NorSA: Accelerating LLM Decoding via Normalized Sparse Activation

Tianteng Gu[1] **Bo Xiao**[2] **Ke Zeng**[2] **Shuai Shi**[3]
Wangyou Zhang[1] **Chengda Li**[1] **Yanmin Qian**[1*]
[1]Shanghai Jiao Tong University [2]Meituan [3]VUI Labs
995999277@sjtu.edu.cn yanminqian@sjtu.edu.cn

Abstract

Sparse activation accelerates Large Language Model (LLM) decoding by removing redundant computations and memory access, but existing methods often assume hidden-state dimensions are independent and identically distributed (i.i.d.). We replace this i.i.d. perspective with a covariance-aware view by analyzing contextual dependencies across tokens and inter-dimensional correlations within hidden states. We introduce Normalized Sparse Activation (NorSA), a training-free framework that combines dynamic thresholding with decorrelating rotation. Across the LLaMA, Mistral, and Qwen families, NorSA consistently improves perplexity and downstream accuracy over prior training-free baselines, especially in high-sparsity regimes. On LLaMA3-8B at 50% sparsity, it stays within 0.44 perplexity points of the dense model while achieving a 1.32× end-to-end decoding speedup.

1 Introduction

Large Language Models (LLMs) have demonstrated exceptional capabilities across diverse tasks [OpenAI, 2024, Google, 2025, Anthropic, 2026], yet their deployment in time-sensitive applications remains limited by slow inference. Autoregressive decoding is largely memory-bound: matrix multiplications dominate both runtime and bandwidth usage. Existing acceleration approaches, including quantization [Frantar et al., 2022, Lin et al., 2024, Liu et al., 2025c, Tseng et al., 2024], pruning [Han et al., 2016, Frantar and Alistarh, 2023, Sun et al., 2024, Ma et al., 2023], and speculative decoding [Leviathan et al., 2023], each address part of this bottleneck.

Sparse activation offers a promising alternative by selectively eliminating activation dimensions during forward propagation [Li et al., 2023, Liu et al., 2023]. By avoiding unnecessary weight loads, it reduces both computation and memory access without changing model architecture. Prior work shows that many LLMs can be sparsified at inference time with only modest degradation.

This advantage is especially relevant in latency-sensitive `BS=1` interactive inference, where tokens are generated sequentially and wall-clock latency is a primary concern. On-device agents, coding copilots, and agents running on smartphones typically operate in such settings, making sparse activation a natural target for edge deployment.

The central challenge is identifying which dimensions to prune while preserving output quality. Deja Vu [Liu et al., 2023] uses a small neural network for this purpose, but introduces extra overhead. Recent training-free methods, including CATS [Lee et al., 2024], TEAL [Liu et al., 2025a], and La RoSA [Liu et al., 2025b], instead rely on thresholding or top-k selection. However, they implicitly assume activations are independent and identically distributed (i.i.d.), ignoring the structured variance and correlation present in LLM features.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In this work, we replace this i.i.d. perspective with a covariance-aware view of activation sparsity. Our analysis reveals both inter-dimensional correlations within hidden states and contextual dependencies across tokens, which are reflected in each token's hidden state through attention.

We propose Normalized Sparse Activation (NorSA), a training-free framework with two components: (1) dynamic thresholding, which scales with each hidden-state norm to account for contextual magnitude and suppress outlier effects; and (2) decorrelating rotation, which reduces linear correlations and improves importance estimation. Experiments across the LLaMA [Touvron et al., 2023, Grattafiori et al., 2024], Mistral [Jiang et al., 2023], and Qwen [Yang et al., 2024] families show that NorSA consistently improves perplexity and downstream accuracy over prior training-free baselines. On LLaMA3-8B at 50% sparsity, it stays within 0.44 perplexity points of the dense model while achieving a 1.32$\times$ end-to-end decoding speedup. These results establish NorSA as an effective training-free sparse activation method for accelerating LLM decoding.

2 Related Work

Sparse activation is a promising technique for accelerating inference of large language models by reducing unnecessary computations. It leverages the characteristic that many hidden states in a model can be zeroed out with minimal impact on performance. Early works, such as Li et al. [2023] and Liu et al. [2023], successfully utilized this natural sparsity in ReLU-based architectures to enhance inference speed.

As large language models evolved to incorporate non-ReLU activations like SwiGLU, researchers pursued new strategies to induce sparsity. Efforts by Mirzadeh et al. [2024] and Zhang et al. [2024] included architectural modifications to foster natural sparsity, while Song et al. [2025] introduced regularization techniques during training to improve sparse activation patterns.

Recently, training-free methods have gained popularity for their applicability to existing models. CATS [Lee et al., 2024] introduced magnitude pruning for nonlinearly activated intermediate hidden states in feedforward networks. TEAL [Liu et al., 2025a] expanded this line by applying sparsification to input activations across model components, and La RoSA [Liu et al., 2025b] further introduced rotation before Top-k selection. Despite their differences, these methods largely assume that activations are approximately i.i.d., or should be transformed toward a more homogeneous i.i.d.-like form so that magnitude-based selection becomes reliable. This shared premise is exactly what we revisit.

We argue that this assumption is problematic for LLM activations. Evidence from quantization points in the opposite direction: LLM.int8() [Dettmers et al., 2022], AWQ [Lin et al., 2024], and later analyses [An et al., 2025] show systematic outliers and highly non-uniform channel importance. In other words, activation structure is intrinsic to LLM representations and should be modeled rather than homogenized away.

Our work builds on this observation in a different way from quantization. Prior quantization methods use non-i.i.d. activations mainly to protect salient channels during mixed-precision compression. In contrast, we use the same evidence to motivate a covariance-aware view of sparse activation: rotation is useful not because it Gaussianizes features, but because it decorrelates them and compacts energy. This turns non-i.i.d. activation structure into a sparsification principle for training-free sparsification.

3 Background

3.1 Sparse Activation

In Large Language Models, linear layers dominate the computational workload. A standard forward pass is formulated as $\mathbf{y} = \mathbf{W}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^{d_{in}}$, $\mathbf{W} \in \mathbb{R}^{d_{out} \times d_{in}}$, and $\mathbf{y} \in \mathbb{R}^{d_{out}}$. Writing $\mathbf{w}_i = \mathbf{W}_{:,i}$ for the i-th column of $\mathbf{W}$, this operation can be decomposed into a sum of column-wise contributions:

$$\mathbf{y} = \sum_{i=1}^{d_{in}} x_i \mathbf{w}_i$$

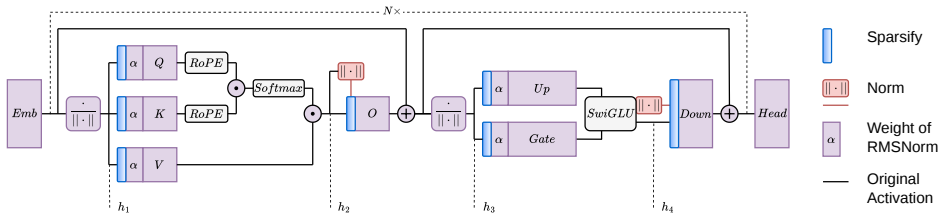

Figure 1: Sparsely activated Transformer architecture with NorSA. Adjacent operations indicate opportunities for kernel fusion or weight merging. RMSNorm's learnable affine parameters are folded into subsequent linear layers to reduce norm calculations. For each linear projection, NorSA adaptively computes thresholds based on the norm of the incoming hidden state $\mathbf{h}$. The norm computation in the feed-forward network receives SwiGLU outputs directly in registers. Detailed kernel fusion is illustrated in Appendix A.

Sparse activation offers a path to acceleration. The key insight is that when any element $x_i = 0$, the corresponding weight column $\mathbf{w}_i$ neither needs to be loaded from memory nor participate in computation. This selective processing directly reduces memory bandwidth pressure and FLOPs.

To leverage this, a sparsification function $\text{Sparsify}(\cdot)$ is introduced to zero out selected elements of $\mathbf{x}$ based on their importance. The resulting sparse linear layer output is computed as:

$$\mathbf{y}' = \mathbf{W}\,\text{Sparsify}(\mathbf{x}) \tag{1}$$

3.2 Magnitude-Based and Top-k Activation Sparsification

Current activation sparsification methods primarily follow two paradigms. TEAL employs a magnitude-based approach where elements with absolute values below a fixed threshold τ are zeroed out:

$$[\text{Sparsify}_{\text{TEAL}}(\mathbf{x})]_i = \begin{cases} x_i & \text{if } |x_i| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This method is computationally efficient, requiring only element-wise comparisons. However, TEAL applies a static threshold across all hidden states, a design choice rooted in the assumption that activation dimensions are i.i.d. random variables. This rigidity prevents the model from adapting to the varying scales of different tokens.

La RoSA improves upon simple thresholding by implementing a Top-k selection strategy, considering all dimensions:

$$[\text{Sparsify}_{\text{La RoSA}}(\mathbf{x})]_i = \begin{cases} x_i & \text{if } |x_i| \in \text{Top-}k(|\mathbf{x}|) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

La RoSA adapts thresholds per token but incurs high costs from Top-k sorting. Moreover, it interprets rotation as a way to homogenize activations toward an i.i.d. Gaussian distribution via the Central Limit Theorem. In contrast, we study rotation from a covariance perspective and find empirically that its benefit is strongest when it reduces inter-dimensional correlation. This suggests that decorrelation and energy compaction, rather than Gaussianization alone, are the relevant effects for activation sparsification.

4 Method

4.1 Motivation

Existing activation sparsification approaches, such as TEAL and La RoSA, are often implicitly based on the assumption that values across different dimensions of activations can be treated as i.i.d. random variables. However, we question the validity of this assumption within the context of autoregressive decoding in Large Language Models (LLMs). Two key insights challenge this i.i.d. perspective:

First, **inter-token dependencies**. In autoregressive decoding, each token is conditioned on the preceding sequence via attention, so the hidden state at position t already encodes contextual history from tokens 1 to $t-1$. This induces token-wise scale variation and contextual magnitude shifts that contradict an i.i.d. view across the sequence.

