Accelerate Speculative Decoding with Sparse Computation in Verification

Anonymous ACL submission

Abstract

Speculative decoding accelerates autoregressive language model inference by verifying multiple draft tokens in parallel. However, the verification stage often becomes the dominant computational bottleneck, especially for long-context inputs and mixture-of-experts (MoE) models. Existing sparsification methods are designed primarily for token-by-token autoregressive decoding to remove substantial computational redundancy in LLMs. This work systematically adopts different sparse methods on the verification stage of the speculative decoding and identifies structured redundancy across multiple dimensions. Based on these observations, we propose a sparse verification framework that jointly sparsifies attention, FFN, and MoE components during the verification stage to reduce the dominant computation cost. The framework further incorporates an inter-draft token and inter-layer retrieval reuse strategy to further reduce redundant computation without introducing additional training. Extensive experiments across summarization, question answering, and mathematical reasoning datasets demonstrate that the proposed methods achieve favorable efficiency-accuracy trade-offs and can be translated into practical latency reduction with optimized sparse kernels.

1 Introduction

Autoregressive large language models (LLMs) (Meta-AI, 2024; OpenAI et al., 2024; Guo et al., 2025) have achieved remarkable success across a wide range of tasks, but their inference cost continues to grow rapidly with increasing model size and context length. While training-time optimizations have received extensive attention, inference efficiency remains a critical bottleneck. This issue becomes particularly severe in long-context inference, where attention computation over large KV caches introduces substantial latency overhead.

Speculative decoding (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2023) has recently emerged as an effective technique to accelerate autoregressive generation. By introducing a lightweight draft model to propose multiple candidate tokens and verifying them in parallel using the target model, it amortizes decoding overhead across multiple tokens per step. However, this acceleration shifts the computational burden to the verification stage. In particular, when draft lengths grow, or long-context inputs are involved, verification incurs substantial overhead due to full attention over large KV caches, dense feed-forward computation, and multi-expert evaluation in MoE models. As a result, verification becomes the dominant bottleneck in speculative decoding.

Module	FLOPs
Full Attention	$6\,BTd^2 \;+\; 4\,BT^2d$
Sparse Attention	$6\,BTd^2 \;+\; 4\,B\,(1-s_a)\,T^2d$
Dense FFN	$6\,BTd\,d_{\mathrm{f}}$
Sparse FFN	$2BTd\,d_{\mathrm{f}}\,(3-2s_f)$
MoE FFN	$6\,BT\,k\,d\,d_{\mathrm{e}} \;+\; 2\,BTd\,E$
Sparse MoE	$6\,BT\,(1-s_e)k\,d\,d_{\mathrm{e}} \;+\; 2\,BTd\,E$

Table 1: FLOPs estimation for different modules. L: number of layers, d: hidden size, d_{f}: FFN intermediate size, d_{e}: expert hidden size, n_h: number of attention heads, T: sequence length, B: batch size, E: number of experts, k: number of active experts per token. s_a, s_f, s_e indicate sparsity of attention, sparsity of down projection in FFN, sparsity of active experts.

To better understand speculative verification, we analyze its internal computation patterns and observe that verification exhibits structured redundancy across multiple dimensions. First, draft tokens within the same verification step tend to retrieve highly overlapped contextual regions, suggesting that retrieval results can be shared across tokens. Second, retrieval patterns across adjacent transformer layers exhibit strong similarity, especially in middle and deep layers, indicating that

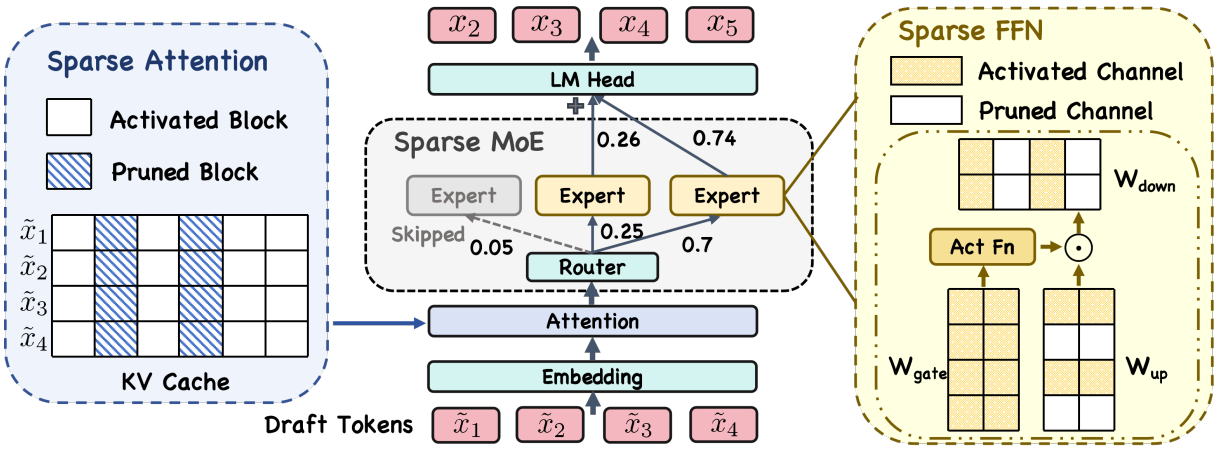

Figure 1: An illustration of the three sparse verification strategies within a single Transformer layer. Residual connections and normalization layers are omitted for clarity. A KV block contains the key-value states of multiple prefix tokens. In sparse attention, the selection of blocks is determined by the dot product between the query of the first draft token and a representative value of each block, and multiple draft tokens select the same set of blocks. In sparse FFN, the pruned channels are determined by the output of the activation function and a given threshold. In sparse MoE, a dynamic strategy based on the weights of the selected experts determines whether an expert participates in computation, and different draft tokens may involve different numbers of experts.

retrieval computation itself is highly redundant. Third, speculative verification is more tolerant to moderate approximation errors than standard autoregressive decoding because verification mainly depends on preserving acceptance behavior rather than reconstructing exact token distributions.

