computation. GPTQ Frantar et al. (2022) extends second-order weight reconstruction to large-scale models, while AWQ Lin et al. (2024) protects salient weight channels through activation-aware scaling. SmoothQuant Xiao et al. (2023) migrates quantization difficulty from activations to weights via per-channel transformation, and OmniQuant Shao et al. (2024) learns equivalent transformations and clipping parameters. Subsequent methods improve ultra-low-bit quantization through finer-grained error modeling: ZeroQuant-V2 Yao et al. (2022) introduces low-rank compensation, SpQR Dettmers et al. (2023) separates sensitive weight outliers, QuIP Chee et al. (2023) uses incoherence processing and adaptive rounding, and CBQ Ding et al. (2025) models cross-block dependencies to reduce error accumulation. Recent methods transform weight and activation distributions before quantization: QuaRot Ashkboos et al. (2024) applies orthogonal rotations to suppress outliers, SpinQuant Liu et al. (2025) learns rotation matrices from calibration data, FlatQuant Sun et al. (2024b) combines affine transformations with block reconstruction, and RRS Yi et al. (2025) introduces rotated runtime smoothing for training-free INT4 inference. Although these methods substantially improve low-bit quantization, they do not explicitly optimize structured sparsity or model its interaction with quantization error.

## 2.2 Post-Training Sparsification

Post-training sparsification removes redundant weights from pretrained LLMs without expensive full-model retraining. SparseGPT Frantar & Alistarh (2023) uses approximate second-order information to select pruned weights and compensate the resulting reconstruction error. Wanda Sun et al. (2024a) introduces a lightweight importance score based on weight magnitude and input-activation norms. DSnoT Zhang et al. (2024) dynamically searches sparse masks by pruning and regrowing weights according to reconstruction-induced fluctuations, improving one-shot sparsification without retraining. OWL Yin et al. (2025) identifies layer-wise outlier distributions and allocates non-uniform sparsity according to the proportion of salient weights. More recent approaches improve pruning through gradient information, mask search, and layer-wise allocation. Wanda++ Yang et al. (2025) incorporates regional gradients into pruning scores and performs block-level regional optimization. Pruner-Zero Dong et al. (2024) automatically discovers pruning metrics through symbolic search rather than relying on manually designed importance criteria. SAFE Lee et al. (2025) seeks sparse solutions in flatter regions of the loss landscape to improve pruning robustness. LSA Yang et al. (2026) assigns non-uniform layer sparsities by minimizing estimated linear reconstruction error under a global sparsity constraint. These methods improve unstructured or semi-structured pruning, but their criteria are generally derived without accounting for subsequent low-bit quantization or the coupled errors produced by joint compression.

## 2.3 Joint Sparsification and Quantization

Joint sparsification and quantization combines two complementary compression dimensions but must account for their interaction. Pruning changes the tensor presented to the quantizer, while quantization perturbs the retained sparse values; optimizing the two operations independently can therefore compound their errors. JSQ Guo et al. (2024) introduces the Salience with Activation Range metric to make sparsification aware of the activation range encountered by subsequent quantization, together with activation editing for outlier suppression. SLIM Mozaffari et al. (2024) integrates sparsification, quantization, and low-rank approximation into a one-shot compression pipeline. OBR Guo et al. (2025) formulates joint compression through Hessian-based restoration and compensates sparse-quant perturbations during weight reconstruction. Despite this progress, existing joint methods primarily optimize compression decisions within individual layers or blocks. They do not explicitly retain sparse and quantization residuals for subsequent correction, nor do they directly constrain a block according to the output drift induced in the following block. SparseSliderQuant addresses these limitations by coordinating mask and scale optimization within overlapping block windows, feeding coupled residuals back to unpruned weights, and propagating next-block drift to current-block compression parameters.

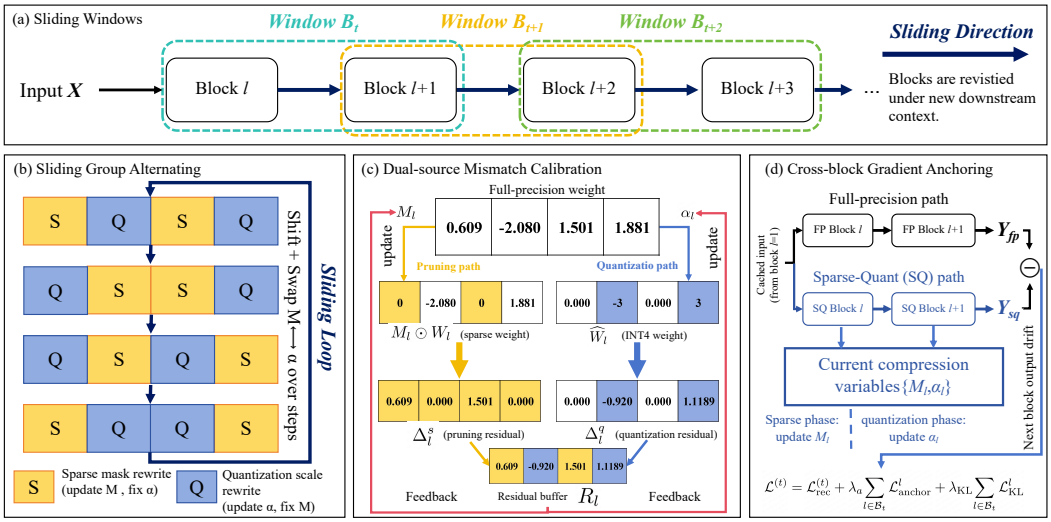



Figure 2: Overview of SparseSliderQuant. Within overlapping Transformer-block windows, sliding group alternation updates sparse masks and quantization scales in turn. Dual-source mismatch calibration accumulates pruning and quantization residuals and feeds them back to unpruned weights, while cross-block gradient anchoring uses next-block output drift to guide compression decisions.

# 3 METHOD

In this section, we describe the formulation of our SparseSliderQuant and detail its implementation.

## 3.1 BASIC CONCEPT: FIXED-SIZE SLIDING SPARSE QUANTIZATION

Given a pre-trained LLM with $L$ Transformer blocks, we perform weight–activation quantization: the low-precision input $\widehat{X}$ is quantized from $X$ beforehand, and the sparse–quantized weight is:

$$\widehat{W}_l = Q_4(M_l \odot W_l;\, \alpha_l), \tag{1}$$

where $M_l$ is the N:M sparse mask, $\alpha_l$ is the quantization scale, $Q_4(\cdot)$ denotes 4-bit symmetric uniform quantization, and $\odot$ is the Hadamard product. A fixed-size sliding window $\mathcal{B}_t = \{b_t, \ldots, b_t + s - 1\}$ moves along the block direction, where $s$ denotes the window size and $b_{t+1} = b_t + r$ advances the window by stride $r$ at each step, yielding an overlap of $s - r$ between consecutive windows. Let $W = \{W_{b_t}, \ldots, W_{b_t+s-1}\}$ be the pre-trained weight matrices within the current window and $X$ the input activation corresponding to the calibration samples of $c$ .

$$\min_{\widehat{W}_l} \left\| F_l(W_l, X_l) - F_l(\widehat{W}_l, \widehat{X}_l) \right\|_2^2. \tag{2}$$

This formulation overlooks two dependencies: (i) quantization is applied to the pruned tensor, coupling their errors; (ii) a small local discrepancy alters the input distribution of the next block and accumulates with depth. Joint methods update $M_l$ and $\alpha_l$ simultaneously within each block:

$$\min_{M_l,\, \alpha_l} \left\| F_l(W_l, X_l) - F_l\big(Q_4(M_l \odot W_l; \alpha_l),\, \widehat{X}_l\big) \right\|_2^2. \tag{3}$$

When $s$ equals one block and $r = s$ (no overlap), Eq. equation 2 and Eq. equation 3 reduce to layer-wise and block-wise frameworks popularly used in existing PTQ works. That is, they are special cases of fixed-size sliding sparse quantization. A larger window size increases memory cost but enjoys better accuracy than layer-wise methods, as overlapping windows allow compression errors near one boundary to remain visible when the neighboring window is optimized.