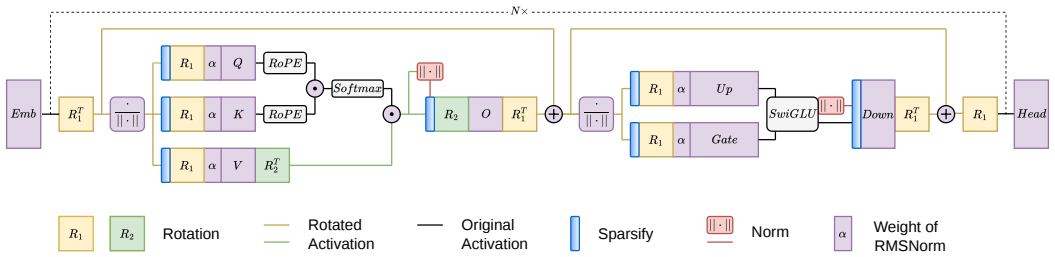

Figure 2: Sparsely activated transformer architecture incorporating rotation matrices. Rotation matrix R_1 is applied prior to Q/K/V and Up/Gate projections, while R_2 (structured as block-diagonal to accommodate multi-head attention) precedes the Output projection. By decorrelating hidden-state dimensions, these rotations enable more accurate sparsification decisions. In practical inference, most rotation operations can be fused into adjacent linear layers.

Second, **inter-dimension correlations in embeddings.** The input to the initial Transformer layer consists of word embeddings [Mikolov et al., 2013], which are deterministic, fixed constants during inference rather than stochastic samples. Crucially, these embeddings are highly structured and inter-dimensionally correlated to encode rich semantic information. Dimensions within an embedding vector are not independent; rather, they exhibit intrinsic directional and magnitude relationships where similar semantic concepts are clustered in specific sub-spaces. Such structured relationships are incompatible with the i.i.d. assumption.

Given the non-trivial correlations between tokens and dimensions, we propose to relax the i.i.d. assumption and develop a sparsification mechanism capable of capturing these dependencies. To ensure both effectiveness and practicality, our approach is guided by the following three criteria:

Contextual Awareness: Instead of treating each dimension in isolation, the function should consider the entire hidden state. Deciding whether to retain a dimension must be informed by the global context of the current token.

Computational Efficiency: The sparsification process must be lightweight. Since it sits directly in the inference path, any overhead here could negate the speed gains from subsequent sparse matrix multiplications.

Flexible Parameter Allocation: Inspired by speculative decoding [Leviathan et al., 2023], where a smaller draft model is capable of predicting certain tokens as accurately as the target model, we recognize that computational demands vary across tokens. The number of activated parameters should be dynamically allocated based on the complexity of each token, rather than using a fixed budget.

4.2 NorSA: Normalized Sparse Activation

To address the limitations of i.i.d.-based methods, we propose Normalized Sparse Activation (NorSA). The core philosophy of NorSA is to account for **inter-token dependencies** through token-wise scale variation, using the hidden-state scale as a lightweight contextual-magnitude signal. Since each hidden state already summarizes preceding context through attention, NorSA evaluates each dimension relative to the scale of the full activation vector rather than under a rigid absolute threshold.

This "relative evaluation" ensures that a dimension's significance is defined by its contribution relative to the scale of the full activation vector. Formally, for an activation vector $\mathbf{x} \in \mathbb{R}^d$, the sparsification function is:

$$[\text{Sparsify}_{\text{NorSA}}(\mathbf{x})]_i = \begin{cases} x_i & \text{if } |x_i| > \tau \cdot \|\mathbf{x}\|_2 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\|\mathbf{x}\|_2$ is the ℓ_2 norm and τ is a layer-specific threshold ratio. We determine τ such that the expected sparsity on a calibration distribution matches a target sparsity level $s^\star$:

$$\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\text{Sparsity}(\text{Sparsify}_{\text{NorSA}}(\mathbf{x}))\right] = s^\star \tag{5}$$

NorSA fulfills our design criteria through the following advantages:

Contextual Awareness: By scaling the threshold with $\|\mathbf{x}\|_2$, NorSA implements a "relative filter." This is crucial for handling varying hidden-state scales across different tokens. In high-magnitude vectors, the bar for retention is higher to filter out noise, whereas in low-magnitude vectors, the threshold automatically drops. This prevents "weak" but potentially essential hidden states from being entirely zeroed out, preserving the model's representational capacity.

Computational Efficiency: Computing a norm is significantly faster than the sorting required by Top-k methods. Moreover, the norm calculation can often be fused with existing Pre-Norm operations or activation functions in the FFN, incurring near-zero additional latency.

Flexible Parameter Allocation: Because the number of elements exceeding $\tau \cdot \|\mathbf{x}\|_2$ varies per token, NorSA naturally allocates more parameters to "complex" tokens with heavy-tailed distributions, focusing computational effort where it is most needed.

4.3 De-correlating Dimensions via Rotation

While NorSA incorporates global context, **inter-dimensional correlations** within hidden states complicate the estimation of dimension importance. Unlike prior approaches that view rotation as a means to homogenize distributions (i.i.d. Gaussian), we posit that correlations encode structured semantic relationships. When dimensions are coupled, a dimension's true contribution also depends on its interaction terms. To formalize this, consider the reconstruction error from zeroing a subset S:

$$\|\mathbf{y} - \mathbf{y}'\|_2^2 = \|\mathbf{W}(\mathbf{x} - \text{Sparsify}(\mathbf{x}))\|_2^2 = \underbrace{\sum_{i \in S} x_i^2 (\mathbf{W}^\top \mathbf{W})_{ii}}_{\text{Self-contribution}} + \underbrace{\sum_{i \in S, j \in S, j \neq i} x_i x_j (\mathbf{W}^\top \mathbf{W})_{ij}}_{\text{Cross-correlation}} \tag{6}$$

Ideally, a sparsification function evaluates a dimension's impact by considering both terms. However, while the self-contribution is computable in $O(d)$, the cross-correlation term typically requires $O(d^2)$ operations. For high-dimensional LLMs, these $O(d^2)$ interactions are too computationally demanding for real-time inference. Moreover, to further reduce overhead, we solely utilize input $\mathbf{x}$ for importance estimation while omitting the weight matrix $\mathbf{W}$.

To bridge this gap, we employ orthogonal rotation to make the $O(d)$ self-contribution a more reliable proxy for total error. By applying a pre-calculated orthogonal matrix $\mathbf{R}$ to reduce inter-dimensional correlations, we "diagonalize" the sparsification impact. Subsequent weights are transformed offline ($\mathbf{W}' = \mathbf{W}\mathbf{R}$) to maintain computational invariance.

Crucially, this rotation facilitates energy compaction; by decorrelating hidden states, it concentrates information into fewer dimensions. This renders the remaining dimensions statistically redundant, allowing them to be zeroed out with minimal fidelity loss.

4.3.1 Optimization Objective

We leverage the computational invariance of Transformer architectures [Ashkboos et al., 2024] to insert rotation matrices without altering the model's output (see Figure 2). Here $\mathbf{h}$ denotes the hidden state. Consistent with our decorrelation objective, we seek an orthogonal matrix $\mathbf{R}$ that minimizes the off-diagonal energy of the covariance matrix $\mathbf{C} = \mathbb{E}_{\mathbf{h} \sim p_{\text{data}}}[\mathbf{h}\mathbf{h}^\top]$:

$$\min_{\mathbf{R}} \sum_{i \neq j} (\mathbf{R}^\top \mathbf{C} \mathbf{R})_{ij}^2 \tag{7}$$

This objective is optimized using gradient descent adapted for orthogonal constraints [Chen et al., 2021b, Liu et al., 2025c, Gu et al., 2025].

To reduce overhead, we share rotation matrices across multiple projections where possible. For instance, a single matrix R_1 is applied to the inputs of both Q/K/V and Up/Gate projections. In such cases, $\mathbf{R}$ is optimized by minimizing the summed off-diagonal energy over the associated covariance matrices $\{\mathbf{C}^{(k)}\}$:

$$\min_{\mathbf{R}} \sum_k \sum_{i \neq j} (\mathbf{R}^\top \mathbf{C}^{(k)} \mathbf{R})_{ij}^2, \quad \mathbf{C}^{(k)} = \mathbb{E}_{\mathbf{h}^{(k)} \sim p_{\text{data}}}[\mathbf{h}^{(k)} (\mathbf{h}^{(k)})^\top] \tag{8}$$

Due to the multi-head attention structure, R_2 is constrained to a block-diagonal format, where each block corresponds to an individual query head. For models using Grouped Query Attention (GQA), query heads that share the same KV heads also share the same block in R_2 to maintain structural consistency.

4.3.2 Implementation via PCA Initialization

We find empirically that PCA initialization provides an efficient starting point for rotation optimization. Let H_1, H_2, and H_3 denote calibration activation matrices whose rows are sampled inputs to the Q/K/V, Output, and Up/Gate projections, respectively. We initialize R_1 from the eigenvectors of the composite Gram matrix $H_1^\top H_1 + H_3^\top H_3$, and initialize each block of R_2 from the corresponding block covariance of $H_2^\top H_2$. While PCA eigenvectors capture most performance gains, R can be further refined via Eq. 8, balancing optimization quality with calibration efficiency.

4.4 Hardware-Aware Implementation and Complexity Analysis

To translate theoretical sparsity into practical speedup, we implement hardware-optimized kernels using Triton [OpenAI, 2021], focusing on three key optimizations:

1. Sparse GEMV Kernel: Leveraging insights from [Liu et al., 2023, 2025a], we develop an efficient Sparse Matrix-Vector (SpMV) kernel. Weights are stored in column-major format for fast selective access based on activation indices. Furthermore, we fuse the sparsification, selective loading, and split-K multiplication into a single kernel, reducing memory bandwidth pressure and launch overhead.

2. Fused Operators: To minimize memory access, we fuse NorSA's norm computation with the SwiGLU activation kernel in the FFN. This enables thresholding and activation function to occur in a single pass over the hidden state.

3. Architectural Fusing: We refine the architecture for zero-cost thresholding by folding the RMSNorm's learnable affine parameters into the subsequent linear layer weights. Consequently, the input to the Q/K/V and Up/Gate projections becomes the normalized vector $\mathbf{x}/\|\mathbf{x}\|_2$, which has unit norm.