Motivated by these observations, we propose a verification-oriented sparse computation framework for speculative decoding. Instead of independently applying existing sparse inference methods, our framework coordinates sparsification across attention, FFN, and MoE components while explicitly considering the unique properties of multi-token verification.

Specifically, we introduce (1) an importance-based sparse attention mechanism tailored for multi-token verification, where block retrieval is performed once using the first draft token and shared across all draft tokens within the same verification step, combined with a piecewise budget control strategy to ensure stability on short contexts; (2) an inter-layer retrieval reuse scheme that avoids redundant block selection across similar layers; (3) a sparse feed-forward verification method that prunes low-activation channels during inference; and (4) a generalized sparse MoE strategy that adaptively skips low-contribution experts. We further combine these techniques into a hybrid sparse verification method that sparsifies verification along three orthogonal dimensions. We provide a detailed FLOPs analysis of the proposed methods in Table 1, showing that sparse verification substantially reduces the dominant computation terms in attention, FFN, and MoE layers. Extensive experiments across long-context, question answering, and mathematical reasoning benchmarks demonstrate that our methods achieve significant efficiency gains while maintaining competitive verification accuracy and stable acceptance length. We further implement a Triton-based sparse attention kernel to efficiently support sparse verification under long-context settings. Experimental results demonstrate that the proposed kernel achieves substantial practical latency reduction, providing up to $4.6\times$ acceleration over standard PyTorch full attention at 256K context length.

2 Preliminary

2.1 Speculative Decoding

Speculative decoding (Stern et al., 2018; Leviathan et al., 2023; Xia et al., 2023) is an inference technique for autoregressive models that aims to accelerate generation. Let $p_\theta(x_t \mid x_{<t})$ denote the conditional distribution of the target token x_t given previous tokens $x_{<t}$ under the target model with parameters θ. Speculative decoding introduces a lightweight draft model, with parameters ϕ, to propose multiple candidate tokens in advance. At time step t, the draft model generates K speculative tokens $(\tilde{x}_{t+1}, \tilde{x}_{t+2}, \ldots, \tilde{x}_{t+K})$ sequentially:

$$\tilde{x}_{t+k} \sim p_\phi(x_{t+k} \mid x_{<t+k}), \quad k = 1, \ldots, K. \quad (1)$$

During verification, the target model computes the probabilities $P_{t:t+K} = p_\theta(\tilde{x}_{t:t+K} \mid x_{<t})$ for the draft tokens $\tilde{x}_{t:t+K}$. Each draft token $\tilde{x}_{t+k}$ is evaluated with an acceptance probability:

$$\alpha_{t+k} = \min\left(1, \frac{p_\theta(\tilde{x}_{t+k} \mid x_{<t+k})}{p_\phi(\tilde{x}_{t+k} \mid x_{<t+k})}\right), \quad (2)$$

where p_ϕ is the draft model's output probability. Note that once a token $\tilde{x}_{t+k}$ is rejected, all its subsequent draft tokens are discarded. The rejected token is resampled from the adjusted distribution:

$$\hat{p} = norm(\max\left(0, p_\theta - p_\phi\right)), \quad (3)$$

ensuring that the overall generation remains consistent with the target model distribution.

2.2 KV Cache Eviction in Sparse Verification

A major trend in key-value cache eviction is to determine which caches to activate based on the current token information (Tang et al., 2024a; Lu et al., 2025). For speculative decoding, each step of the decoding process involves multiple tokens as input. Therefore, it is necessary to analyze the retrieval differences of these tokens to better adapt to these strategies. To this end, we perform speculative decoding on Llama3.1-8B (Grattafiori et al., 2024) and analyze the overlap of cache blocks retrieved by different draft tokens based on their respective query values. Figure 2 displays the results. In the experiments, a tree-structured draft with 60 draft tokens is adopted. We measure the overlap ratio of retrieved blocks between token pairs at different positional distances. The overlap is calculated as the intersection of the retrieved blocks for the two tokens divided by the total number of blocks.

The results show that tokens within the same step retrieve highly similar blocks, with an average overlap exceeding 0.8 in most layers. Moreover, tokens that are closer in position tend to retrieve more similar blocks. On the other hand, in the draft tree, the first token is the accepted token sampled from the previous step, making it more important than the others since it influences the verification of all subsequent tokens. Taking into account both aspects, we use the first token to perform block retrieval, while the other tokens reuse its retrieved blocks during verification.