## 3.2 SLIDING GROUP ALTERNATING REWRITE

The analysis in Section 3.1 identifies a fundamental tension: static assignment isolates gradient signals, while simultaneous updates destabilize optimization because discrete mask changes continuously alter the weight tensor on which quantization scales depend. We resolve this by partitioning

weights within each block into four groups and alternating their compression roles across rewrite steps, ensuring every group is exposed to both modes over successive steps while only one variable is active per group, thereby avoiding signal isolation and update instability during calibration.

Within each block, weights form four groups $\{g_1, g_2, g_3, g_4\}$ of size $g$. At rewrite step $k$, the assignment $\pi^{(k)}$ maps each group to either sparse ($S$) or quantization ($Q$) mode:

$$\pi^{(k)} : \{g_1, g_2, g_3, g_4\} \to \{S, Q\}, \quad \mathcal{G}_k^S = \big\{g : \pi^{(k)}(g) = S\big\}, \quad \mathcal{G}_k^Q = \big\{g : \pi^{(k)}(g) = Q\big\}. \tag{4}$$

The assignments follow a four-step cycle in which two adjacent groups swap roles at each step while the other two remain fixed:

$$\big(\pi^{(0)}, \pi^{(1)}, \pi^{(2)}, \pi^{(3)}\big) = \big(SQSQ,\ QSSQ,\ QSQS,\ SQQS\big), \tag{5}$$

after which $\pi^{(4)} = \pi^{(0)}$ and the cycle repeats. This ensures that each group alternates between $S$ and $Q$ every two steps—groups $g_1, g_2$ swap on odd steps, groups $g_3, g_4$ on even steps—so every group experiences both compression modes without two groups in the same block being simultaneously reassigned. During the sparse phase, only mask variables in $\mathcal{G}_k^S$ are updated with quantization fixed:

$$M_l^{(k+1)} = M_l^{(k)} - \eta_s \, \nabla_{M_l} \mathcal{L}^{(t)}\Big|_{\alpha_l = \bar{\alpha}_l}, \quad g \in \mathcal{G}_k^S. \tag{6}$$

During the quantization phase, only scale variables in $\mathcal{G}_k^Q$ adapt on the fixed sparse support:

$$\alpha_l^{(k+1)} = \alpha_l^{(k)} - \eta_q \, \nabla_{\alpha_l} \mathcal{L}^{(t)}\Big|_{M_l = \bar{M}_l}, \quad g \in \mathcal{G}_k^Q. \tag{7}$$

Let $F_{\mathcal{B}_t}$ denote the sequential composition of blocks in $\mathcal{B}_t$. From the same cached input $X_{b_t}$, the full-precision output $Y_{\mathcal{B}_t}^{\mathrm{fp}}$ and compressed output $Y_{\mathcal{B}_t}^{\mathrm{sq}}$ define the window reconstruction loss:

$$\mathcal{L}_{\mathrm{rec}}^{(t)} = \left\| Y_{\mathcal{B}_t}^{\mathrm{sq}} - Y_{\mathcal{B}_t}^{\mathrm{fp}} \right\|_2^2. \tag{8}$$

Unlike isolated block loss, Eq. equation 8 exposes each rewrite to all downstream blocks within the window. A block near the end of one window reappears near the beginning of the next and is reoptimized under a different downstream context, coordinating adjacent blocks.

### 3.3 Dual-Source Mismatch Calibration

Sliding alternation coordinates mask and scale optimization but does not itself recover the information discarded by either operation. As noted in Section 3.1, pruning and quantization produce residuals from different sources that remain coupled: pruning removes entries while quantization rounds retained values. Existing methods discard these residuals after local reconstruction, leaving subsequent blocks to process distorted activations. We address this by decomposing the mismatch by source, accumulating it across rewrite steps, and feeding it back to surviving weights.

The pruning and quantization residuals are:

$$\Delta_l^s = (1 - M_l) \odot W_l, \qquad \Delta_l^q = M_l \odot W_l - Q_4(M_l \odot W_l; \alpha_l). \tag{9}$$

Their sum exactly recovers the total compression error:

$$\Delta_l^s + \Delta_l^q = W_l - \widehat{W}_l. \tag{10}$$

Although separated for calibration, the two terms are not independent—quantization acts on the sparsified tensor. At rewrite step $k$, both residuals accumulate in a block-specific buffer:

$$R_l^{(k)} = \beta R_l^{(k-1)} + \Delta_l^{s,(k)} + \Delta_l^{q,(k)}, \tag{11}$$

where $\beta \in [0, 1)$ controls the history contribution, preserving persistent mismatch across successive rewrites instead of discarding it after each local update.

The accumulated residual cannot be written back indiscriminately, as modifying masked entries violates N:M pattern. We restrict compensation to surviving support $\Omega_l^{\mathrm{free}} = \{j : (M_l)_j = 1\}$:

$$W_l^{(k+1)} = W_l^{(k)} + \eta_R \, \Pi_{\Omega_l^{\mathrm{free}}}\Big(R_l^{(k)}\Big), \tag{12}$$

where $\Pi_{\Omega_l^{\mathrm{free}}}$ projects onto unpruned entries and $\eta_R$ controls feedback strength. The projection preserves the target sparsity pattern, while the momentum term suppresses abrupt corrections. Unpruned weights gradually absorb persistent mismatch from both compression sources.

### 3.4 CROSS-BLOCK GRADIENT ANCHORING

Dual-source calibration compensates weight-space mismatch within the current block, but even a small residual may alter the next block's input distribution—the root cause of cross-block error accumulation identified in Section 3.1. We therefore use the next-block response as an explicit optimization anchor, constraining each compression decision by its downstream effect.

Given the same cached input, $Y_{l+1}^{\mathrm{fp}}$ and $Y_{l+1}^{\mathrm{sq}}$ denote the outputs of block $l+1$ along the full-precision and sparse–quantized paths. The normalized anchor loss is:

$$\mathcal{L}_{\mathrm{anchor}}^{l} = \frac{\left\| Y_{l+1}^{\mathrm{sq}} - Y_{l+1}^{\mathrm{fp}} \right\|_2^2}{\left\| Y_{l+1}^{\mathrm{fp}} \right\|_2^2 + \epsilon}, \tag{13}$$

where $\epsilon$ prevents numerical instability. The anchor provides a downstream signal for both variables: during a sparse rewrite, its gradient evaluates whether changing $M_l$'s support preserves the next-block response; during a quantization rewrite, it adjusts $\alpha_l$ on the fixed sparse support.

In addition to response matching, we align output distributions through block-level distillation:

$$\begin{aligned} p_{l+1}^{\mathrm{fp}} &= \mathrm{softmax}(Y_{l+1}^{\mathrm{fp}}/T), \\ p_{l+1}^{\mathrm{sq}} &= \mathrm{softmax}(Y_{l+1}^{\mathrm{sq}}/T), \\ \mathcal{L}_{\mathrm{KL}}^{l} &= T^2 \, D_{\mathrm{KL}}\left( p_{l+1}^{\mathrm{fp}} \,\|\, p_{l+1}^{\mathrm{sq}} \right), \end{aligned} \tag{14}$$

where $T$ is the distillation temperature. The objectives are complementary: the anchor constrains magnitude and direction of output drift, while KL divergence preserves distributional structure.