Complexity Analysis NorSA balances predictive performance with inference efficiency. Excluding the rotation matrices, its inference-time complexity is only marginally higher than TEAL because it requires a norm computation in addition to element-wise thresholding. This remains substantially cheaper than La RoSA, which requires sorting or partial sorting operations ($O(d \log d)$ or $O(d \log k)$) for every linear layer.

The inclusion of rotation matrices (e.g., R_1) introduces a space and time complexity of $O(d_{\text{model}}^2)$. While this adds approximately 7% more parameters, the measured decoding overhead is much smaller than this asymptotic worst case because the additional linear transforms are fused where possible and remain only one part of the overall decoding stack. Detailed kernel implementation and efficiency analysis are provided in Appendix A.

5 Experiment

Models and Baselines: We evaluate NorSA on LLaMA-2 [Touvron et al., 2023], LLaMA-3 [Grattafiori et al., 2024], Mistral v0.3 [Jiang et al., 2023], and Qwen-2.5 [Yang et al., 2024], using TEAL and La RoSA as training-free baselines. Following La RoSA, we sparsify all token activations during inference, and baseline numbers are quoted from their paper under aligned settings.

Calibration: We use 16 sequences of 2,048 tokens from the WikiText-2 [Merity et al., 2017] training set to determine NorSA's thresholds and rotations, TEAL's thresholds, and La RoSA's rotations.

Mixed Sparsity Configuration: We adopt TEAL's block-wise greedy optimization [Liu et al., 2025a] with a 1% global sparsity step. For LLaMA-3-8B, this process takes 26 minutes on an NVIDIA A100.

Rotation Matrices: For efficiency, we use PCA-initialized rotations without further refinement. As shown in 5.3.2, this is near-optimal. Computing them for LLaMA-3-8B takes about 8 minutes on an NVIDIA A100.

Table 1: Main results. We report zero-shot accuracy (Acc), 5-shot MMLU, and WikiText-2 perplexity (PPL); PPL is shown up to 50% sparsity.

		LLaMA-2						LLaMA-3						Qwen-2.5						Mistral		
		7B			70B			8B			70B			7B			72B			7B		
	Method	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓	Acc 0-shot	MMLU 5-shot	PPL ↓
0%	Dense	66.69	45.85	5.47	73.66	68.80	3.32	70.05	65.26	6.13	76.29	78.71	2.85	70.34	74.21	6.85	75.58	86.08	3.87	70.44	62.34	5.31
25%	TEAL	65.94	44.66	6.31	73.31	67.70	3.43	69.40	63.85	6.37	73.23	74.86	3.95	69.76	73.21	6.93	75.05	85.44	3.93	70.06	61.51	5.53
	NorSA w/o R	**66.54**	45.61	5.57	73.37	68.39	3.39	**69.91**	64.62	6.25	76.05	78.59	2.96	70.11	73.78	6.92	**75.59**	85.75	3.93	**70.60**	61.80	5.43
	La RoSA	66.39	45.66	**5.51**	73.38	**68.74**	**3.34**	69.54	64.85	6.23	**76.30**	78.13	2.94	**70.12**	73.74	6.90	75.53	85.62	3.92	70.25	61.81	5.34
	NorSA	66.52	**45.81**	5.54	**73.71**	68.56	3.36	**69.91**	**65.06**	**6.19**	76.28	**78.71**	**2.91**	69.97	**73.90**	**6.87**	75.57	**85.80**	**3.89**	70.49	**62.22**	**5.33**
40%	TEAL	64.92	43.46	6.40	72.47	66.78	3.61	68.14	59.84	6.83	72.24	73.23	4.45	68.61	71.44	7.20	74.65	84.80	4.07	68.76	60.17	5.98
	NorSA w/o R	65.39	43.33	5.87	73.34	67.54	3.50	68.78	62.29	6.65	73.92	77.63	3.49	69.22	72.65	7.15	75.44	85.18	4.10	69.72	60.39	5.94
	La RoSA	66.15	44.66	5.64	73.31	68.16	3.44	68.79	62.61	6.60	75.41	77.62	3.37	69.67	72.33	7.10	**75.68**	85.53	4.06	69.44	61.15	5.44
	NorSA	**66.38**	**45.42**	**5.59**	**73.64**	**68.22**	**3.40**	**69.71**	**64.07**	**6.33**	**75.85**	**78.29**	**3.04**	**69.72**	**73.35**	**6.97**	75.65	**85.83**	**3.96**	**70.29**	**61.23**	**5.38**
50%	TEAL	63.22	39.57	6.80	71.92	64.43	3.91	64.92	52.78	7.56	70.80	69.20	5.61	67.76	68.53	7.81	73.74	83.54	4.31	66.73	57.34	7.17
	NorSA w/o R	64.09	40.33	6.41	72.77	66.28	3.72	67.02	58.95	7.32	74.09	76.61	4.32	68.47	70.29	7.56	74.72	84.95	4.28	69.08	58.19	6.23
	La RoSA	64.61	43.10	5.87	72.86	67.57	3.62	67.19	58.65	7.22	73.81	76.51	4.10	69.09	70.09	7.42	75.18	84.34	4.26	68.46	58.80	5.62
	NorSA	**65.49**	**43.25**	**5.72**	**73.31**	**67.73**	**3.47**	**68.82**	**62.12**	**6.57**	**75.24**	**78.15**	**3.33**	**69.26**	**72.28**	**7.13**	**75.45**	**85.66**	**4.07**	**69.84**	**60.41**	**5.46**

Evaluation: We report perplexity (PPL) on WikiText-2 and downstream accuracy using lm-evaluation-harness [Gao et al., 2024]: zero-shot ARC-Easy/Challenge [Clark et al., 2018], PIQA [Bisk et al., 2020], BoolQ [Clark et al., 2019], HellaSwag [Zellers et al., 2019], OpenBookQA [Mihaylov et al., 2018], and WinoGrande [Sakaguchi et al., 2020], plus 5-shot MMLU [Hendrycks et al., 2021].

5.1 Main Result

Downstream Task Performance. Table 1 shows that NorSA consistently matches or exceeds baseline downstream performance across architectures and sparsity levels. On the challenging 5-shot MMLU benchmark, its advantage is especially clear: at 50% sparsity on LLaMA-3-8B, NorSA retains 62.12% accuracy, nearly 3.5 points above La RoSA and far above TEAL's 52.78%. This suggests that context-aware thresholding better preserves the reasoning-critical features needed for MMLU.

Language Modeling Performance. Table 1 also reports WikiText-2 perplexity. NorSA achieves the lowest or near-lowest PPL across almost all tested models, with larger gains at higher sparsity. At 25% sparsity, its PPL is nearly identical to the dense baseline; at 50%, it remains stable while TEAL and La RoSA degrade substantially. On Mistral-7B, NorSA reaches 5.46 PPL, versus 5.62 for La RoSA and 7.17 for TEAL, indicating that modeling both inter-dimensional correlation and token-level context better preserves the data distribution.

5.2 Ablation Study: Calibration Data Sensitivity

We evaluate sensitivity to calibration data on LLaMA-2-7B and LLaMA-3-8B using WikiText-2 [Merity et al., 2017], C4 [Raffel et al., 2020], and Alpaca [Taori et al., 2023], each with 16 sequences of 2,048 tokens.

Table 2 shows that NorSA is generally robust to the calibration set, while Alpaca performs best at high sparsity. At 60%, Alpaca reaches 61.78% MMLU on LLaMA-3-8B, compared with 57.96% for WikiText-2 and 58.37% for C4.

We attribute this to distributional alignment: instruction-style data may better match the output distribution induced by LLM decoding, making it a more effective calibration source.

Table 2: Effect of calibration data on NorSA. PPL denotes WikiText-2 perplexity.

		LLaMA-2 7B			LLaMA-3 8B		
Sparsity	**Dataset**	PPL ↓	0-shot ↑	MMLU ↑	PPL ↓	0-shot ↑	MMLU ↑
0%	Dense	5.47	66.69	45.85	6.13	70.05	65.26
25%	WikiText-2	5.54	**66.52**	**45.81**	6.19	69.91	65.06
	C4	5.54	66.49	45.62	6.20	**70.06**	**65.22**
	Alpaca	**5.53**	66.43	45.61	**6.17**	69.93	65.05
40%	WikiText-2	5.59	**66.38**	45.42	6.33	69.71	64.07
	C4	5.62	65.85	44.53	6.36	69.68	64.33
	Alpaca	**5.56**	66.29	**45.44**	**6.25**	**69.78**	**64.68**
50%	WikiText-2	5.72	65.49	43.25	6.57	68.82	62.12
	C4	5.78	65.62	42.88	6.61	69.20	63.09
	Alpaca	**5.63**	**66.12**	**44.71**	**6.39**	**69.88**	**63.94**
60%	WikiText-2	6.02	63.88	39.18	7.16	67.33	57.96
	C4	6.14	63.31	39.47	7.20	68.01	58.37
	Alpaca	**5.78**	**65.00**	**43.37**	**6.73**	**68.88**	**61.78**

5.3 Ablation Study: Dissecting the NorSA Framework

To clarify NorSA's gains, we decompose it into two components: norm-based contextual thresholding and de-correlating rotation.

5.3.1 Impact of Contextual Sparsification

NorSA's core innovation is contextual awareness, which scales the pruning threshold with each hidden-state norm. We isolate this effect via NorSA w/o R, which uses contextual sparsification without rotation.

Table 1 shows that NorSA w/o R consistently outperforms TEAL. This gap likely stems from activation outliers [Dettmers et al., 2022]: the inputs to Output and Down projections are not normalized, so their magnitudes vary substantially across tokens and make static thresholds unreliable.

From a vector perspective, a dimension's significance is relative to the norm of its hidden state: a value such as 0.1 may be noise in a large-norm outlier state but a critical feature in a small-norm one. A fixed threshold therefore over-prunes small-norm states and under-prunes large-norm ones. By using relative thresholding, NorSA preserves vector direction and retains the most important components in both cases.

5.3.2 Impact of De-correlating Rotation

The de-correlating rotation is designed to mitigate inter-dimensional correlations within each token's hidden state. To isolate its contribution, we compare the full NorSA framework against NorSA w/o R and baselines.