3 Sparse Verification

In this section, we introduce a sparse verification framework that exploits redundancy in the verification process. The key idea is that verification

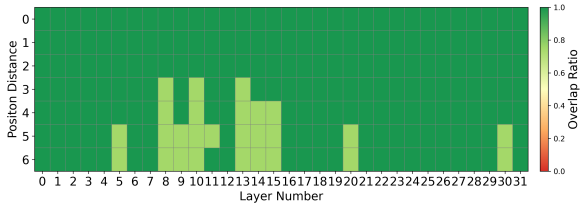

Figure 2: The overlap ratio of the KV block selection for the draft token pairs with different position distances.

does not require full computation across all model components. We therefore apply controlled sparsification along three dimensions: attention, feed-forward networks, and mixture-of-experts layers. Unlike prior sparsity-aware methods (Kwon et al., 2022; Yuan et al., 2025) that require retraining or architectural modification, our approach operates entirely at inference time. This design choice allows sparse verification to be directly applied to off-the-shelf large models.

3.1 Sparse Attention

Existing query-aware retrieval methods assume that each decoding step contains only a single query token. In speculative verification, however, multiple draft tokens are verified simultaneously. Directly performing independent retrieval for every draft token introduces two issues. First, retrieval overhead grows proportionally with the number of draft tokens, partially offsetting the acceleration brought by speculative decoding. Second, independently retrieved blocks may introduce inconsistent contextual evidence across draft tokens, potentially destabilizing token acceptance behavior.

To address this problem, we exploit the observation that draft tokens within the same verification step exhibit highly overlapped retrieval patterns. We therefore perform retrieval only once using the first draft token and share the retrieved blocks across all draft tokens during verification.

Formally, each block contains KV entries corresponding to one attention head and a fixed number of consecutive tokens, denoted as the block size B. If there are H KV heads and a total sequence length of L_{seq}, the KV cache can be represented as:

$$\mathcal{K} = \{\mathbf{K}_{h,b}\}, \quad \mathcal{V} = \{\mathbf{V}_{h,b}\}, \quad (4)$$
$$h \in [1, H], \; b \in [1, \lceil L_{\text{seq}}/B \rceil].$$

For speculative verification, we compute an importance score for each block using the query of the first draft token $\mathbf{q}_0$ and the mean vector of all

key states within the block:

$$s_{h,b} = \mathbf{q}_0^\top \left(\frac{1}{B} \sum_{i=1}^{B} \mathbf{k}_{h,b,i} \right), \tag{5}$$

and pick N blocks with highest s for calculation.

Moreover, due to the existence of the attention sink (Xiao et al., 2024), where early tokens tend to attract disproportionate attention across layers, and the locality bias (Yang et al., 2018a; Wu et al., 2025), where neighboring tokens play a more critical role in contextual reasoning, we retain both the first and last few blocks without eviction. This boundary-preserving design prevents the loss of essential contextual information that is frequently referenced during verification.

3.2 Piecewise Budget Control

In practice, we observe that when the KV cache length is below a certain threshold, the verification time does not grow noticeably with the cache size. Moreover, for short-text scenarios, aggressive KV eviction may introduce performance degradation. To balance efficiency and stability, we adopt a piecewise budget control strategy.

Specifically, when the sequence length L_{seq} is below a fixed threshold L_0, we perform no eviction. When $L_{\text{seq}} > L_0$, the number of retained blocks is adaptively determined by:

$$N_{\text{budget}} = \frac{\big((L_{\text{seq}} - L_0) \times \rho + L_0\big) \times H}{B}, \tag{6}$$

where $0 < \rho < 1$ is a sparsity coefficient controlling the proportion of active blocks, H is the number of KV heads, and B is the block size.

This design ensures that for short sequences, verification remains stable, while for longer contexts, computational cost scales smoothly with sequence length under a controlled sparsity budget.

3.3 Inter-layer Retrieval Reuse

Through empirical analysis of the retrieved blocks across transformer layers, we observe that although the retrieval patterns in shallow layers tend to be irregular and diverse, the block selection in middle and deeper layers exhibits strong inter-layer similarity, particularly among adjacent layers. This observation suggests that performing retrieval at all layers is redundant and leads to unnecessary computational overhead.

To exploit this redundancy, we introduce an inter-layer retrieval reuse strategy that selectively performs retrieval only on representative layers. We first construct a set of calibration data $\mathcal{D}_{\text{cal}}$ covering various input lengths to analyze the model's retrieval behavior. For each input sample, we record the mask of the blocks retrieved for each layer l, denoted as $M_l \in \{0,1\}^{N_b}$, where N_b is the total number of candidate blocks.

We then compute the pairwise retrieval similarity between adjacent layers using Jaccard similarity:

$$J(M_l, M_{l-1}) = \frac{|M_l \cap M_{l-1}|}{|M_l \cup M_{l-1}|}. \tag{7}$$

For the first layer, we set $J(M_1, M_0) = 0$.

Based on these similarity scores, we identify a subset of layers with the lowest similarity to their preceding layers as anchor layers:

$$\mathcal{A} = \text{TopK}_l(1 - J(M_l, M_{l-1})), \tag{8}$$

where $|\mathcal{A}| = K$ and K is the number of anchor layers to be selected.

Only the anchor layers perform block retrieval to determine which blocks participate in computation. For non-anchor layers, we reuse the retrieval results of the nearest preceding anchor layer:

$$M_l = M_{a(l)}, \quad a(l) = \max\{i \in \mathcal{A} \mid i < l\}. \tag{9}$$

This layer-wise reuse mechanism effectively reduces retrieval operations while maintaining comparable verification quality, as the selected anchor layers capture the essential variation of retrieval patterns across the network.