Combining window reconstruction with cross-block objectives yields the unified loss for $\mathcal{B}_t$:

$$\mathcal{L}^{(t)} = \mathcal{L}_{\mathrm{rec}}^{(t)} + \lambda_a \sum_{l \in \mathcal{B}_t} \mathcal{L}_{\mathrm{anchor}}^{l} + \lambda_{\mathrm{KL}} \sum_{l \in \mathcal{B}_t} \mathcal{L}_{\mathrm{KL}}^{l}, \tag{15}$$

where $\lambda_a$ and $\lambda_{\mathrm{KL}}$ balance the terms. The residual buffer in Eq. equation 11 is a calibration state, not an additional loss. Both the N:M top-$n$ operator and quantization rounding are non-differentiable. In the forward pass, we retain the hard N:M mask and INT4 rounding; in the backward pass, we use a temperature-controlled sigmoid surrogate for the top-$n$ threshold:

$$\frac{\partial \, \mathrm{TopN}(s_j)}{\partial s_j} \approx \sigma\left( \frac{|w_j| - \tau_g}{\tau_{\mathrm{temp}}} \right), \tag{16}$$

where $\tau_g$ is the group-wise $n$-th largest magnitude threshold and $\tau_{\mathrm{temp}}$ controls sharpness. Following Eq. equation 5, only mask variables in $\mathcal{G}_{l,k}^{S}$ and scale variables in $\mathcal{G}_{l,k}^{Q}$ are updated at step $k$.

After optimization, SparseSliderQuant exports hard 2:4 masks and INT4 quantization parameters, discarding all auxiliary structures. The resulting model uses standard sparse and low-bit operators with no additional inference-time overhead.

## 4 EXPERIMENTS

**Models and Datasets.** We evaluate SparseSliderQuant on five decoder-only LLMs from the LLaMA and Qwen families: LLaMA-3.1-8B, LLaMA-3.1-70B Grattafiori et al. (2024), Qwen2.5-7B, Qwen2.5-14B, and Qwen2.5-32B Qwen et al. (2025), spanning 7B to 70B parameters. We report perplexity on WikiText2 Merity et al. (2016) and a subset of the C4 validation set Colin (2020), and zero-shot accuracy on eight commonsense reasoning tasks via LM Evaluation Harness Gao et al. (2024): PIQA Bisk et al. (2020), ARC-Easy, ARC-Challenge Clark et al. (2018), OpenBookQA Mihaylov et al. (2018), HellaSwag Zellers et al. (2019), WinoGrande Sakaguchi et al. (2021), BoolQ Clark et al. (2019), and RTE Dagan et al. (2005). The arithmetic mean over all eight tasks is reported as Avg. Acc. Complete per-task results are in Appendix A.5.

Table 1: Zero-shot accuracy (%) across text-only LLMs under W4A4 + 2:4 sparsity. "Ours" denotes SparseSliderQuant.

| Model | Method | ARC-C | WG | ARC-E | BQ | OBQA | HS | RTE | PIQA | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| LLaMA-3.1-70B | Base (FP16) | 59.22 | 81.69 | 85.19 | 87.16 | 36.80 | 67.44 | 70.40 | 83.19 | 71.38 |
| | SparseGPT + SmoothQuant | 25.85 | 49.88 | 26.60 | 44.50 | 27.80 | 25.99 | 49.46 | 49.46 | 37.44 |
| | Wanda + SmoothQuant | 25.60 | 49.57 | 25.17 | 38.35 | 27.00 | 26.16 | 50.90 | 52.07 | 36.85 |
| | SAFE + SmoothQuant | 24.32 | 49.25 | 27.27 | 39.72 | 28.40 | 25.65 | 53.79 | 50.76 | 37.39 |
| | Wanda++ + SmoothQuant | 25.00 | 49.09 | 26.81 | 40.76 | 27.00 | 26.29 | 52.35 | 51.74 | 37.38 |
| | SAFE + GPTAQ | 26.88 | 48.70 | 25.72 | 60.18 | 27.80 | 25.72 | 46.93 | 51.00 | 37.42 |
| | Wanda++ + GPTAQ | 25.00 | 48.00 | 25.00 | 57.00 | 27.00 | 25.63 | 52.00 | 50.00 | 37.09 |
| | JSQ | 23.04 | 55.33 | 35.48 | 58.53 | 30.00 | 49.20 | 54.51 | 57.62 | 43.73 |
| | SLIM | 26.79 | 47.99 | 25.13 | 37.95 | 29.20 | 25.82 | 51.26 | 52.77 | 34.88 |
| | OBR | 24.40 | 53.67 | 42.05 | 58.44 | 27.80 | 36.21 | 46.93 | 59.96 | 41.36 |
| | **Ours** | **44.71** | **71.83** | **71.42** | **81.50** | **38.40** | **69.82** | **59.57** | **78.45** | **64.46** |
| LLaMA-3.1-8B | Base (FP16) | 53.84 | 73.80 | 81.23 | 81.96 | 44.60 | 78.99 | 70.04 | 81.07 | 70.69 |
| | SparseGPT + SmoothQuant | 24.83 | 50.51 | 26.73 | 38.90 | 28.40 | 26.13 | 50.18 | 51.03 | 37.09 |
| | Wanda + SmoothQuant | 25.00 | 50.75 | 25.04 | 40.83 | 25.80 | 26.22 | 51.99 | 52.50 | 37.27 |
| | SAFE + SmoothQuant | 25.34 | 48.30 | 26.73 | 45.57 | 27.40 | 25.96 | 53.79 | 49.84 | 37.87 |
| | Wanda++ + SmoothQuant | 26.62 | 49.17 | 27.02 | 38.84 | 27.60 | 25.96 | 53.79 | 51.09 | 37.51 |
| | SAFE + GPTAQ | 23.98 | 50.04 | 34.09 | 55.87 | 25.00 | 37.09 | 53.43 | 56.09 | 41.95 |
| | Wanda++ + GPTAQ | 21.67 | 50.99 | 30.64 | 54.07 | 26.60 | 34.37 | 53.43 | 53.32 | 40.64 |
| | JSQ | 24.91 | 50.91 | 27.15 | 39.51 | 28.00 | 26.48 | 50.90 | 51.25 | 37.39 |
| | SLIM | 22.70 | 49.57 | 25.08 | 37.83 | 27.60 | 25.04 | 52.71 | 49.51 | 36.26 |
| | OBR | 24.40 | 53.51 | 42.00 | 61.62 | 26.80 | 36.47 | 51.99 | 57.78 | 44.32 |
| | **Ours** | **28.58** | **53.67** | **48.44** | **60.55** | **30.20** | **36.87** | **53.07** | **66.05** | **47.18** |

Table 2: Perplexity on WikiText2 and C4 under W4A4 quantization with 2:4 structured sparsity. Lower is better, and the best compressed result in each column is shown in bold.