Macro Performance: Table 1 shows that decorrelating rotation improves performance across nearly all evaluated models and sparsity levels. At 50% sparsity on LLaMA-3-8B, adding rotation raises MMLU from 58.95% to 62.12% and reduces perplexity from 7.32 to 6.57.

Micro-Mechanism via SLCR Analysis: To quantify de-correlation, we define the Self-Linear-Correlation Ratio (SLCR). Given a hidden-state vector $\mathbf{h} \in \mathbb{R}^d$ with covariance matrix $\mathbf{C} = \mathbb{E}[\mathbf{h}\mathbf{h}^\top] \in \mathbb{R}^{d\times d}$, the SLCR is calculated as:

$$\text{SLCR} = \frac{\sum_{i=1}^{d} C_{ii}^2}{\sum_{i=1}^{d}\sum_{j=1}^{d} C_{ij}^2} \tag{9}$$

This metric captures the diagonalization discussed around Eq. 6: as SLCR approaches 1, more covariance energy is concentrated on the diagonal entries C_{ii} rather than the off-diagonal couplings C_{ij} for $i \neq j$. To validate the empirical link between correlation reduction and performance, we vary the optimization steps and include a reverse-optimization setting (Step -100) that intentionally increases correlation as a negative control.

Table 3 shows that performance improves monotonically with SLCR. The reverse setting (Step -100) performs worse than Identity (Step 0), showing that rotation is helpful only when it reduces correlation. Increasing the SLCR of h_1 to 97.5% (PCA Step 100) lowers perplexity to 5.707 and raises MMLU to 43.26%, supporting our view that a more diagonal covariance makes NorSA's lightweight importance proxy more reliable.

Table 3: Correlation-reduction ablation on LLaMA-2-7B at 50% sparsity. *Step* denotes optimization steps; h_1, h_2, and h_3 are the SLCRs of the Q/K/V, Output, and Up/Gate inputs.

Initialization	**Step**	h_1	h_2	h_3	PPL ↓	0-shot ↑	MMLU ↑
Identity	-100	1.1%	3.5%	0.8%	6.657	63.46	37.98
	0	22.3%	8.5%	26.9%	6.408	64.09	40.33
	10	44.8%	9.9%	52.6%	6.181	64.39	40.68
	50	92.8%	22.3%	95.7%	6.056	63.98	41.42
	100	94.3%	25.2%	97.0%	5.964	64.30	42.56
PCA	0	97.4%	**27.1%**	**98.1%**	5.714	65.49	43.25
	100	**97.5%**	**27.1%**	**98.1%**	**5.707**	**65.70**	**43.26**

Optimization and Initialization: We compare Identity initialization against PCA initialization, optimizing both with a QR-based reparameterization gradient descent optimizer [Gu et al., 2025] at a constant learning rate of 1e-4.

Table 3 further shows that PCA is a near-optimal starting point. At Step 0, it already achieves better SLCR and perplexity (5.714) than Identity after 100 optimization steps, justifying our use of PCA-derived eigenvectors for efficient offline calibration.

5.4 Inference Speedup

Sparse GEMV Benchmarks: We benchmark our Triton-based sparse GEMV kernels on an NVIDIA A100 GPU (Table 4). These kernels use CUDA graphs to reduce launch overhead, and the matrix dimensions match typical LLaMA weight shapes. This isolates the operator-level effect of sparsity before it is embedded into full autoregressive decoding.

Results show consistent speedups that grow with sparsity, especially for larger matrices. We also include the 100% sparsity case to expose the kernel's upper bound and intrinsic overhead. The monotonic trend is important: it indicates that the kernel benefit is not tied to a narrow operating point, but persists as memory traffic is progressively reduced.

Table 4: Sparse GEMV latency and speedup versus `torch.matmul` on NVIDIA A100.

Sparsity	**Matrix Dimensions** (d_{out}, d_{in})											
	(4096, 4096)		(14336, 4096)		(4096, 14336)		(8192, 8192)		(28672, 8192)		(8192, 28672)	
	μs	Speedup	μs	Speedup	μs	Speedup	μs	Speedup	μs	Speedup	μs	Speedup
Dense	41	1.00×	101	1.00×	92	1.00×	105	1.00×	299	1.00×	292	1.00×
0%	41	1.00×	97	1.04×	92	1.00×	106	0.99×	294	1.02×	291	1.00×
25%	33	1.23×	78	1.29×	74	1.23×	85	1.23×	242	1.23×	235	1.24×
50%	32	1.28×	64	1.58×	63	1.45×	70	1.49×	181	1.65×	188	1.55×
75%	31	1.31×	53	1.90×	53	1.73×	58	1.82×	144	2.07×	147	1.98×
100%	29	1.40×	39	2.59×	40	2.28×	42	2.51×	91	3.26×	97	3.01×

End-to-End Decoding Speedup: Table 5 reports `gpt-fast` [Meta, 2023] throughput with batch size 1, CUDA Graph, and `torch.compile` for 200-token generation. Although 0% sparsity incurs a 7–10% latency penalty from rotation and norm computation, NorSA delivers substantial gains as sparsity increases: at 50%, throughput improves by 1.32× for 7/8B models and over 1.42× for 70B models. The gains are already visible at 25% sparsity and widen further at 75%, showing that the additional normalization and rotation overhead is amortized once even moderate activation sparsity is reached. This confirms that reducing weight loads alleviates memory-bandwidth bottlenecks, especially in larger models, and that activation sparsity can translate into real latency reduction in this sequential decoding setting.

Table 5: End-to-end decoding throughput and speedup using `gpt-fast` on NVIDIA A100. 7B/8B models use one GPU; 70B models use tensor parallelism over two GPUs.

Sparsity	**LLaMA-2-7B**		**LLaMA-2-70B**		**LLaMA-3-8B**		**LLaMA-3-70B**	
	token/s	speedup	token/s	speedup	token/s	speedup	token/s	speedup
Dense	106.45	1.00×	22.18	1.00×	96.37	1.00×	21.66	1.00×
0%	95.89	0.90×	20.75	0.94×	87.45	0.91×	20.14	0.93×
25%	115.06	1.08×	26.03	1.17×	104.14	1.08×	25.37	1.17×
50%	140.93	1.32×	31.66	1.43×	127.12	1.32×	30.80	1.42×
75%	160.02	1.50×	38.35	1.73×	142.09	1.47×	36.72	1.69×

6 Conclusion and Limitation

We presented Normalized Sparse Activation (NorSA), a training-free framework for accelerating LLM decoding. Unlike prior methods based on uniform sparsity assumptions, NorSA explicitly models hidden-state correlation through norm-based contextual thresholding and decorrelating rotation. Together, these components preserve critical information across tokens and provide a more informative sparsification signal. Our results show that NorSA consistently improves quality retention at a given sparsity level and yields substantial decoding speedups in our end-to-end evaluation. These findings support covariance-aware sparsification as a practical direction for training-free LLM acceleration. More broadly, our results suggest that effective activation sparsification depends on modeling hidden-state statistics beyond raw magnitude, especially contextual scale variation and inter-dimensional correlation. A natural direction for future work is to extend this covariance-aware perspective to more fine-grained and batch-aware sparsification strategies.

A limitation of sparse activation is that its benefit is most direct in small-batch decoding: under larger-batch serving, the activated dimensions of different tokens are often dispersed, and their union can approach the full hidden dimension, leaving few shared inactive dimensions to skip.

Acknowledgments and Disclosure of Funding

This work was supported in part by China NSFC project under Grants No. 62631016, No. 62601524 and No. U25A20409, and in part by SJTU Med-X (Medicine Engineering) Translational Research Grant YG2025LC09.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Systematic outliers in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=rLX7Vyyzus.

Anthropic. Introducing claude opus 4.7, April 2026. URL https://www.anthropic.com/news/claude-opus-4-7.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vXxardq6db.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021a.

Shixiang Chen, Alfredo Garcia, Mingyi Hong, and Shahin Shahrampour. Decentralized riemannian gradient descent on the stiefel manifold. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1594–1605. PMLR, 18–24 Jul 2021b. URL https://proceedings.mlr.press/v139/chen21g.html.

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 Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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:1803.05457v1*, 2018.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=dXiGWqBoxaD.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training compression for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

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

Google. A new era of intelligence with gemini 3, November 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Tianteng Gu, Bei Liu, Bo Xiao, Ke Zeng, Jiacheng Liu, and Yanmin Qian. Denoiserotator: Enhance pruning robustness for LLMs via importance concentration. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9e2H9DhKPa.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. URL https://arxiv.org/abs/1510.00149.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. CATS: Context-aware thresholding for sparsity in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=v3w2a7EInO.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J. Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, and Sanjiv Kumar. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TJ2nxciYCk-.

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 LLM compression and acceleration. In *MLSys*, 2024.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=dGVZwyq5tV.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and lulu hu. La RoSA: Enhancing LLM efficiency via layerwise rotated sparse activation. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=1b6NNpFYI4.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: LLM quantization with learned rotations. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=ogO6DGE6FZ.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. Deja vu: Contextual sparsity for efficient LLMs at inference time. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/liu23am.html.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Meta. gpt-fast. https://github.com/meta-pytorch/gpt-fast, 2023.

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 *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://api.semanticscholar.org/CorpusID:52183757.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. URL https://api.semanticscholar.org/CorpusID:5959482.

Seyed Iman Mirzadeh, Keivan Alizadeh-Vahid, Sachin Mehta, Carlo C del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. ReLU strikes back: Exploiting activation sparsity in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=osoWxY8q2E.

OpenAI. Introducing triton: Open-source gpu programming for neural networks. Technical report, 2021. URL https://github.com/triton-lang/triton.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL https://jmlr.org/papers/v21/20-074.html.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. ProSparse: Introducing and enhancing intrinsic activation sparsity within large language models. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2626–2644, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.180/.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP$\#$: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=9BrydUVcoe.

An Yang, Baosong Yang, Beichen Zhang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472. URL https://doi.org/10.18653/v1/p19-1472.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. ReLU2 wins: Discovering efficient activation functions for sparse LLMs. *ArXiv*, abs/2402.03804, 2024. URL https://api.semanticscholar.org/CorpusID:267499856.