3.4 Sparse Feedforward Network

Existing sparse FFN (SFFN) methods (Roller et al., 2021; Liu et al., 2023) are primarily designed for the training stage, where sparsity is introduced to reduce the cost of updating parameters. Inspired by these approaches, we leverage the inherent sparsity of the feed-forward network (FFN) during inference to reduce verification cost.

A standard gated FFN consists of a gate-projection, an up-projection, an activation function, and a down-projection. Let the input at layer l be $\mathbf{x}_l \in \mathbb{R}^d$. The FFN computation can be written as:

$$\mathbf{h}_l = \sigma(W_{\text{gate}}\mathbf{x}_l + \mathbf{b}_{\text{gate}}), \tag{10}$$

$$\mathbf{y}_l = W_{\text{down}}(\mathbf{h}_l \odot (W_{\text{up}}\mathbf{x}_l + \mathbf{b}_{\text{up}})) + \mathbf{b}_{\text{down}}, \tag{11}$$

where $\sigma(\cdot)$ is activation function, typically GeLU (Hendrycks, 2016) or SwiGLU (Shazeer, 2020).

Prior work (Song et al., 2024) observes that many FFN channels are rarely activated and proposes to induce structured sparsity by replacing smooth activation functions (e.g., GeLU) with ReLU, followed by retraining or finetuning to adapt the model to the modified activation pattern. However, we observe that even in standard pretrained models with GeLU or SwiGLU activations, a large fraction of FFN channels exhibit near-zero activation values during inference. To exploit this sparsity, we identify inactive channels by thresholding the activation magnitude:

$$\mathcal{S}_l = \{i \mid |h_{l,i}| < \tau\}, \tag{12}$$

where τ is a predefined threshold and $h_{l,i}$ is the i-th dimension of $\mathbf{h}_l$.

Only channels not belonging to $\mathcal{S}_l$ are used in the up and down projection. Thus, the sparse FFN output becomes:

$$\mathbf{h}_{\text{up}} = \mathbf{h}_l^{(\neg\mathcal{S}_l)} \odot (W_{\text{up}}^{(\neg\mathcal{S}_l)}\mathbf{x}_l^{(\neg\mathcal{S}_l)} + \mathbf{b}_{\text{up}}^{(\neg\mathcal{S}_l)}), \tag{13}$$

$$\mathbf{y}_l = W_{\text{down}}^{(\neg\mathcal{S}_l)}\mathbf{h}_{\text{up}} + \mathbf{b}_{\text{down}}^{(\neg\mathcal{S}_l)}, \tag{14}$$

where $(\neg\mathcal{S}_l)$ indicates the set of active channels. This mechanism preserves the structure of the FFN while significantly reducing multiplications in the up-projection and down-projection, which dominate the FFN computational cost. By applying this sparsification during speculative verification, we reduce overall inference computation cost while maintaining comparable validation accuracy.

3.5 Sparse Mixture of Experts

We propose an adaptive expert skipping method for each candidate token for MoE target models. Lu et al. (2024) have introduced a dynamic skipping expert strategy during inference. They consider the case of k=2 activated experts, where the second expert may be skipped based on a threshold determined by the ratio of routing logits. We generalize this mechanism to arbitrarily activated expert counts $k > 2$, and allow skipping up to m experts ($1 \leq m < k$) for each token during inference.

Given k routed experts in an MoE layer, we denote their routing weights, after sorting in ascending order, as $w = \{w_1, w_2, \ldots, w_k\}$. To determine how many experts can be skipped, we compute a layer-specific threshold for every possible skip size m. For a calibration dataset, we first calculate a ratio for all calibration tokens:

$$\frac{\sum_{j=1}^{k-m} w_j}{\sum_{j=1}^{k} w_j}, \tag{15}$$

and define the threshold β_m as the median of these ratio values in that layer. This produces a *threshold map* $\{\beta_1, \beta_2, \ldots, \beta_{k-1}\}$ that specifies allowed skip levels for that layer.

During inference, for a given token, we search for the maximum number of experts i to skip ($0 \leq i \leq m$) such that

$$\frac{\sum_{j=1}^{k-i} w_j}{\sum_{j=1}^{k} w_j} < \beta_m. \tag{16}$$

If such $i > 0$ exists, we skip the i lowest-weight experts $\{e_1, e_2, \ldots, e_i\}$; otherwise ($i = 0$), all k experts are preserved. This allows the skipping strategy to remain stable across different inputs and verification steps.

4 Experiments

4.1 Performance of Sparse Attention

We evaluate the performance of sparse attention verification with EAGLE-3 (Li et al., 2025). We adopt Llama3.1-8B-Instruct (Dubey et al., 2024) as the target model. Tree-structured drafts are used, which have 60 candidate tokens. The experiment is conducted on 8 NVIDIA-H800-80G GPUs. We evaluate the performance with sparse attention on 5 datasets in LongBench (Bai et al., 2024) (including summarization, question answering, and code completion tasks): GovReport (Huang et al., 2021), 2WikiMQA (Ho et al., 2020), HotpotQA (Yang et al., 2018b), LCC (Guo et al., 2023), RepoBench-P (Liu et al., 2024). For sparse attention, the sparsity coefficient ρ is set to 0.1. To test the performance under different sparsity, we search the basic length L_0 in $\{1K, 2K, 4K\}$. We also compare the results with and without our proposed inter-layer retrieval reuse strategy. We randomly select 8 samples from GovReport as the calibration data. Appendix B shows the impact of different number of calibration samples. We additionally observe that the selected anchor layers and retrieval patterns remain relatively stable under different calibration subsets, suggesting that the proposed reuse strategy captures persistent structural redundancy rather than dataset-specific artifacts.