| #Bits | Method | LLaMA3.1-8B | | LLaMA3.1-70B | | Qwen2.5-7B | | Qwen2.5-14B | | Qwen2.5-32B | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Wiki2↓ | C4↓ | Wiki2↓ | C4↓ | Wiki2↓ | C4↓ | Wiki2↓ | C4↓ | Wiki2↓ | C4↓ |
| W16A16 | FP16 | 6.24 | 9.54 | 2.81 | 7.11 | 6.85 | 11.89 | 5.29 | 10.35 | 5.02 | 10.18 |
| W4A4+2:4 | SAFE + GPTAQ | 24.38 | 36.92 | 14.52 | 22.18 | 42.58 | 68.35 | 28.65 | 45.72 | 22.18 | 35.84 |
| | Wanda++ + GPTAQ | 22.65 | 34.58 | 13.82 | 21.35 | 38.42 | 62.18 | 26.82 | 42.95 | 20.45 | 33.28 |
| | JSQ | 28.95 | 45.62 | 15.28 | 24.52 | 48.85 | 78.52 | 32.18 | 52.35 | 25.62 | 41.85 |
| | SLIM | 32.45 | 52.18 | 18.65 | 30.28 | 55.28 | 88.95 | 38.52 | 62.18 | 30.85 | 50.62 |
| | OBR | 36.18 | 48.88 | 12.18 | 18.92 | — | — | 28.95 | 46.52 | 22.35 | 36.18 |
| | **Ours** | **12.78** | **18.52** | **11.15** | **17.82** | **35.52** | **58.85** | **23.82** | **38.28** | **19.05** | **31.65** |

**Baselines and Implementation Details.** We compare with sequential (SAFE+GPTAQ, Wanda+++GPTAQ) and joint methods (JSQ Guo et al. (2024), SLIM Mozaffari et al. (2024), OBR Guo et al. (2025)), with FP16 as reference. All models use W4A4 quantization (group size 128) with 2:4 sparsity. Calibration uses 128 sequences of length 2,048 from C4. We set window size $s$=4, stride $r$=2, group size $g$=128, and a four-step cycle (Eq. 5). Loss weights are $\lambda_a$=1.0, $\lambda_{\mathrm{KL}}$=0.3, $T$=1.0; residual buffer uses $\beta$=0.9, $\eta_R$=0.5. We optimize with AdamW (peak $5 \times 10^{-5}$, linear decay, warmup 0.05). All experiments are conducted on NVIDIA RTX 4090 GPUs.

**Sparse-Quantization Performance.** Tables 1 and 2 report zero-shot accuracy and perplexity. SparseSliderQuant consistently outperforms compressed baselines. On LLaMA-3.1-8B, it achieves WikiText2/C4 perplexities of 12.78/18.52, lower than 22.65/34.58 for Wanda+++GPTAQ and 36.18/48.88 for OBR, while attaining 47.18% zero-shot accuracy compared with 44.32% for OBR and 41.95% for SAFE+GPTAQ. On LLaMA-3.1-70B, it reduces perplexity to 11.15/17.82, outperforming OBR at 12.18/18.92, and improves average accuracy to 64.46%—23.10 points above OBR (41.36%) and 27.04 points above SAFE+GPTAQ (37.42%). Across the Qwen2.5 family, SparseSliderQuant obtains WikiText2/C4 perplexities of 35.52/58.85 on 7B, 23.82/38.28 on 14B, and 19.05/31.65 on 32B, consistently achieving best compressed result at every model scale. As shown in Fig. 1, it also reaches 47.18% accuracy in 88.0 minutes, compared with 44.32% in 93.7 minutes for OBR. These gains confirm that local reconstruction alone is insufficient under W4A4+2:4 compression; alternating sparsification and quantization signals, retaining coupled residuals, and constraining next-block drift better preserve model quality without inference overhead.

**Error Analysis.** We analyze three aspects of compression error on LLaMA-3.1-8B.

*Per-block loss.* Figure 3 compares reconstruction loss and similarity across 32 blocks. SAFE+GPTAQ accumulates error fastest; OBR slows growth within each block but lacks cross-block feedback; SparseSliderQuant maintains lowest loss and highest similarity, with the gap widening after block 16 where calibration and anchoring take effect.

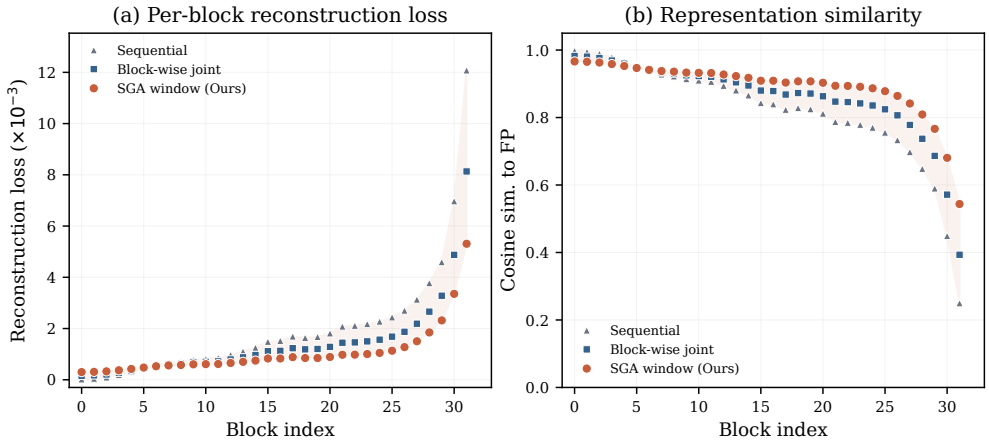



Figure 3: Per-block reconstruction loss (a) and similarity to full-precision (b) across 32 blocks on LLaMA-3.1-8B. SparseSliderQuant maintains the lowest loss and highest similarity throughout.

Table 3: Ablation study of the main components on WikiText2 PPL and average zero-shot accuracy for LLaMA-3.1-8B under W4A4 quantization with 2:4 structured sparsity.

| Metric | Full | w/o SGA | w/o DMC | w/o CGA | w/o ShiftSwap | w/o Window | w/o Block KL |
|---|---|---|---|---|---|---|---|
| WikiText2 PPL ↓ | **12.78** | 16.82 | 14.30 | 15.49 | 15.12 | 16.25 | 13.68 |
| Δ PPL ↓ | — | +4.04 | +1.52 | +2.71 | +2.34 | +3.47 | +0.90 |
| Avg. Acc. ↑ | **47.18** | 43.06 | 45.31 | 44.27 | 44.68 | 43.57 | 46.24 |

*Token-level error.* Figure 4(a) visualizes sparse-versus-full-precision output differences across 2,048 tokens (first 500 dimensions) for five methods. Overall error decreases from JSQ (9), SAFE+GPTAQ (8.5), SLIM (8), OBR (7.5) to Ours (5). SparseSliderQuant's error is concentrated in early layers and attenuates with depth, whereas baselines exhibit persistent high-magnitude patches.