A Hardware aware customized kernel

A.1 Fused Swiglu and Squared Norm

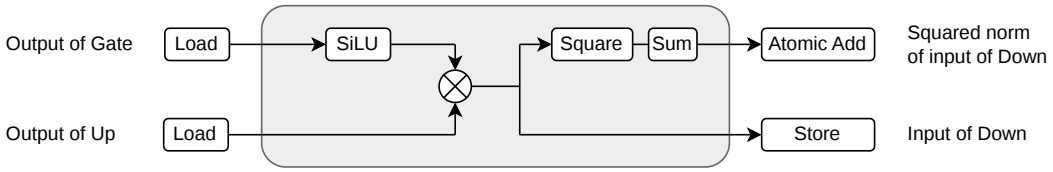

Figure 3: Execution flow diagram of the fused SwiGLU and norm kernel. Gray-shaded areas represent operations performed in registers/cache memory, minimizing data movement and enhancing computational efficiency.

```python
import torch
import triton
import triton.language as tl
from typing import Optional, Tuple, Union

@triton.jit
def silu(x):
    """SiLU activation function: x * sigmoid(x)"""
    return x * tl.sigmoid(x)

norm_configs=[
    # ...
    triton.Config({"BLOCK_SIZE": 256}, num_warps=2),
    # ...
]
@triton.autotune(
    configs=norm_configs,
    key=["hidden_dim"]  # N=input_dim
)
@triton.jit
def swiglu_norm_kernel(
    a_ptr,                      # First input tensor pointer
    b_ptr,                      # Second input tensor pointer
    c_ptr,                      # Output tensor pointer
    norm_squared_ptr,           # Pointer to store squared norm
    hidden_dim,                 # Full hidden dimension size
    BLOCK_SIZE: tl.constexpr    # Block size for processing
):
    # Get block index
    block_idx = tl.program_id(0)

    # Calculate start offset for this block
    start_offset = block_idx * BLOCK_SIZE

    # Create offsets and mask for this block
    offsets = start_offset + tl.arange(0, BLOCK_SIZE)
    mask = offsets < hidden_dim

    # Load input values
    a_vals = tl.load(a_ptr + offsets, mask=mask, other=0).to(tl.float32)
    b_vals = tl.load(b_ptr + offsets, mask=mask, other=0)

    # Compute SwiGLU activation: SiLU(a) * b
    silu_a = silu(a_vals)
    c_vals = silu_a.to(b_vals.dtype) * b_vals

    # Store activated values
    tl.store(c_ptr + offsets, c_vals, mask=mask)

    # Compute squared values
    c_squared = c_vals.to(tl.float32) * c_vals.to(tl.float32)
```

```
    # Sum squared values for this block
    block_sum = tl.sum(c_squared, axis=0)

    # Atomic add to global norm squared
    tl.atomic_add(norm_squared_ptr, block_sum)


def swiglu_norm(a: torch.Tensor, b: torch.Tensor) -> Tuple[torch.Tensor, torch.Tensor]:
    """
    fused SwiGLU with squared norm calculation for (1,1,hidden_dim) shaped inputs

    Args:
        a: First input tensor of shape (1,1,hidden_dim)
        b: Second input tensor of shape (1,1,hidden_dim)

    Returns:
        Tuple of (activated_tensor, norm_squared)
    """
    # Verify input shapes
    assert a.shape == b.shape, f"Input shapes must match: {a.shape} vs {b.shape}"
    assert a.shape[0] == 1 and a.shape[1] == 1, f"Input must have shape (1,1,hidden_dim), got {a.shape}"

    # Get hidden dimension
    hidden_dim = a.shape[2]

    # Create output tensor
    c = torch.empty_like(a)

    # Create tensor to store squared norm
    norm_squared = torch.zeros(1, dtype=torch.float32, device=a.device)

    grid = lambda META: (
            triton.cdiv(hidden_dim, META["BLOCK_SIZE"]),
    )

    # Launch kernel with one block per BLOCK_SIZE elements
    swiglu_norm_kernel[grid](
        a,
        b,
        c,
        norm_squared,
        hidden_dim=hidden_dim,
    )
    return c, norm_squared
```

A.2 Sparse Matrix-Vector Multiply (GEMV) Kernel

Our sparse GEMV implementation is optimized for GPU execution using Triton. The kernel design leverages several key insights to maximize performance while implementing NorSA's sparsification.

Figure 4 illustrates the execution flow of our sparse GEMV kernel, highlighting how operations performed in registers or cache memory (gray-shaded areas) minimize global memory accesses. The weights are selectively loaded based on the sparsification mask, significantly reducing memory bandwidth requirements and computational overhead.

The kernel processes input vector X and weight matrix W to produce output vector Y, where W is stored in column-major format to facilitate selective column loading based on the sparsification mask. Each thread block processes a segment of X and computes partial results for a segment of Y. The final output for each segment of Y is the sum of contributions from all thread blocks processing the same output segment.

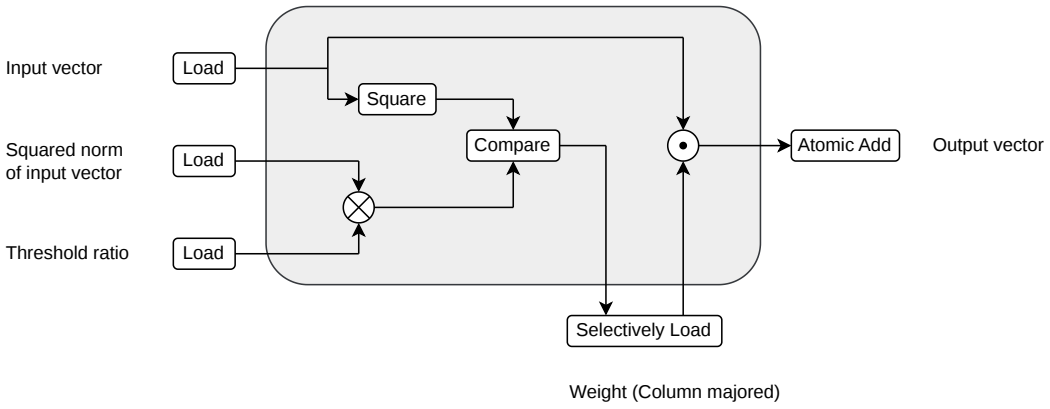

Figure 4: Execution flow diagram of the sparse GEMV kernel. Gray-shaded areas represent operations performed in registers/cache memory. Weights are stored in column-major format and selectively loaded based on the sparsification mask, reducing memory bandwidth requirements and computational overhead. In our implementation, we use squared norm values and compare against squared element values, which is mathematically equivalent to the formulation described in the Method section.

Our implementation computes the squared norm of the input vector and compares each element's squared value against the threshold (proportional to the squared norm), which is mathematically equivalent to the formulation in our method section but computationally more efficient. This approach requires only a single pass through the data with no sorting or additional memory accesses.

Through extensive autotuning, we determined that the optimal configuration uses block dimensions of BLOCK_M = 128 and BLOCK_N = 64 with 4 warps and 2 stages on NVIDIA A100 GPU. The thread blocks can execute completely in parallel, with the latency of parallel block-wise matrix-vector multiplication being $O(\text{BLOCK_M} \cdot \text{BLOCK_N})$.

Additionally, NorSA requires computing the squared norm for the input to the O projection. This calculation has a latency of $O(\log \text{Norm_BLOCK_N})$ in parallel computing scenarios due to the reduction operation required within thread blocks.

Unlike top-k sparsification that requires sorting operations, our NorSA approach incorporates contextual information and considers all dimension values without the need for sorting. While our block-wise matrix multiplication has a practical latency of $O(\text{BLOCK_N} \cdot \text{BLOCK_M})$ (e.g., $O(128 \cdot 64)$), sorting would introduce substantial latency of $O((\log N)^2)$ in parallel scenarios, where N is the full dimension of the input vector (e.g., $O((\log 14336)^2)$). This theoretical complexity disadvantage is further exacerbated in practice because dimensions like the 14,336-dimensional input of the Down projection in LLaMA-2-7B exceed the processing capacity of a single thread block, necessitating inter-block communication/synchronization and significantly increasing latency.

```python
import torch
import triton
import triton.language as tl
from typing import Optional
import os
os.environ['TRITON_PRINT_AUTOTUNING'] = '1'

def init_to_zero(*names):
    def init_func(nargs):
        for name in names:
            nargs[name].zero_()
    return init_func

norm_configs=[
    # ...
    triton.Config({"Norm_BLOCK_N": 256}, num_warps=2, pre_hook=init_to_zero("Y")),
    # ...
]
```

```
@triton.autotune(
    configs=norm_configs,
    key=["N"]  # N=input_dim
)
@triton.jit
def norm_squared_kernel(
    Y,  # Output pointer (1 x 1)
    X,  # Input pointer (1 x 1 x N)
    N,  # N=input_dim
    Norm_BLOCK_N: tl.constexpr,  # Block size for input_dim (N)
):
    # Block indices
    start_n = tl.program_id(0)
    input_indices = start_n * Norm_BLOCK_N + tl.arange(0, Norm_BLOCK_N)
    X_ptr = X + input_indices
    Y_ptr = Y

    # Load input and create sparsity mask
    input_mask = input_indices < N

    x = tl.load(X_ptr, mask=input_mask, eviction_policy='evict_last')
    acc = tl.sum(x.to(tl.float32) * x.to(tl.float32))
    tl.atomic_add(Y_ptr, acc)


configs=[
    # ...
    triton.Config({"BLOCK_M": 128, "BLOCK_N": 64}, num_warps=4, pre_hook=
    init_to_zero("Y")),
    # ...
]

@triton.autotune(
    configs=configs,
    key=["N", "M", "threshold_ratio"]  # N=input_dim, M=out_dim
)
@triton.jit
def sparse_gemv_kernel(
    Y,  # Output pointer (1 x 1 x M)
    A,  # Weight pointer (M x N, column-major)
    X,  # Input pointer (1 x 1 x N)
    threshold_ratio,  # Threshold for sparsity
    norm_squared_ptr,     # Squared norm of X
    N, M,             # N=input_dim, M=out_dim
    BLOCK_N: tl.constexpr,  # Block size for input_dim (N)
    BLOCK_M: tl.constexpr,  # Block size for out_dim (M)
):
    # Block indices
    start_n = tl.program_id(0)
    start_m = tl.program_id(1)
    input_indices = start_n * BLOCK_N + tl.arange(0, BLOCK_N)
    output_indices = start_m * BLOCK_M + tl.arange(0, BLOCK_M)

    # A is column major: A[output][input] = A + input * M + output
    A_ptr = A + (input_indices[None, :] * M + output_indices[:, None])
    X_ptr = X + input_indices
    Y_ptr = Y + output_indices

    # Load input and create sparsity mask
    input_mask = input_indices < N
    output_mask = output_indices < M

    norm_squared = tl.load(norm_squared_ptr)

    x = tl.load(X_ptr, mask=input_mask, eviction_policy='evict_last')
    x_squared = x * x
    sparse_mask = (x_squared > threshold_ratio * norm_squared) & input_mask