Results are shown in Table 2. Given a base length L_0, the sparsity ratio adapts based on the average sequence lengths across datasets. Under moderate sparsity levels (e.g., $L_0 = 4K$), SA exhibits minimal degradation compared with strict verification, with ROUGE/F1 drops typically within 0.3-1.0 points. This indicates that a large portion of

Verification	GovReport		2WikiMQA		HotpotQA		LCC		RepoBench-P	
	s_a	ROUGE	s_a	F1	s_a	F1	s_a	Edit Sim	s_a	Edit Sim
Strict	0	34.18	0	39.20	0	47.66	0	54.24	0	21.76
SA ($L_0 = 4K$)	0.34	33.50	0.39	39.32	0.41	47.43	0.41	55.10	0.45	22.23
SA* ($L_0 = 4K$)		33.28		35.78		44.78		55.18		22.24
SA ($L_0 = 2K$)	0.60	31.86	0.63	38.81	0.64	46.55	0.56	55.24	0.66	22.00
SA* ($L_0 = 2K$)		31.67		36.77		44.18		54.27		22.14
SA ($L_0 = 1K$)	0.75	30.62	0.77	38.86	0.77	46.28	0.72	54.99	0.78	21.61
SA* ($L_0 = 1K$)		29.45		37.87		43.18		53.52		21.76

Table 2: Performance of verification with sparse attention on different datasets. "Strict" indicates standard speculative decoding with full attention. "SA" represents sparse attention without an inter-layer retrieval reuse strategy, while "SA*" represents sparse attention with the reuse strategy. s_a is sparsity of attention, that is, the average ratio of the number of activated blocks to the total number of blocks.

attention computation is redundant. When sparsity increases, performance decreases more noticeably, yet the degradation remains within an acceptable range for most datasets. Datasets with stronger local dependencies (e.g., HotpotQA and LCC) show better robustness, suggesting that verification mainly relies on a subset of salient contextual tokens rather than the full attention map.

Introducing inter-layer reuse (SA*) slightly changes the trade-off. SA* consistently performs similarly to or slightly below SA, especially on QA datasets (e.g., 2WikiMQA, HotpotQA). This implies that while reuse reduces attention computation further, it introduces mild information loss due to cross-layer propagation of sparse patterns. However, for long-input summarization tasks such as GovReport and code-edit similarity tasks such as RepoBench-P, SA* performs comparably or even slightly better, indicating that reuse is more beneficial when long-range consistency, rather than fine-grained token interactions, dominates the verification process.

To further evaluate the practical efficiency of sparse attention, we implement a custom Triton-based sparse attention kernel and benchmark the latency of a single verification attention layer on NVIDIA H800 GPUs. Figure 3 compares the decoding latency under different KV cache lengths with standard PyTorch full attention and FlashAttention (Dao et al., 2022). As the KV cache length increases, sparse attention achieves significantly better scalability and reduces decoding latency by up to $4.6\times$ at 256K context length.

The results show that sparse attention provides increasingly significant acceleration as the KV

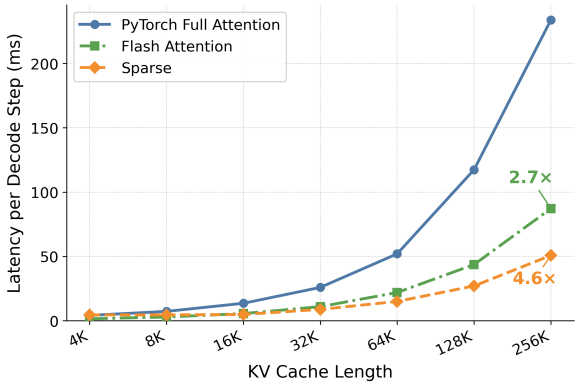

Figure 3: Latency comparison of a single attention layer under different KV cache lengths on NVIDIA H800 GPUs. We compare standard PyTorch full attention, FlashAttention, and our Triton-based sparse attention kernel. The reported speedup is computed relative to PyTorch full attention.

cache length grows. At short contexts, the latency difference remains relatively small because the computation is not yet dominated by KV retrieval. However, under long-context settings, sparse attention reduces the computation overhead by avoiding dense attention over the entire KV cache. At 256K context length, our sparse attention kernel achieves up to $4.6\times$ speedup over PyTorch full attention and $1.8\times$ speedup over FlashAttention.