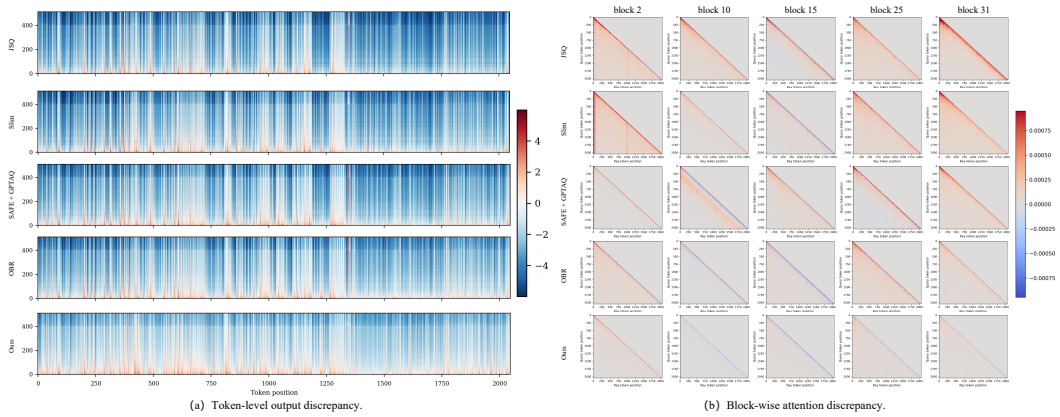
*Attention-level error.* Figure 4(b) shows signed sparse-minus-FP attention differences (red: positive, blue: negative, white: zero) for blocks 2, 10, 15, 25, 31. Early blocks exhibit strong patches across all methods. Error attenuates with depth; block 31 is near-white. SparseSliderQuant achieves lightest patterns from block 15 onward, confirming residual feedback and anchoring suppress attention distortion where baselines retain visible color. Additional analyses are in Appendix A.6.

### 4.1 Ablation Study

We conduct ablation studies on LLaMA-3.1-8B under W4A4 + 2:4 sparsity. The full method achieves WikiText2 PPL of 12.78 and Avg. Acc. of 47.18%.

**Effect of Core Components.** Table 3 examines each component's contribution. Removing sliding group alternation causes the largest degradation (PPL 12.78→16.82), as mask and scale groups receive isolated signals. Removing sliding window raises PPL to 16.25, confirming block-wise reconstruction cannot constrain cross-block propagation. Removing ShiftSwap role exchange raises PPL to 15.12, as groups remain in their initial $S/Q$ assignment and cannot experience both compression modes. Removing KL alignment yields a smaller increase to 13.68. Removing mismatch calibration raises PPL to 14.30 and drops accuracy to 45.31%, whereas removing gradient anchoring causes larger degradation (15.49 PPL, 44.27% accuracy), suggesting residual feedback compensates local mismatch while gradient anchoring constrains downstream propagation.

**Effect of Initial Group Assignment.** Table 4 varies the initial sparse/quant group ratio. The balanced 50/50 assignment achieves the best result (PPL 12.78, Acc. 47.18%). Skewing toward either mode degrades performance: 75/25 raises PPL to 13.65, 25/75 to 13.42, and extremes 100/0 and 0/100 to 16.45 and 15.88, confirming both compression modes require sufficient capacity.




(a) Token-level output discrepancy. (b) Block-wise attention discrepancy.

Figure 4: Token-level and attention-level error analysis on LLaMA-3.1-8B. In **(a)**, SparseSliderQuant best aligns raw logits with FP16 and suppresses heavy-tailed deviations. In **(b)**, it minimizes attention error across blocks, preserving diffuse and localized token dependencies.

**Effect of Different Mask Sources.** The differentiable N:M mask achieves PPL 12.78 / Acc. 47.18%, outperforming fixed masks from SparseGPT (13.45), Wanda (13.68), SAFE (13.82), and magnitude (13.91). Fixed masks commit to sparse support before observing quantization or cross-block errors, while the differentiable mask revises pruning boundaries via downstream gradients. The 0.67 PPL gap over the best fixed mask (SparseGPT) rivals mismatch calibration alone (Table 3), indicating mask plasticity is as important as residual feedback under joint W4A4 compression.

**Influence of Residual-Feedback Strength.** Table 4 varies $\eta_R$. Disabling feedback ($\eta_R$=0) yields PPL 14.30 / Acc. 45.31%. Increasing to 0.10 and 0.25 improves to 13.97/45.88% and 13.24/46.42%. The default $\eta_R$=0.50 achieves best performance (12.78/47.18%), balancing insufficient and excessive correction. Further increasing to 1.00 degrades results to 13.51/46.05%, suggesting excessive feedback over-corrects retained weights.

Table 4: Ablation of group ratio, mask source, and residual strength under W4A4 + 2:4 sparsity.

| *Group ratio (Sparse/Quant)* | | | | | |
|---|---|---|---|---|---|
| Ratio | 100/0 | 75/25 | 50/50 | 25/75 | 0/100 |
| WikiText2 ↓ | 16.45 | 13.65 | **12.78** | 13.42 | 15.88 |
| Avg. Acc. ↑ | 43.12 | 45.82 | **47.18** | 46.21 | 44.37 |
| *Mask source* | | | | | |
| Method | Mag. | SparseGPT | Wanda | SAFE | Ours |
| WikiText2 ↓ | 13.91 | 13.45 | 13.68 | 13.82 | **12.78** |
| Avg. Acc. ↑ | 45.21 | 46.08 | 45.73 | 45.49 | **47.18** |
| *Residual strength $\eta_R$* | | | | | |
| $\eta_R$ | 0 | 0.10 | 0.25 | 0.50 | 1.00 |
| WikiText2 ↓ | 14.30 | 13.97 | 13.24 | **12.78** | 13.51 |
| Avg. Acc. ↑ | 45.31 | 45.88 | 46.42 | **47.18** | 46.05 |

## 5 Conclusion

We presented SparseSliderQuant, a joint sparsification and quantization framework that reformulates independent block-wise compression as coordinated cross-block error optimization. Within overlapping Transformer-block windows, sliding group alternation exposes parameter groups to both sparse-mask and quantization-scale updates while avoiding unstable simultaneous optimization. Dual-source mismatch calibration decomposes pruning and quantization residuals, accumulates their coupled error, and feeds it back to unpruned weights. Cross-block gradient anchoring further optimizes masks and scales according to next-block output drift, aligning local reconstruction with downstream behavior. Experiments on the LLaMA and Qwen model families demonstrate that SparseSliderQuant consistently improves language-modeling perplexity and zero-shot reasoning accuracy under W4A4 quantization with 2:4 structured sparsity. The method introduces no auxiliary inference-time modules and exports standard INT4 scales and hard 2:4 masks after calibration. These results establish cross-block error coordination as an effective approach to accurate and hardware-friendly sparse-quantized LLM deployment.

## AI USE STATEMENT

We did not use generative AI tools for any task in this work, including research design, methodology, experiments, data analysis, or writing. All content, including text, figures, tables, and code, was produced solely by the authors. We take full responsibility for the final content of this work.

## 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.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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.

Wenhua Cheng, Weiwei Zhang, Heng Guo, Haihao Shen, and Zaner Ma. Signroundv2: Toward closing the performance gap in extremely low-bit post-training quantization for llms. *arXiv preprint arXiv:2512.04746*, 2025.

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.

Raffel Colin. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21, 2020.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pp. 177–190. Springer, 2005.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale, 2022. URL `https://arxiv.org/abs/2208.07339`.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

Xin Ding, Xiaoyu Liu, Zhijun Tu, Yun Zhang, Wei Li, Jie Hu, Hanting Chen, Yehui Tang, Zhiwei Xiong, Baoqun Yin, et al. Cbq: Cross-block quantization for large language models. In *International Conference on Learning Representations*, volume 2025, pp. 7056–7075, 2025.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. *arXiv preprint arXiv:2406.02924*, 2024.

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.

L Gao et al. A framework for few-shot evaluation of language models. *EleutherAI*, 2024.

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.

Hang Guo, Yawei Li, and Luca Benini. Optimal brain restoration for joint quantization and sparsification of llms. *arXiv preprint arXiv:2509.11177*, 2025.

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.

Dongyeop Lee, Kwanhee Lee, Jinseok Chung, and Namhoon Lee. Safe: Finding sparse and flat minima to improve pruning. *arXiv preprint arXiv:2506.06866*, 2025.