```

```
    # Load weights using sparsity mask
    a = tl.load(A_ptr, mask=output_mask[:, None] & sparse_mask[None, :],
                other=0.0, eviction_policy='evict_first')

    # Compute sparse matrix-vector product
    acc = tl.sum(a.to(tl.float32) * x.to(tl.float32)[None, :], axis=1)

    # Atomic add to output
    tl.atomic_add(Y_ptr, acc, mask=output_mask)


def sparse_gemv(
    x: torch.Tensor,
    weight: torch.Tensor,
    threshold_ratio: float,
    norm_squared: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    """
    Compute y = weight @ sparse(x), where weight is stored in column-major
    format.

    Args:
        x: Input tensor [1, 1, input_dim]
        weight: Weight matrix [out_dim, input_dim] (column-major)
        threshold_ratio: Sparsity threshold ratio
        norm_squared: Optional, squared norm of input vector. If None, it
    will be computed

    Returns:
        output: Result tensor [1, 1, out_dim]
    """
    M, N = weight.shape  # M=out_dim, N=input_dim
    batch_size, seq_len, input_dim = x.shape

    input_vec = x.contiguous()

    if weight.stride(0) != 1:
        raise ValueError("Weight matrix must be column-major (first dimension
     contiguous)")

    # Compute squared norm (if not provided)
    if norm_squared is None:
        norm_squared = torch.zeros(1, device=x.device, dtype=torch.float32)
        grid = lambda META: (
            triton.cdiv(N, META["Norm_BLOCK_N"]),
        )
        norm_squared_kernel[grid](
            norm_squared,
            input_vec,
            N
        )

    # Create output tensor
    output_vec = torch.zeros(
        1, 1, M,  # Output dimension is the row count of weight matrix
        device=x.device,
        dtype=torch.float32,
    )

    # Define grid size
    grid = lambda META: (
        triton.cdiv(N, META["BLOCK_N"]),  # Number of blocks in input
    dimension
        triton.cdiv(M, META["BLOCK_M"]),  # Number of blocks in output
    dimension
    )

    # Call the optimized kernel
```

```
    sparse_gemv_kernel[grid](
        output_vec,         # Output pointer
        weight,             # Weight matrix (column-major)
        input_vec,          # Input vector
        threshold_ratio,    # Sparsity threshold ratio
        norm_squared,       # Squared norm of input vector
        N,                  # Input dimension
        M,                  # Output dimension
    )

    output_vec = output_vec.to(dtype=x.dtype)

    return output_vec
```

B Additional Evaluations on MoE and Long Contexts

B.1 Validation on MoE Models

MoE and NorSA sparsify computation at different granularities. MoE performs token-to-expert routing, while NorSA removes redundant dimensions within the activated experts. This makes the two mechanisms complementary rather than conflicting. To verify this, we evaluate NorSA on the MoE model Qwen3-30B-A3B [Yang et al., 2025] and apply it to the linear layers inside the activated experts.

Table 6: MoE results on Qwen3-30B-A3B. NorSA is applied at 50% sparsity inside the activated experts.

Metric	Dense	NorSA (50%)
WikiText-2 PPL ↓	8.705	8.760
MMLU (5-shot) ↑	79.62	78.48
GSM8K strict EM ↑	89.61	89.01

These results confirm that NorSA's dimension-level sparsity is orthogonal to MoE's expert-level sparsity. They support the view that expert routing provides macro-level sparsity, while NorSA adds a complementary micro-level sparsification mechanism within the activated computation path.

B.2 Long-Context Performance

We further examine whether calibration performed on 2,048-token sequences transfers to substantially longer contexts. Using the same precomputed PCA rotation matrices without recalibration, we evaluate Qwen3-4B-Thinking-2507 [Yang et al., 2025] on the LongBench v1 dataset [Bai et al., 2024] together with reasoning [Cobbe et al., 2021] and coding tasks [Chen et al., 2021a], with contexts up to 32K tokens. Table 7 summarizes the main results.

Table 7: Reasoning, code, and long-context results on Qwen3-4B-Thinking-2507, evaluated without recalibrating the PCA matrices.

Metric	Category	Dense	NorSA (40%)	NorSA (50%)	NorSA (60%)
GSM8K strict EM ↑	Math	87.26	85.90	86.13	85.29
HumanEval pass@1 ↑	Code	62.80	62.80	61.63	57.93
LongBench: Retrieval (En) ↑	Long-context	86.13	85.79	81.38	82.17

While high sparsity incurs a trade-off on retrieval and coding tasks, NorSA preserves most reasoning capabilities and long-range attention at moderate sparsity. Under 40% sparsity, GSM8K remains at 85.90%, HumanEval stays unchanged at 62.80%, and English passage retrieval is 85.79% versus 86.13% for the dense model. These results indicate that the same precomputed rotation matrices still transfer to extended contexts without recalibration.

C Detailed experiment result

Table 8: Performance comparison of different sparsification methods on Mistral-7B-v0.3.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Dense	73.80	82.31	44.00	80.40	82.11	78.40	52.04	70.44	62.34
25%	TEAL	72.61	81.82	44.40	80.05	81.44	78.24	51.86	70.06	61.51
	NorSA w/o R	**74.27**	81.66	**45.20**	80.17	**82.29**	78.32	**52.30**	**70.60**	61.80
	La RoSA	72.85	**82.15**	44.40	80.14	81.71	**78.49**	52.05	70.25	61.81
	NorSA	73.88	**82.15**	44.20	**80.29**	**82.29**	78.45	52.22	70.49	**62.22**
40%	TEAL	69.55	80.36	44.40	79.08	80.49	77.14	50.30	68.76	60.17
	NorSA w/o R	70.48	**81.83**	44.00	79.85	**82.02**	77.74	52.13	69.72	60.39
	La RoSA	70.88	81.07	43.40	79.63	81.68	**78.32**	51.11	69.44	61.15
	NorSA	**72.77**	81.77	**44.80**	**80.16**	81.56	78.16	**52.82**	**70.29**	**61.23**
50%	TEAL	68.71	79.52	40.40	76.97	79.55	74.34	47.63	66.73	57.34
	NorSA w/o R	70.24	81.12	44.60	78.49	**81.59**	77.44	50.09	69.08	58.19
	La RoSA	69.93	**81.61**	43.20	78.93	80.92	75.55	49.06	68.46	58.80
	NorSA	**70.56**	**81.61**	**45.00**	**80.03**	81.56	**78.07**	**52.05**	**69.84**	**60.41**
60%	NorSA w/o R	67.32	79.49	39.40	76.79	80.28	74.54	47.18	66.42	54.05
	NorSA	**71.35**	**81.07**	**43.60**	**79.00**	**80.83**	**77.31**	**49.66**	**68.97**	**58.63**

Table 9: Performance comparison of different sparsification methods on LLaMA-3-8B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Dense	73.64	80.41	44.80	79.13	81.10	77.78	53.50	70.05	65.26
25%	TEAL	72.48	80.36	**45.40**	78.45	80.07	77.08	52.00	69.40	63.85
	NorSA w/o R	72.38	**80.69**	44.80	78.83	**81.56**	**77.69**	**53.41**	**69.91**	64.62
	La RoSA	**72.93**	80.63	45.00	**79.09**	80.40	76.68	52.05	69.54	64.85
	NorSA	**72.93**	80.52	45.00	79.03	81.22	77.53	53.16	**69.91**	**65.06**
40%	TEAL	70.34	78.51	**45.80**	76.55	79.57	76.47	49.74	68.14	59.84
	NorSA w/o R	71.11	79.71	43.00	77.68	**81.10**	77.53	51.37	68.78	62.29
	La RoSA	71.11	79.87	44.00	77.85	80.09	76.98	51.62	68.79	62.61
	NorSA	**73.01**	**81.07**	44.80	**78.56**	80.43	**77.74**	**52.39**	**69.71**	**64.07**
50%	TEAL	66.71	77.09	42.80	72.61	76.26	72.71	46.24	64.92	52.78
	NorSA w/o R	69.69	77.69	43.20	75.67	79.24	74.75	48.89	67.02	58.95
	La RoSA	68.98	79.98	44.20	76.85	77.55	74.75	48.04	67.19	58.65
	NorSA	**72.14**	**80.09**	**44.40**	**77.70**	**79.60**	**76.60**	**51.19**	**68.82**	**62.12**
60%	NorSA w/o R	65.35	76.33	40.20	70.58	76.36	70.88	43.60	63.32	50.29
	NorSA	**70.80**	**78.73**	**43.40**	**75.84**	**78.26**	**75.63**	**48.63**	**67.33**	**57.96**