4.2 Performance of Sparse FFN

Similar to sparse attention, we also evaluate the performance of sparse FFN with EAGLE-3. We employ Qwen3-30B-A3B (Qwen et al., 2025) as the target model. We apply SFFN to each expert. Unlike SA, SFFN is also applicable to short texts. Therefore, in addition to the three long text datasets,

Verification	s_f	GovReport	2WikiMQA	HotpotQA	GSM8K	Math	CollegeMath
		ROUGE	F1	F1	Acc.	Acc.	Acc.
Strict	0	32.67	43.93	63.32	90.0	65.6	23.8
SFFN ($\tau = 0.01$)	0.18	32.40	46.77	63.36	91.0	63.0	23.6
SFFN ($\tau = 0.05$)	0.47	32.96	43.99	63.33	90.7	63.0	23.8
SFFN ($\tau = 0.1$)	0.64	33.51	41.96	62.01	91.0	63.6	24.8

Table 3: Performance of verification with sparse FFN on different datasets with Qwen3-30B-A3B. "Strict" indicates standard speculative decoding with strict verification. s_f is the mean sparsity of FFN module.

we added additional mathematical tasks for our experiments, including GSM8K (Cobbe et al., 2021), Math (Hendrycks et al., 2021), and CollegeMath (Tang et al., 2024b). Note that we remove LCC and RepoBench-P because Qwen3-MoE performs poorly on code tasks. We search the threshold τ in $\{0.01, 0.05, 0.1\}$ for channel selection.

Table 3 displays the results under different sparsity levels. First, summarization (GovReport) remains stable across all sparsity levels. Notably, the ROUGE score slightly improves from 32.67 (strict) to 33.51 at $s_f = 0.64$. This suggests that SFFN does not disrupt the model's semantic structure checking and may even introduce regularization effects beneficial for long-form text generation. For QA (2WikiMQA, HotpotQA), performance also remains highly robust, indicating that reducing FFN computations in verification has minimal impact on tasks requiring compositional reasoning. Regarding math reasoning tasks (GSM8K, Math, CollegeMath), sparse verification again shows strong resilience. Importantly, even at $s_f = 0.64$, accuracy degradation is negligible, indicating that SFFN preserves the logical consistency required for step-by-step reasoning during verification.

Across all benchmarks, we observe no systematic degradation as sparsity increases, even though up to nearly two-thirds of FFN activations are pruned. These results indicate that SFFN verification achieves a more computationally efficient mechanism without compromising the correctness.

4.3 Performance of Sparse MoE

We evaluate the performance of Sparse MoE with Deepseek-R1 (Guo et al., 2025) on the same datasets as in Section 4.2. Deepseek-R1 assigns 8 experts for each token in the MoE layers. We consider a skipping budget m in $\{2, 3, 4\}$. We adopt the pretrained MTP (Multi-Token Prediction) heads as the draft model. The draft length is set to 4.

Table 4 shows that sparse MoE verification (SMoE) reduces expert usage while preserving competitive accuracy. Moderate sparsification ($m = 2, 3$) yields stable or even improved performance on several datasets (e.g., 2WikiMQA and Math with $m = 3$), indicating that skipping low-contribution experts can remove noisy validation signals. However, as sparsity increases further ($m = 4$), performance begins to degrade, particularly on math-heavy tasks, showing that excessive expert skipping may remove informative reasoning paths. Therefore, SMoE requires careful sparsity control to avoid harming verification reliability.

4.4 Hybrid Sparse Methods

While sparse attention, sparse FFN, and sparse MoE can independently reduce the verification cost of speculative decoding, each technique targets a different computational bottleneck and introduces distinct approximation errors. Therefore, it is crucial to evaluate whether combining multiple sparse mechanisms can achieve complementary benefits without causing systematic degradation in verification quality. We evaluate the performance of a hybrid sparse verification method with the proposed strategies. We use Deepseek-R1 as the target model and conduct speculative decoding with MTP. Basic length L_0 for SA is set to 4K. Deepseek-R1 has 61 layers, of which 30 layers are selected as anchor layers to provide retrieval results to other layers. Threshold τ for FFN is set to 0.05. Expert skipping budget m is set to 3. From a FLOPs perspective, it reduces the computational cost by approximately 36% compared with strict verification.

Table 5 reports the performance of strict verification and the proposed hybrid method with three-dimensional sparsity. Compared with the strict baseline, the hybrid method achieves higher scores on GovReport, 2WikiMQA, and HotpotQA, while maintaining comparable performance on GSM8K

Verification	s_e	GovReport	2WikiMQA	HotpotQA	GSM8K	Math	CollegeMath
		ROUGE	F1	F1	Acc.	Acc.	Acc.
Strict	0	26.90	77.39	73.43	98.0	82.0	58.0
SMoE ($m = 2$)	0.11	26.74	79.81	74.60	98.0	80.0	56.0
SMoE ($m = 3$)	0.16	26.73	78.52	75.03	97.0	87.0	57.0
SMoE ($m = 4$)	0.22	26.76	78.88	74.77	95.0	75.0	52.0

Table 4: Performance of verification with sparse MoE on different datasets with Deepseek-R1. "Strict" indicates standard speculative decoding with strict verification. Deepseek-R1 activates 8 experts per token on average, and s_e is defined as the ratio between the number of skipped activated experts per token and 8.