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.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 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, 2024. *URL https://arxiv. org/abs/2405.16406*, 2025.

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.

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.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

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 Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024a.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

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*, 2023b.

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.

Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus Müller, Jonas M Kübler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, et al. Wanda++: Pruning large language models via regional gradients. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4321–4333, 2025.

Zhiguo Yang, Changjian Deng, Qinke Chen, Zijing Zhou, and Jian Cheng. Lsa: Layer-wise sparsity allocation for large language model pruning based on minimal linear reconstruction error. In *The Fourteenth International Conference on Learning Representations*, 2026.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in neural information processing systems*, 35:27168–27183, 2022.

Ke Yi, Zengke Liu, Chengyuan Li, Tong Zhang, Junyang Lin, Jingren Zhou, et al. Rotated runtime smooth: Training-free activation smoother for accurate int4 inference. In *International Conference on Learning Representations*, volume 2025, pp. 17168–17184, 2025.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity, 2025. URL https://arxiv.org/abs/2310.05175.

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.

Yuxin Zhang, Lirui Zhao, Mingbao Lin, Sun Yunyun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. In *International Conference on Learning Representations*, volume 2024, pp. 249–264, 2024.

# A APPENDIX

## A.1 IMPLEMENTATION DETAILS OF SPARSESLIDERQUANT

This supplementary section details the quantization and sparsity configurations used throughout the paper, followed by additional ablation studies, results across diverse model families, a detailed error analysis, and the full SparseSliderQuant algorithm.

## A.2 QUANTIZATION AND SPARSITY CONFIGURATION

We adopt symmetric uniform quantization for both weights and activations. Given a tensor $z$, bit-width $b$, and $q_{\max} = 2^{b-1} - 1$, the quantization scale and quantizer are

$$\alpha = \frac{\max |z|}{q_{\max}}, \qquad Q_b(z; \alpha) = \alpha \, \mathrm{clip}\Big(\mathrm{round}\Big(\frac{z}{\alpha}\Big), -q_{\max}, q_{\max}\Big). \tag{A}$$

The quantizer is applied group-wise with group size 128 to both weights and activations. We set $b = 4$ for W4A4 compression and use the straight-through estimator (STE) to approximate the gradient of the rounding operation during calibration.

For weights, each group of 128 consecutive elements shares a single scale $\alpha_l^{(g)}$. The sparse–quantized weight is $\widehat{W}_l = Q_4(M_l \odot W_l; \alpha_l)$, where $M_l$ enforces the 2:4 structured sparsity pattern. For activations, per-token quantization is applied: each token's hidden vector is divided into groups of 128 elements, each with its own scale. Activation quantization is performed before the first rewrite step and remains fixed throughout calibration.

For the N:M structured sparsity, we use the 2:4 pattern throughout all linear layers. Within each consecutive group of 4 elements, the 2 entries with the largest magnitudes are retained. The mask $M_l$ is initialized via magnitude pruning and then updated during calibration via the sliding group alternation described in Section 3.2.

## A.3 HYPER-PARAMETER SETTINGS

Table A summarizes the default hyper-parameters used across all experiments.

Table A: Default hyper-parameter settings for SparseSliderQuant.

| Configuration | Setting |
|---|---|
| Calibration set | C4 |
| Number of calibration samples | 128 |
| Tokens per sample | 2,048 |
| Quantization | W4A4 symmetric, group size 128 |
| Sparsity | 2:4 structured |
| Window size $s$ | 4 |
| Stride $r$ | 2 |
| Group size $g$ | 128 |
| Number of groups per block | 4 |
| Assignment cycle | $(SQSQ, QSSQ, QSQS, SQQS)$ |
| Rewrite steps $K$ | 30 |
| Momentum $\beta$ | 0.9 |
| Feedback strength $\eta_R$ | 0.50 |
| $\lambda_a$ | 1.0 |
| $\lambda_{\mathrm{KL}}$ | 0.3 |
| Distillation temperature $T$ | 1.0 |
| STE temperature $\tau_{\mathrm{temp}}$ | 0.1 |
| Optimizer | AdamW |
| Peak learning rate | $5 \times 10^{-5}$ |
| Learning rate schedule | linear decay, warmup 0.05 |

### A.4 Effect of Calibration Sample Count

To evaluate the sensitivity of SparseSliderQuant to calibration data size, we vary the number of calibration samples from 32 to 512 on LLaMA-3.1-8B under W4A4 + 2:4 sparsity. As shown in Table B, even with only 32 samples, SparseSliderQuant achieves competitive perplexity. Performance improves gradually with more samples, and 128 samples offer a good balance between calibration cost and model quality. We adopt 128 samples as the default throughout all experiments.

Table B: Effect of the number of calibration samples on LLaMA-3.1-8B under W4A4 + 2:4 sparsity.

| #Samples | WikiText2 ↓ | C4 ↓ | Time (min) |
|---|---|---|---|
| 32 | 13.85 | 20.07 | 51.8 |
| 64 | 13.21 | 19.15 | 62.7 |
| 128 | 12.78 | 18.52 | 88.0 |
| 256 | 12.71 | 18.42 | 168.6 |
| 512 | 12.66 | 18.35 | 331.9 |

### A.5 More Results Across Diverse LLM Families and Quantization Settings

Table C: Zero-shot accuracy (%) across Qwen2.5 models under W4A4 + 2:4 sparsity. "Ours" denotes SparseSliderQuant.