Table 10: Performance comparison of different sparsification methods on LLaMA-3-70B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Dense	80.11	84.49	48.80	84.97	85.22	86.15	64.33	76.29	78.71
25%	TEAL	76.63	82.31	48.20	81.82	83.24	81.64	58.78	73.23	74.86
	NorSA w/o R	79.79	84.33	48.20	84.92	85.23	85.98	63.91	76.05	78.59
	La RoSA	**81.45**	84.39	**49.00**	**85.07**	84.89	85.65	63.65	**76.30**	78.13
	NorSA	80.27	**84.66**	48.40	84.97	**85.26**	**86.15**	**64.25**	76.28	**78.71**
40%	TEAL	76.32	81.93	45.80	81.56	80.64	81.90	57.50	72.24	73.23
	NorSA w/o R	79.48	82.64	48.00	83.74	84.68	80.56	58.36	73.92	77.63
	La RoSA	**79.72**	83.62	**48.40**	84.71	85.05	84.26	62.12	75.41	77.62
	NorSA	79.48	**84.39**	47.80	**84.80**	**85.08**	**85.44**	**63.99**	**75.85**	**78.29**
50%	TEAL	73.08	80.30	43.60	79.90	81.74	80.17	56.82	70.80	69.20
	NorSA w/o R	77.27	83.13	45.80	84.25	84.31	82.79	61.09	74.09	76.61
	La RoSA	77.90	82.59	46.40	84.24	84.10	82.49	58.96	73.81	76.51
	NorSA	**78.85**	**83.62**	**47.20**	**84.56**	**84.86**	**85.23**	**62.37**	**75.24**	**78.15**
60%	NorSA w/o R	74.66	78.29	44.80	78.66	81.22	72.01	46.84	68.07	72.98
	NorSA	**78.22**	**83.19**	**47.60**	**84.14**	**84.86**	**84.51**	**62.37**	**74.98**	**76.09**

Table 11: Performance comparison of different sparsification methods on LLaMA-2-7B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Baseline	69.14	79.05	44.20	75.98	77.71	74.54	46.25	66.69	45.85
25%	TEAL	67.95	78.18	**44.00**	75.20	76.71	73.91	45.64	65.94	44.66
	NorSA w/o R	68.58	78.34	**44.00**	**75.91**	77.46	**74.87**	**46.59**	**66.54**	45.61
	La RoSA	68.43	**79.05**	**44.00**	75.84	77.22	74.66	45.56	66.39	45.66
	NorSA	**69.38**	79.00	43.20	75.75	**77.80**	74.49	46.08	66.52	**45.81**
40%	TEAL	66.93	77.96	42.60	74.09	75.90	73.31	43.68	64.92	43.46
	NorSA w/o R	67.09	78.02	44.00	75.07	76.02	72.85	44.71	65.39	43.33
	La RoSA	68.90	78.18	**44.20**	**75.35**	76.85	73.86	45.73	66.15	44.66
	NorSA	**69.38**	**78.56**	43.60	75.30	**77.37**	**74.37**	**46.08**	**66.38**	**45.42**
50%	TEAL	65.27	77.53	41.20	71.39	73.33	71.17	42.66	63.22	39.57
	NorSA w/o R	65.98	77.37	41.40	73.21	75.26	71.25	44.20	64.09	40.33
	La RoSA	67.32	77.69	**43.40**	74.37	73.85	72.31	43.34	64.61	43.10
	NorSA	**68.67**	**78.29**	**43.40**	**74.86**	**75.29**	**73.36**	**44.54**	**65.49**	**43.25**
60%	NorSA w/o R	65.19	75.68	36.00	68.60	69.76	66.84	39.25	60.18	33.59
	NorSA	**67.01**	**77.86**	**42.40**	**73.30**	**72.57**	**71.59**	**42.41**	**63.88**	**39.18**

Table 12: Performance comparison of different sparsification methods on LLaMA-2-70B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Baseline	77.97	82.80	48.80	83.81	83.79	81.10	57.33	73.66	68.80
25%	TEAL	**78.56**	82.42	48.80	82.66	83.25	81.26	56.24	73.31	67.90
	NorSA w/o R	78.14	82.92	47.80	**83.78**	83.27	80.81	56.91	73.37	68.39
	La RoSA	77.58	80.09	48.60	83.63	**83.88**	**82.70**	57.17	73.38	**68.74**
	NorSA	78.22	**82.97**	**49.20**	83.70	83.70	80.56	**57.59**	**73.71**	68.56
40%	TEAL	76.21	81.65	48.30	82.02	82.94	80.87	55.30	72.47	66.78
	NorSA w/o R	77.51	83.08	48.20	83.45	82.66	**81.52**	57.00	73.34	67.54
	La RoSA	77.43	**83.57**	47.40	83.37	**83.36**	80.51	57.76	73.31	68.16
	NorSA	**78.14**	82.59	**49.00**	**83.58**	83.30	80.68	**58.19**	**73.64**	**68.22**
50%	TEAL	75.74	81.33	47.40	80.67	81.80	80.35	56.16	71.92	64.43
	NorSA w/o R	76.40	81.77	48.20	83.00	82.75	79.97	**57.34**	72.77	66.28
	La RoSA	76.80	**82.86**	**48.80**	82.99	**82.78**	79.63	56.14	72.86	67.57
	NorSA	**77.98**	**82.86**	**48.80**	**83.51**	82.48	**80.60**	56.91	**73.31**	**67.73**
60%	NorSA w/o R	75.30	81.23	46.00	81.52	79.88	79.04	**55.80**	71.25	64.08
	NorSA	**76.95**	**82.37**	**49.40**	**82.80**	**80.76**	**80.35**	**55.80**	**72.63**	**66.75**

Table 13: Performance comparison of different sparsification methods on Qwen-2.5-7B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Dense	72.77	79.98	47.60	78.91	84.56	77.44	51.11	70.34	74.21
25%	TEAL	72.12	79.22	46.40	78.49	84.40	76.85	50.85	69.76	73.21
	NorSA w/o R	72.14	79.71	**47.40**	78.68	84.53	**77.40**	50.94	70.11	73.78
	La RoSA	**72.69**	**79.92**	46.60	**78.77**	84.40	77.02	**51.45**	**70.12**	73.74
	NorSA	71.67	79.82	47.00	78.72	**84.65**	76.94	51.02	69.97	**73.90**
40%	TEAL	70.30	78.78	45.00	76.78	84.05	75.44	49.96	68.61	71.44
	NorSA w/o R	70.72	79.05	45.20	77.73	84.10	75.88	**51.88**	69.22	72.65
	La RoSA	70.72	79.38	46.20	78.28	**84.65**	**77.86**	50.60	69.67	72.33
	NorSA	**71.35**	**79.82**	**46.40**	**78.43**	84.31	76.98	50.77	**69.72**	**73.35**
50%	TEAL	68.90	77.18	44.00	74.60	83.76	76.59	49.34	67.76	68.53
	NorSA w/o R	69.22	78.45	44.00	76.08	83.76	76.89	50.94	68.47	70.29
	La RoSA	**70.24**	79.11	**45.60**	77.20	83.55	**78.03**	49.91	69.09	70.09
	NorSA	69.85	**79.54**	**45.60**	**77.59**	**84.31**	76.56	**51.37**	**69.26**	**72.28**
60%	NorSA w/o R	67.09	78.40	40.60	72.55	80.34	74.83	48.46	66.38	66.91
	NorSA	**70.01**	**79.00**	**43.80**	**75.75**	**84.04**	**77.19**	**50.94**	**68.67**	**69.78**

Table 14: Performance comparison of different sparsification methods on Qwen-2.5-72B.

Sparsity	**Method**	WinoGrande	PiQA	OBQA	HellaSwag	BoolQ	ARC-E	ARC-C	**Avg**	MMLU
0%	Dense	77.90	83.62	46.40	86.05	89.14	83.33	62.62	75.58	86.08
25%	TEAL	78.11	83.39	46.50	85.23	87.63	82.62	61.87	75.05	85.44
	NorSA w/o R	**79.40**	83.62	46.40	85.96	88.87	82.91	62.03	**75.59**	85.75
	La RoSA	77.66	83.73	**46.80**	**85.97**	**89.17**	82.83	**62.54**	75.53	85.62
	NorSA	78.06	**84.22**	46.60	85.93	88.90	**83.04**	62.29	75.57	**85.80**
40%	TEAL	77.47	83.01	46.50	84.52	87.52	82.43	61.11	74.65	84.80
	NorSA w/o R	77.35	**83.90**	**47.20**	85.74	89.02	83.33	61.52	75.44	85.18
	La RoSA	78.45	83.03	**47.20**	**86.01**	**89.48**	83.12	**62.46**	**75.68**	85.53
	NorSA	**79.32**	83.73	47.00	85.88	88.81	**83.38**	61.43	75.65	**85.83**
50%	TEAL	77.32	82.25	45.00	83.03	87.54	81.61	59.46	73.74	83.54
	NorSA w/o R	76.40	82.81	46.40	85.65	88.47	82.91	60.41	74.72	84.95
	La RoSA	76.48	83.57	45.60	85.81	**89.51**	**83.59**	**61.69**	75.18	84.34
	NorSA	**77.51**	**83.68**	**47.80**	**85.85**	88.81	83.08	61.43	**75.45**	**85.66**
60%	NorSA w/o R	74.19	82.26	45.20	85.02	88.32	81.94	57.85	73.54	83.31
	NorSA	**77.19**	**83.13**	**47.20**	**85.53**	**88.59**	**82.24**	**61.95**	**75.12**	**84.96**

D Fair Efficiency Comparison and Attribution

All measurements use LLaMA-3-8B at batch size 1 with FP16 `gpt-fast` decoding under PyTorch 2.4.0+cu124, Triton 3.0.0, and Transformers 4.44.1. Decoding runs under `torch.compile(fullgraph=True, mode="reduce-overhead")` with CUDA Graphs, a fixed 255-token WikiText-2 prompt, 5 warmup tokens, and 200 measured teacher-forced tokens with identical prerecorded token IDs; $\pm$ denotes the sample standard deviation over three independent processes.

Symmetric dense path. Dense and NorSA fold the learned RMSNorm affine scale into the following weights and both retain the all-one scale multiplication, so the dense baseline forgoes no optimization that NorSA receives; the comparison isolates sparsity and rotation. Both paths also share the compiled graph, CUDA Graph capture, KV-cache layout, and attention backend. The A100 results in Table 5 use a different protocol (200-token generation, and two-way tensor parallelism for 70B), so absolute latencies are not comparable across the two tables.

D.1 Fair dense baseline

Table 15 reports the same-GPU paired comparison. At 50% sparsity NorSA reduces per-token latency from 17.2336 to 11.9218 ms, a $1.4456\times$ speedup and a paired net saving of 5.3119 ± 0.0309 ms.

Table 15: Fair end-to-end decoding latency on LLaMA-3-8B (RTX 4090, batch size 1, FP16). Dense and NorSA fold the learned RMSNorm affine scale and both retain the all-one scale multiplication.