Verification	GovReport		2WikiMQA		HotpotQA		GSM8K		Math		CollegeMath	
	ROUGE	α	F1	α	F1	α	Acc.	α	Acc.	α	Acc.	α
Strict	26.90	2.44	77.39	2.71	73.43	2.67	98.0	2.84	82.0	2.85	58.0	2.89
Hybrid	27.40	2.42	79.82	2.70	74.57	2.66	96.0	2.81	81.0	2.84	53.0	2.89

Table 5: Performance of verification with hybrid method with the 3-dimensional sparsity with Deepseek-R1. "Strict" indicates standard speculative decoding with strict verification. "α" represents mean acceptance length.

and Math. We observe that the impact of sparse verification varies across tasks. Generation-oriented tasks such as summarization and open-domain QA exhibit relatively stable performance under moderate sparsity, whereas reasoning-intensive tasks, particularly mathematical problem solving, are more sensitive to sparsification. This suggests that tasks requiring precise token-level verification may demand stricter alignment between draft and target models.

4.5 Impact on Acceptance Length

Besides computational cost, the mean acceptance length is a crucial factor affecting the efficiency of speculative decoding. Therefore, we discuss the impact of these sparse verification methods on the average acceptance length in this section.

We report the mean acceptance length on each dataset in Table 5. The acceptance length is primarily determined by the alignment between the draft and target models. Since the draft model is trained to align with a non-sparse target model, sparsification in the target model can negatively affect this alignment. While sparsification effectively reduces verification cost, it also introduces distributional shifts between the draft and target models, which can negatively affect token acceptance. The mean acceptance length α shows a consistent but negligible reduction across most datasets, which has little effect on inference efficiency. Although minor numerical differences may exist due to different kernel implementations and low-level computation orders, we empirically observe no noticeable additional impact on acceptance behavior beyond the sparsification strategy itself.

5 Conclusion

This work investigates sparse verification in speculative decoding and demonstrates that substantial computational redundancy exists in the verification stage across multiple dimensions, including attention, feed-forward networks, and expert activation. By analyzing the structural characteristics of speculative verification, we show that sparsification strategies originally designed for standard decoding can be adapted to verification with careful control of acceptance behavior. We propose a unified framework that integrates multi-dimensional sparsity into speculative verification without requiring additional training or model modification. In particular, the framework incorporates adaptive expert skipping and retrieval reuse to reduce redundant computation while preserving verification correctness. Empirical results across diverse benchmarks indicate that sparse verification achieves consistent efficiency gains, with only a moderate impact on acceptance length under appropriate sparsity levels. Moreover, our Triton-based sparse attention kernel demonstrates that the proposed sparsity pattern can be effectively translated into practical wall-clock acceleration on modern GPUs.

Limitations

Although the proposed sparse verification framework achieves substantial efficiency gains with minimal accuracy degradation, it has several limitations. First, the effectiveness of sparse attention relies on the assumption that draft tokens within the same verification step attend to similar contextual regions. While this assumption holds in most cases, especially for nearby draft tokens, it may be violated in rare cases where draft branches diverge significantly, potentially leading to suboptimal block selection. Second, the sparsity hyperparameters, such as the base length L_0, sparsity coefficients, and expert skipping budgets, are determined through empirical calibration. Although we observe consistent behavior across datasets, these parameters may require tuning when applied to models with substantially different architectures or routing behaviors. Third, while sparse FFN and sparse MoE verification preserve performance under moderate sparsity, excessive sparsification can negatively affect tasks that require fine-grained token-level reasoning, such as mathematical problem solving. This suggests that the optimal sparsity level is task-dependent and may require adaptive control mechanisms. Although we implement a Triton-based sparse attention kernel that achieves substantial layer-wise latency reduction on H800 GPUs, the current system implementation still lacks a fully optimized end-to-end sparse execution framework for all verification components. In particular, sparse FFN and sparse MoE operations are currently implemented using standard dense operators with masking-based computation, which limits the achievable end-to-end acceleration. Developing fully hardware-aware sparse kernels and scheduling strategies for all sparse verification modules remains an important direction for future work.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*.

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. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Preprint*, arXiv:2101.03961.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning*.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive kv cache compression for llms. In *12th International Conference on Learning Representations, ICLR 2024*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Canwen Xu, Nan Duan, Jian Yin, and Julian McAuley. 2023. Longcoder: A long-range pre-trained language model for code completion. In *International Conference on Machine Learning*, pages 12098–12107. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. 2024. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1582–1595,

Mexico City, Mexico. Association for Computational Linguistics.

D Hendrycks. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *Preprint*, arXiv:2104.02112.

Woosuk Kwon, Sehoon Kim, Michael W. Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Preprint*, arXiv:2204.09656.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024a. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*.

Tianyang Liu, Canwen Xu, and Julian McAuley. 2024. Repobench: Benchmarking repository-level code auto-completion systems. In *The Twelfth International Conference on Learning Representations*.

Zeyu Liu, Tim Dettmers, Xi Lin, Veselin Stoyanov, and Xian Li. 2023. Towards a unified view of sparse feed-forward network in pretraining large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15038–15061.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, and 6 others. 2025. Moba: Mixture of block attention for long-context llms. *Preprint*, arXiv:2502.13189.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. 2024. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6159–6172, Bangkok, Thailand. Association for Computational Linguistics.

Meta-AI. 2024. Introducing llama 3.1: Our most capable models to date.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, and 1 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. 2022. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. *Preprint*, arXiv:2201.05596.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, and 1 others. 2021. Hash layers for large sparse models. *advances in neural information processing systems*, 34:17555–17566.