| Model | Method | ARC-C | WG | ARC-E | BQ | OBQA | HS | RTE | PIQA | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| Qwen2.5-32B | Base (FP16) | 55.29 | 75.30 | 77.99 | 87.46 | 44.20 | 84.06 | 80.87 | 82.26 | 73.43 |
| | SparseGPT + SmoothQuant | 24.32 | 51.14 | 29.04 | 42.78 | 27.40 | 26.90 | 50.54 | 51.31 | 37.93 |
| | Wanda + SmoothQuant | 25.17 | 49.96 | 26.18 | 39.48 | 28.60 | 25.54 | 54.87 | 51.63 | 37.68 |
| | SAFE + SmoothQuant | 24.74 | 51.70 | 27.82 | 40.21 | 25.40 | 26.59 | 49.82 | 51.69 | 37.25 |
| | Wanda++ + SmoothQuant | 24.66 | 51.78 | 30.64 | 46.09 | 25.20 | 29.18 | 49.82 | 53.75 | 38.89 |
| | SAFE + GPTAQ | 25.51 | 50.51 | 33.46 | 54.22 | 29.00 | 36.59 | 53.79 | 52.39 | 41.93 |
| | Wanda++ + GPTAQ | 23.12 | 50.59 | 26.60 | 43.15 | 26.40 | 29.30 | 52.35 | 51.41 | 37.87 |
| | JSQ | 21.76 | 48.93 | 27.10 | 60.18 | 32.00 | 27.35 | 52.71 | 52.50 | 40.32 |
| | SLIM | 22.70 | 49.57 | 25.08 | 38.26 | 27.80 | 25.04 | 52.71 | 49.51 | 36.33 |
| | OBR | 23.63 | 51.85 | 31.40 | 46.48 | 25.20 | 29.89 | 51.62 | 52.99 | 39.13 |
| | **Ours** | **30.05** | **55.80** | **41.16** | **63.42** | **33.40** | **44.87** | **56.68** | **58.92** | **48.04** |
| Qwen2.5-14B | Base (FP16) | 59.22 | 75.06 | 79.25 | 85.35 | 45.20 | 82.98 | 79.78 | 79.78 | 73.61 |
| | SparseGPT + SmoothQuant | 26.02 | 51.07 | 24.16 | 54.65 | 27.00 | 26.77 | 48.38 | 49.73 | 38.47 |
| | Wanda + SmoothQuant | 24.40 | 47.59 | 26.18 | 48.32 | 29.40 | 25.66 | 48.74 | 51.41 | 37.71 |
| | SAFE + SmoothQuant | 26.54 | 50.59 | 27.27 | 47.83 | 27.60 | 25.95 | 53.43 | 52.99 | 39.02 |
| | Wanda++ + SmoothQuant | 24.74 | 49.41 | 28.54 | 55.17 | 26.40 | 27.96 | 51.99 | 49.24 | 39.18 |
| | SAFE + GPTAQ | 21.50 | 49.80 | 34.05 | 51.44 | 25.80 | 30.35 | 53.07 | 55.06 | 40.13 |
| | Wanda++ + GPTAQ | 24.32 | 50.28 | 27.31 | 42.26 | 28.60 | 27.35 | 51.26 | 51.90 | 37.91 |
| | JSQ | 25.68 | 48.62 | 25.13 | 39.27 | 24.80 | 25.64 | 51.99 | 48.64 | 36.22 |
| | SLIM | 22.70 | 56.67 | 25.08 | 48.72 | 31.80 | 28.67 | 58.84 | 49.51 | 40.25 |
| | OBR | 24.57 | 51.07 | 27.44 | 40.34 | 25.80 | 26.09 | 58.12 | 50.98 | 38.05 |
| | **Ours** | **28.24** | **57.14** | **42.08** | **59.85** | **33.20** | **39.42** | **60.65** | **61.48** | **47.76** |
| Qwen2.5-7B | Base (FP16) | 51.19 | 73.32 | 77.61 | 84.56 | 47.00 | 78.91 | 81.23 | 79.87 | 71.71 |
| | SparseGPT + SmoothQuant | 26.54 | 51.22 | 28.11 | 38.26 | 27.60 | 26.42 | 51.62 | 50.60 | 37.55 |
| | Wanda + SmoothQuant | 24.06 | 51.38 | 26.56 | 40.89 | 26.60 | 26.28 | 50.54 | 50.44 | 37.09 |
| | SAFE + SmoothQuant | 23.21 | 49.57 | 26.47 | 40.89 | 28.00 | 26.20 | 51.99 | 49.56 | 36.99 |
| | Wanda++ + SmoothQuant | 23.12 | 48.22 | 26.94 | 40.40 | 29.60 | 26.75 | 52.35 | 50.87 | 37.28 |
| | SAFE + GPTAQ | 25.94 | 50.83 | 26.98 | 38.75 | 50.83 | 25.76 | 51.26 | 49.62 | 36.89 |
| | Wanda++ + GPTAQ | 26.54 | 49.49 | 25.93 | 38.07 | 25.40 | 26.22 | 49.82 | 50.54 | 36.50 |
| | JSQ | 25.77 | 49.25 | 25.46 | 38.29 | 27.40 | 26.18 | 54.51 | 52.07 | 37.37 |
| | SLIM | 22.70 | 49.57 | 25.08 | 37.83 | 27.60 | 25.04 | 52.71 | 49.51 | 36.26 |
| | OBR | — | — | — | — | — | — | — | — | — |
| | **Ours** | **30.48** | **54.86** | **35.72** | **45.62** | **31.60** | **33.28** | **56.68** | **57.45** | **43.21** |

In the main paper, we report results on LLaMA-3.1-8B/70B and Qwen2.5-7B/14B/32B. Table C provides the complete zero-shot accuracy results for the three Qwen2.5 models under W4A4 + 2:4 sparsity, complementing the LLaMA results in the main text. We compare SparseSliderQuant with sequential pipelines (SAFE + GPTAQ, Wanda++ + GPTAQ), joint compression methods (JSQ, SLIM, OBR), and SmoothQuant-based baselines. The FP16 model is included as an upper reference.

Across all three Qwen2.5 models, SparseSliderQuant is the only method that consistently ranks first on average zero-shot accuracy. On Qwen2.5-7B, OBR results are unavailable because its

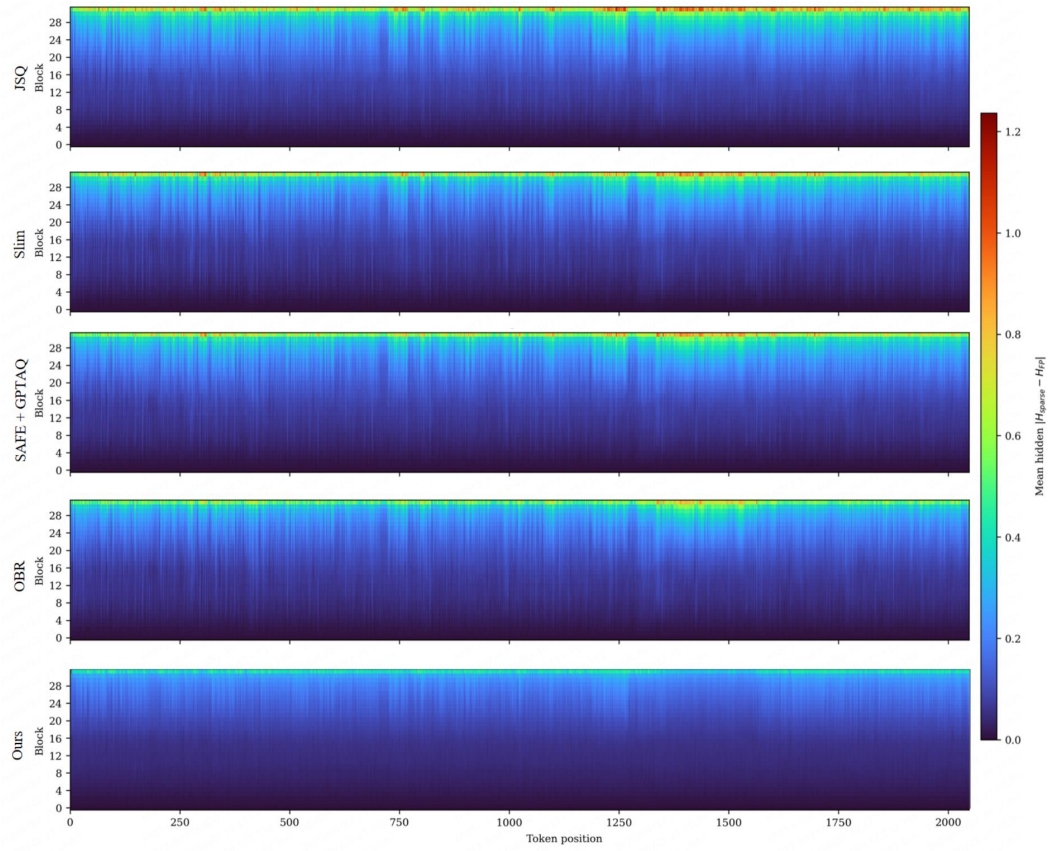



Figure A: Average hidden-state reconstruction error across Transformer blocks on LLaMA-3.1-8B. Each row reports $\text{mean}_h|H_{\text{sparse}} - H_{\text{FP}}|$ per block and token position, using a shared color scale. SparseSliderQuant achieves the lowest and most uniform error among all methods, demonstrating its ability to suppress cross-block accumulation of sparsification and quantization errors.