Method	**Sparsity**	**Rotation**	**ms/token** $\downarrow$	**Speedup** $\uparrow$
Dense	0%	–	17.2336 ± 0.0027	$1.0000\times$
NorSA	0%	online	19.0030 ± 0.0181	$0.9068\times$
NorSA	50%	online	$\mathbf{11.9218 \pm 0.0298}$	$\mathbf{1.4456}\times$

D.2 Component attribution

Table 16 decomposes the change in a separate session. The complete sparse path is 1.7709 ms/token *slower* than Dense at nominal 0% sparsity, so the gain does not come from a faster zero-sparsity backend; raising sparsity to 50% saves 7.0876 ms/token, more than offsetting the selector, layout, and rotation overhead. Online rotation is the dominant NorSA-specific cost (1.4525 ± 0.0295 ms/token); pre-norm reuse saves 0.1779 ms/token and fusing SwiGLU with the next norm is neutral.

Table 16: Separate component sweep on the same stack; its Dense reference differs from Table 15 by 0.0015 ms because it is an independent session. Rows below the second rule are differences within the sweep and carry no separate standard deviation.

Path / component	**Effect** (ms/token)
Dense	17.2321 ± 0.0015
NorSA, nominal 0%	19.0030 ± 0.0181
NorSA, 50%	11.9154 ± 0.0538
Sparse-path overhead at 0%	$+1.7709$
Activation sparsity, 0%$\rightarrow$50%	-7.0876
Net change at 50%	-5.3167 ($1.4462\times$)
Online rotation at 50%	$+1.4525 \pm 0.0295$
Pre-norm reuse at 0%	-0.1779
Fused SwiGLU and norm	$+0.0082$ (neutral)

D.3 Cross-hardware replication

Table 17 repeats the paired comparison on H20 with the same stack. The attribution is unchanged, but the magnitude tracks memory bandwidth and dense-kernel quality, so every platform is compared

against its own dense baseline. Online rotation costs 1.4525 ± 0.0295 ms/token on RTX 4090 and 0.6503 ± 0.0049 ms/token on H20.

Table 17: Cross-hardware replication of the paired comparison (LLaMA-3-8B, batch size 1, FP16, identical software stack).

GPU	**Dense** (ms/token)	**NorSA, 50%** (ms/token)	**Speedup**
RTX 4090	17.2336 ± 0.0027	11.9218 ± 0.0298	$1.4456\times$
H20	11.4874 ± 0.0971	7.6219 ± 0.0100	$1.5072\times$

D.4 Numerical checks

On a trace disjoint from the main evaluation set, the realized sparsity is 24.71% for a 25% target and 49.46% for a 50% target. Composing adjacent rotations reduces the number of online transforms per token from 64 to 33 and reproduces the reference logits with cosine similarity 0.9999868, relative ℓ_2 error 0.41%, and identical top-1 predictions. Reusing the pre-norm statistic changes the selector decision for 0.11% of entries on average (maximum 1.89%) because of the RMSNorm ϵ; the quality results apply the exact selector of Eq. (4), and the reuse is confined to the latency-optimized path measured above.

E Matched-Harness Comparison with TEAL and La RoSA

TEAL uses its upstream split-K Triton `splitk_sparse_gemv` kernel (commit `fb7373c9`) through a `torch.library` wrapper. NorSA shares its layout, autotuning list, FP32 tile reduction, and FP16 output atomic, changing the selector from $|x_i| > \tau$ to $x_i^2 > \tau^2 \|\mathbf{x}\|^2$. The La RoSA-style control combines exact `torch.topk` with our indexed FP16 GEMV. Its end-to-end path uses NorSA's WikiText-2 PCA-initialized $\mathbf{R}_1$, without NorSA's $\mathbf{R}_2$, and adjacent-layer rotation composition. It is a shared-rotation control, not a timing of an upstream La RoSA runtime or a reproduction of its full rotation-calibration procedure.

E.1 End-to-end decoding

We use the decoding protocol in Appendix D, with method measurements collected in separate sessions; the displayed Dense reference comes from the rotated La RoSA-style control. Dense, NorSA, and the rotated La RoSA-style path fold the block RMSNorm affine scales and retain the all-one multiplication; TEAL retains its original RMSNorm path. This compares method-specific implementations, not identical graph transformations.

Table 18: End-to-end decoding latency (ms/token) on LLaMA-3-8B and RTX 4090. NorSA and the La RoSA-style control include online rotation. Lower is better.

Sparsity	**TEAL**	**NorSA**	**La RoSA-style**
Dense (reference)		17.2335 ± 0.0015	
0%	$\mathbf{17.4099 \pm 0.0242}$	19.0030 ± 0.0181	22.1291 ± 0.0352
25%	$\mathbf{13.8417 \pm 0.0018}$	15.3731 ± 0.0030	18.8242 ± 0.0189
50%	$\mathbf{10.2447 \pm 0.0346}$	11.9154 ± 0.0538	14.9844 ± 0.0058

At 50%, Table 18 gives approximately $1.68\times$, $1.45\times$, and $1.15\times$ over the displayed Dense reference for TEAL, NorSA, and the La RoSA-style control. The paired study in Appendix D measures NorSA's rotation overhead as 1.4525 ms/token. A separate no-rotation run reaches 10.4952 ± 0.0073 ms/token, suggesting rotation explains most of the gap to TEAL; the remaining end-to-end difference includes implementation effects and cannot be attributed solely to selector arithmetic. For quality context, Table 1 reports NorSA/TEAL perplexity of $6.57/7.56$ and MMLU of $62.12\%/52.78\%$ at 50%. These are the main-paper quality results, not a re-evaluation of the latency-optimized paths.

Table 19: Operator-level macro latency (μs) and speedup over the dense operator on RTX 4090, FP16, real decode traces. Online rotation is excluded; the La RoSA-style row measures exact Top-k plus our indexed GEMV.

Sparsity	**TEAL**		**NorSA**		**La RoSA-style**	
	μs	Speedup	μs	Speedup	μs	Speedup
0%	63.73	1.05×	63.16	1.10×	84.04	0.67×
25%	49.70	1.26×	49.68	1.27×	69.66	0.78×
50%	28.57	1.99×	28.42	1.99×	48.00	1.14×

E.2 Operator level

Table 19 aggregates five layers (0, 7, 15, 23, 31), all seven projection types, and three independent runs. Latencies are macro means of per-operator median timings; speedups are averaged per-operator Dense/sparse ratios, not ratios of macro mean latencies. NorSA reuses an available norm. Its latency is close to TEAL's at matched target sparsity, whereas the La RoSA-style control includes exact Top-k selection overhead. At nominal 0%, TEAL and NorSA prune about 0.05% exact zeros; those rows primarily reflect backend overhead rather than sparsity.

F Relation to CATS and Sparsity Scope

CATS sparsifies only the gated-MLP intermediate: it computes $\mathbf{W}_{\mathrm{gate}}$ densely, thresholds $\mathrm{SiLU}(\mathbf{x}\mathbf{W}_{\mathrm{gate}})$ at a per-block calibrated percentile, and masks the output of $\mathbf{W}_{\mathrm{up}}$ and the input of $\mathbf{W}_{\mathrm{down}}$; attention projections and the gate input stay dense. NorSA, TEAL, and La RoSA instead sparsify all linear inputs, so equal nominal sparsity has a different model-wide scope: on LLaMA-2-7B, 25% model-level sparsity corresponds to roughly 56% MLP-intermediate sparsity under CATS, and 40% to roughly 90%, where quality collapses.

Table 20: WikiText-2 perplexity at 25% model-level sparsity, quoted from TEAL [Liu et al., 2025a]. CATS has a narrower scope and the values were not measured in our harness.

Method	**LLaMA-2-7B**	**LLaMA-3-8B**	**Mistral-7B**
Dense (0%)	5.07	5.87	4.92
CATS (25% model-level)	5.52	6.78	5.87
TEAL (25%)	5.09	5.94	5.01

Table 20 is reference only: the values come from a different protocol and were not re-measured, so they should not be read as a controlled comparison with NorSA. At 40% model-level sparsity TEAL reports CATS at 10^4-order perplexity on LLaMA-3-8B and Mistral-7B, and at 43.8 and 171 on LLaMA-2-7B/70B.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution, the method components, and the empirical scope of the paper. The main claims are supported by the experiments in Sections 5 and Appendix C, and the limitation discussion is included in the Conclusion and Limitation section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss a practical limitation of sparse activation methods in the Conclusion and Limitation section. In particular, we note that the system-level benefit can become less direct in larger-batch serving settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper states the assumptions used in the derivations and provides the corresponding formulas and derivation steps in the main text. The presentation is intended to support the method design rather than introduce standalone formal theorems.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper specifies the evaluated model families, tasks, sparsity levels, calibration setting, baselines, and kernel setting used for the main results. Additional implementation and detailed result tables are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide an anonymized code package at submission time, together with the instructions needed to reproduce the main experimental results. The paper and appendix also describe the main evaluation setup.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper describes the evaluation setup, including models, tasks, sparsity levels, calibration procedure, and hardware setting for the speed results. The appendix provides additional kernel and evaluation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars or repeated-run statistics in this submission. Most results are deterministic evaluation metrics on fixed models, datasets, and sparsity settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the key compute setting used for the main efficiency experiments, including the GPU type and tensor parallelism configuration. These details are sufficient to understand and reproduce the reported system measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, this work conforms to the NeurIPS Code of Ethics. The paper studies inference-time efficiency methods and does not involve human subjects or sensitive private data collection.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: We do not include a dedicated broader-impact discussion in the current submission. This paper focuses on an inference efficiency method rather than a deployment study, and we chose not to make strong societal claims beyond the technical scope of the work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release a new model, dataset, or other asset in this submission. The paper studies an inference-time method applied to existing model families.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We credit the original model and method sources used in the experiments and explicitly state the license name CC BY 4.0 where applicable, while following the stated terms of use of the assets we rely on.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: We do not introduce or release a new dataset, model, or benchmark asset as part of this submission.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This work does not involve crowdsourcing or experiments with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve human subjects, so IRB review or an equivalent process was not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Any LLM usage was limited to writing or editing assistance. It was not part of the core method, experiments, or scientific claims, so declaration is not required under the checklist guidance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.