Ranajoy Sadhukhan, Jian Chen, Zhuoming Chen, Vashisth Tiwari, Ruihang Lai, Jinyuan Shi, Ian En-Hsu Yen, Avner May, Tianqi Chen, and Beidi Chen. 2024. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. In *The Thirteenth International Conference on Learning Representations*.

Apoorv Saxena. 2023. Prompt lookup decoding.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2024. Powerinfer: Fast large language model serving with a consumer-grade gpu. *Preprint*, arXiv:2312.12456.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024a. Quest: Query-aware sparsity for efficient long-context llm inference. In *International Conference on Machine Learning*, pages 47901–47911. PMLR.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024b. Mathscale: Scaling instruction tuning for mathematical reasoning. *Preprint*, arXiv:2403.02884.

Jikai Wang, Yi Su, Juntao Li, Qingrong Xia, Zi Ye, Xinyu Duan, Zhefeng Wang, and Min Zhang. 2024. Opt-tree: Speculative decoding with adaptive draft tree structure. *Transactions of the Association for Computational Linguistics*, 13:188–199.

Xinyi Wu, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. 2025. On the emergence of position bias in transformers. In *Forty-second International Conference on Machine Learning*.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. 2023. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. 2024. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7655–7671, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. 2018a. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458.

Penghui Yang, Cunxiao Du, Fengzhuo Zhang, Haonan Wang, Tianyu Pang, Chao Du, and Bo An. 2025. Longspec: Long-context speculative decoding with efficient drafting and verification. *arXiv preprint arXiv:2502.17421*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018b. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, and 1 others. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

A Related Work

Sparse Inference has been explored in multiple dimensions of language models. KV Cache Eviction (Zhang et al., 2023; Ge et al., 2024; Xiao et al., 2024; Tang et al., 2024a; Li et al., 2024a) has emerged as an effective approach to mitigate the memory and latency overhead of large language model inference as the context length continues to grow. H_2O (Zhang et al., 2023) proposes an oracle-based method that retains heavy-hitter tokens with the highest attention importance. Quest (Tang et al., 2024a) introduces a query-aware mechanism that dynamically selects important tokens to reduce redundant cache entries. NSA (Yuan et al., 2025) designs hardware-aligned, natively trainable sparse attention patterns to reduce computation and memory costs while maintaining model accuracy.

Sparse FFN methods focus on reducing computation in feed-forward layers. Early approaches (Roller et al., 2021; Liu et al., 2023) exploit structured sparsity to activate only a subset of parameters for each token. MoE models (Shazeer et al., 2017; Fedus et al., 2022) route tokens to a small number of experts, significantly reducing per-token FLOPs while maintaining model capacity. Subsequent works further improve efficiency by refining routing strategies and expert utilization, including expert pruning, load balancing, and dynamic expert selection during inference (Rajbhandari et al., 2022; Lu et al., 2024).

Speculative Decoding (Stern et al., 2018; Xia et al., 2023, 2024) follows a draft-verify paradigm to achieve lossless acceleration in autoregressive generation. At each decoding step, a lightweight drafter first drafts several candidate tokens, which are then verified in parallel by the target model. Some studies (Leviathan et al., 2023; Cai et al., 2024; Li et al., 2024b) employ independently trained small models or auxiliary modules as draft generators, while others (Saxena, 2023; Fu et al., 2024; He et al., 2024) retrieve drafts from pre-constructed draft pools. The draft structures have evolved from simple n-grams (Leviathan et al., 2023; Fu et al., 2024) to more sophisticated draft trees (Li et al., 2024b; Wang et al., 2024). Increasing the draft length imposes additional computational burden during verification, particularly under high-load conditions such as long-context scenarios. While prior studies (Sadhukhan et al., 2024; Yang et al., 2025) have investigated KV cache compression in the draft stage for long contexts, the target model's KV cache during verification is much larger and thus represents the key bottleneck.

B Impact of the Number of Calibration Samples

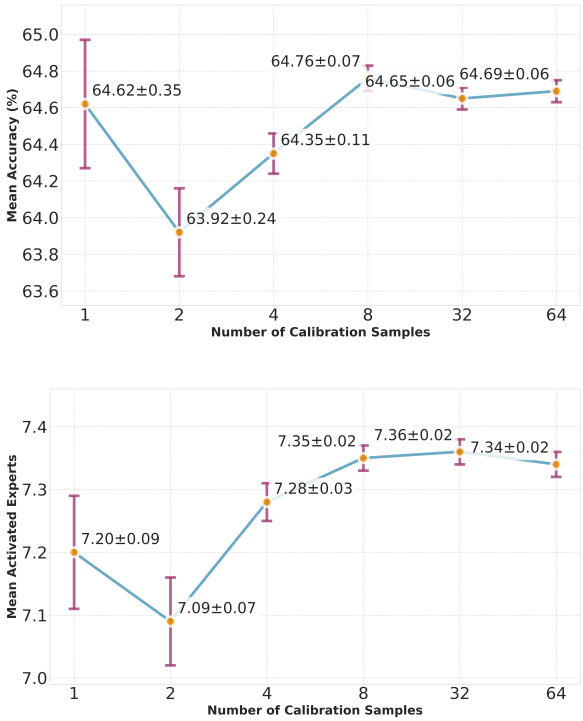

Figure 4: The average accuracy and average number of activation experts per token across five datasets vary with the number of calibration samples. The vertical line at each point represents the standard deviation.