Hessian-based weight restoration requires computing the Hessian inverse, which produces NaN values due to numerical instability in matrix inversion, preventing successful calibration. SparseSliderQuant outperforms the next-best method (JSQ) by 5.84 points in Avg. Acc. (43.21% vs. 37.37%). SmoothQuant-based baselines collapse across all models, rarely exceeding 38% average accuracy, confirming that weight-only activation smoothing cannot compensate for the coupled pruning–quantization error under joint W4A4 compression.

### A.6 Detailed Error Analysis

This section complements the per-block loss, token-level, and attention-level analyses in the main text (Figure 3 and Figure 4) by examining hidden-state reconstruction error at the block level, providing a spatial view of how compression errors distribute across token positions and decoder depth.

#### A.6.1 Hidden-State Reconstruction Error

Figure A reports the average $\ell_2$ hidden-state reconstruction error $\text{mean}_h|H_{\text{sparse}} - H_{\text{FP}}|$ at each Transformer block on LLaMA-3.1-8B, using a single shared color scale so that absolute error magnitudes are directly comparable across methods. The error is computed on one calibration sequence of length 2,048; each row corresponds to a Transformer block and each column to a token position. SparseSliderQuant maintains low error throughout all 32 blocks, with the spatial distribution of error remaining relatively uniform across token positions. In contrast, JSQ and SLIM exhibit visible error growth starting around block 16, with the largest deviations concentrated in the upper half of the decoder—particularly at sequence positions corresponding to high-attention tokens. SAFE+GPTAQ shows a similar but less severe pattern, as its GPTQ-based weight calibration partially compensates for pruning error but cannot address the coupling between the two compression sources. OBR yields the second-lowest error among the compared methods, consistent with its joint

**Algorithm 1** SparseSliderQuant Rewrite Procedure

**Require:** FP16 blocks $\{F_l, W_l\}_{l=1}^{L}$, calibration data $\mathcal{D}$, window size $s$, stride $r$, rewrite steps $K$, group size $g$, momentum $\beta$, feedback strength $\eta_R$, coefficients $\lambda_a, \lambda_{\mathrm{KL}}$, temperature $T$, STE temperature $\tau_{\mathrm{temp}}$

**Ensure:** W4A4 + 2:4 sparse-quantized model

Initialize $M_l \leftarrow$ magnitude-based 2:4 mask, $\alpha_l \leftarrow$ per-group scale, $R_l \leftarrow \mathbf{0}$ for all $l$
Initialize $\pi^{(0)} \leftarrow SQSQ$ {Eq. equation 5}
**for** $t = 0$ **to** $L - s$ **step** $r$ **do**
  $\mathcal{B}_t \leftarrow \{t, t+1, \ldots, t+s-1\}$
  Cache $X_{b_t}$ from $\mathcal{D}$; compute $Y^{\mathrm{fp}}_{\mathcal{B}_t} \leftarrow F_{\mathcal{B}_t}(\{W_l\}, X_{b_t})$
  Cache next-block targets $Y^{\mathrm{fp}}_{t+s}$ from FP path
  **for** $k = 1$ **to** $K$ **do**
    **for** each block $l \in \mathcal{B}_t$ **do**
      $\widehat{W}_l \leftarrow Q_4(M_l \odot W_l;\, \alpha_l)$ {Eq. equation 1}
      $\Delta^s_l \leftarrow (1 - M_l) \odot W_l, \quad \Delta^q_l \leftarrow M_l \odot W_l - \widehat{W}_l$ {Eq. equation 9}
      $R_l \leftarrow \beta\, R_l + \Delta^s_l + \Delta^q_l$ {Eq. equation 11}
      $W_l \leftarrow W_l + \eta_R\, \Pi_{\Omega^{\mathrm{free}}_l}(R_l)$ {Eq. equation 12}
    **end for**
    Compute $Y^{\mathrm{sq}}_{\mathcal{B}_t} \leftarrow F_{\mathcal{B}_t}(\{\widehat{W}_l\}, X_{b_t})$
    $\mathcal{L}_{\mathrm{rec}} \leftarrow \|Y^{\mathrm{sq}}_{\mathcal{B}_t} - Y^{\mathrm{fp}}_{\mathcal{B}_t}\|_2^2$ {Eq. equation 8}
    **for** each block $l \in \mathcal{B}_t$ **do**
      Compute $Y^{\mathrm{sq}}_{l+1}$ from sparse–quantized path; retrieve cached $Y^{\mathrm{fp}}_{l+1}$
      $\mathcal{L}^l_{\mathrm{anchor}} \leftarrow \|Y^{\mathrm{sq}}_{l+1} - Y^{\mathrm{fp}}_{l+1}\|_2^2 \,/\, (\|Y^{\mathrm{fp}}_{l+1}\|_2^2 + \epsilon)$ {Eq. equation 13}
      $\mathcal{L}^l_{\mathrm{KL}} \leftarrow T^2\, D_{\mathrm{KL}}\big(\mathrm{softmax}(Y^{\mathrm{fp}}_{l+1}/T) \,\|\, \mathrm{softmax}(Y^{\mathrm{sq}}_{l+1}/T)\big)$ {Eq. equation 14}
    **end for**
    $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{rec}} + \lambda_a \sum_l \mathcal{L}^l_{\mathrm{anchor}} + \lambda_{\mathrm{KL}} \sum_l \mathcal{L}^l_{\mathrm{KL}}$ {Eq. equation 15}
    Update $M_l$ for $g \in \mathcal{G}^S_k$ via $\nabla_{M_l}\mathcal{L}$ with $\alpha_l$ fixed; use STE surrogate (Eq. equation 16) {Eq. equation 6}
    Update $\alpha_l$ for $g \in \mathcal{G}^Q_k$ via $\nabla_{\alpha_l}\mathcal{L}$ with $M_l$ fixed {Eq. equation 7}
    $\pi^{(k \bmod 4)} \leftarrow$ next in cycle $(SQSQ \to QSSQ \to QSQS \to SQQS)$ {Eq. equation 5}
  **end for**
  Carry over $\{M_l, \alpha_l, R_l\}$ for $l \in \mathcal{B}_t \cap \mathcal{B}_{t+1}$ as warm start
**end for**
Export hard 2:4 masks $\{M_l\}$ and INT4 scales $\{\alpha_l\}$; discard $\{R_l\}$, cached FP targets, and anchor states
**return** Sparse-quantized LLM

optimization formulation, but its per-block local objective cannot prevent the gradual upward drift observed after block 24. These observations confirm that the cross-block gradient anchoring and dual-source residual feedback in SparseSliderQuant are effective not only at reducing the average error magnitude but also at maintaining spatial uniformity across the sequence, which is important for preserving long-range dependencies in autoregressive generation.

### A.7 SparseSliderQuant Algorithm

Algorithm 1 summarizes the SparseSliderQuant procedure. Full-precision window outputs and next-block targets are cached once per window. Each rewrite step constructs the sparse-quantized weight, decomposes pruning and quantization residuals, and accumulates them into a dual-source buffer with momentum. The buffer is projected onto unpruned support and fed back to retained weights. Mask and scale parameters are updated alternately per the four-step cycle (Eq. equation 5): $\mathcal{G}^S_k$ groups receive mask gradients with scales frozen, $\mathcal{G}^Q_k$ groups receive scale gradients with masks frozen. The unified objective (Eq. equation 15) combines window reconstruction, cross-block anchoring, and KL distillation. After $K$ steps, calibrated states of overlapping blocks serve as warm starts. After calibration, hard 2:4 masks and INT4 scales are exported; auxiliary structures are discarded.