# SeerAttention: Self-distilled Attention Gating for Efficient Long-context Prefilling

**Yizhao Gao**[1*†] **Zhichen Zeng**[2*†] **Dayou Du**[3†] **Shijie Cao**[4‡] **Peiyuan Zhou**[5]
**Jiaxing Qi**[5] **Junjie Lai**[5] **Hayden Kwok-Hay So**[1] **Ting Cao**[6] **Fan Yang**[4] **Mao Yang**[4]
[1]University of Hong Kong [2]University of Washington
[3]University of Edinburgh [4]Microsoft Research [5]NVIDIA [6]Tsinghua University

## Abstract

Attention is the cornerstone of modern Large Language Models (LLMs). Yet its quadratic complexity hinders efficiency and scalability, especially for long-context processing. A promising approach is to leverage sparsity in attention. However, existing sparsity-based solutions predominantly rely on *predefined patterns or heuristics* at the attention head level, struggling to adapt dynamically to different contexts efficiently. We propose SeerAttention, a simple yet effective attention mechanism that directly learns the block-level attention sparsity from the LLM itself. Inspired by the gating mechanism in Mixture of Experts (MoE), SeerAttention augments the conventional attention with a **learnable gate** that **selectively activates important blocks** within the attention map. Specifically, the gate first pools the query (Q) and key (K) tensors along the sequence dimension and processes them through learnable linear layers. The resulting matrices are then multiplied together to produce the gating scores, which are used to predict block-level attention sparsity. Combined with our block-sparse FlashAttention kernel, SeerAttention can achieve significant speedup on GPUs. When applied to pre-trained LLMs, SeerAttention only requires training the gate parameters in a lightweight self-distillation manner, allowing rapid convergence. Our evaluation results demonstrate that SeerAttention achieves better model accuracy and lower latency for long-context pre-filling compared to prior methods. Code is available at: https://github.com/microsoft/SeerAttention.

## 1 Introduction

Attention is a fundamental mechanism in transformer-based LLMs [51]. Despite its effectiveness, the quadratic complexity of attention demands substantial computation and memory resources, limiting the scalability and efficiency of LLMs, especially for long-context windows. This challenge has become an active research topic in the community. One potential solution is to replace the quadratic attention with cheaper architectures like linear attention or recurrent networks [30, 20, 40, 47] with subquadratic complexity. While these approaches are more efficient, the majority of state-of-the-art large language models (LLMs) continue to use full attention to achieve better performance.

A promising approach with increasing interests is to leverage sparsity in attention. Sparsity commonly exists in attention maps, and it becomes more prominent in longer contexts. In certain LLM attention heads, the sparsity ratio can reach 95% or even 99%, posing great opportunities for efficiency improvements. However, prior post-training methods often rely on predefined sparsity patterns or heuristics to approximate the attention mechanism [28, 18, 32, 64, 22, 54]. The sparsity observed in

*Equal Contribution.
†Work partially done during the internship at Microsoft Research.
‡Corresponding author: shijiecao@microsoft.com.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

attention maps varies significantly across different models, input contexts and attention heads, making predefined patterns or heuristics insufficient. On the other hand, pre-training a sparse attention model from scratch like Native Sparse Attention [58] or MoBA [38] is costly and can not be directly applied to other dense pre-trained models.

In this paper, we introduce SeerAttention, a simple yet effective post-training distillation method that brings tangible attention sparsity to any full-attention models through self-distillation without relying on predefined sparsity patterns. To achieve this, SeerAttention augments conventional attention with a learnable gate called *AttnGate* that selectively activates a small subset of important blocks in the attention map, drawing inspiration from the gating mechanism in MoE [45]. The AttnGates are trained with the 2D block-level sparsity ground truth generated by the original LLMs. Importantly, this distillation process only requires learning the gating parameters, while all other model parameters remain fixed. This leads to fast training process as only the newly added gate weights requires to compute gradient. In this way, without relying on human heuristic observations, users can obtain tailored AttnGates for different models.

Our results demonstrate that SeerAttention surpasses state-of-the-art post-training sparse attention methods like MInference [28], MoA [18] and DuoAttention [54] in terms of long context model accuracy and pre-filling latency. SeerAttention achieves highly linear speedup over dense configurations, delivering a 7.3× speedup with 90% sparsity on sequences of 128k. Notably, in contrast to previous methods that require careful calibration of sparse configuration for different heads, SeerAttention offers strong capabilities of adaptation to different heads and contexts. Remarkably, on top of block-sparse pattern, SeerAttention exhibits the ability to learn more diverse patterns, including A-shape and Vertical-Slash, further demonstrating its versatility and performance.

Our contributions can be summarized as follows:

- We propose SeerAttention, an innovative learnable attention gating mechanism to enhance efficiency for long-context LLMs.
- We have developed a self-distillation training scheme to efficiently train the AttnGate, enabling it to learn the intrinsic sparsity of a pre-trained model.
- Experiments show that SeerAttention outperforms previous approaches, offering adaptability to various context lengths and sparsity ratios.

## 2 Background and Related Works

**Powerful but Complex Attention in Transformer.** The advent of attention mechanisms, particularly within the Transformer architecture [51], marked a significant advancement in natural language processing. Attention enables improved handling of long-range dependencies and a better understanding of context by attending each token to every other token in the sequence, resulting in a quadratic memory and time complexity $O(n^2)$, where $n$ is the sequence length. This presents a significant challenge as the community moves towards LLMs that can process increasingly longer contexts. Many studies explore alternative attention mechanisms to mitigate this complexity. The Reformer architecture [31] reduces the complexity to $O(n \log n)$ and the linear attention mechanism [30, 57] further decreases complexity to $O(n)$. Recently, there has been a trend of revisiting recurrent neural networks, leading to the proposal of new architectural frameworks such as RWKV [40], RetNet [47], and Mamba [20]. Despite their promise of efficiency, these methods struggle to match the performance of full attention mechanisms, particularly with larger models and longer contexts.

**Intrinsic but Dynamic Sparsity in Attention.** Attention mechanisms inherently exhibit sparsity, which arises from the attention map $\mathbf{A}$ generated by $\mathbf{Q}$ and $\mathbf{K}$: $\mathbf{A} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^{\mathbf{T}}/\sqrt{d})$. The softmax function often produces a multitude of negligible scores that can be treated as zeros without impacting model accuracy [59, 35, 52, 7, 36]. Attention sparsity becomes more pronounced with longer contexts, presenting opportunities to optimize inference speed. Unfortunately, this sparsity is dynamic, varying across different context inputs and attention heads, each displaying distinct sparsity locations and ratios. Prior research has attempted to approximate attention sparsity using predefined patterns and heuristics [18, 28] for different attention heads. Yet, these methods lack generality and often rely on handcrafted features, struggling to fully capture the sparsity behavior of attention mechanisms. The dynamic and input-dependent nature of attention sparsity echoes the principles of

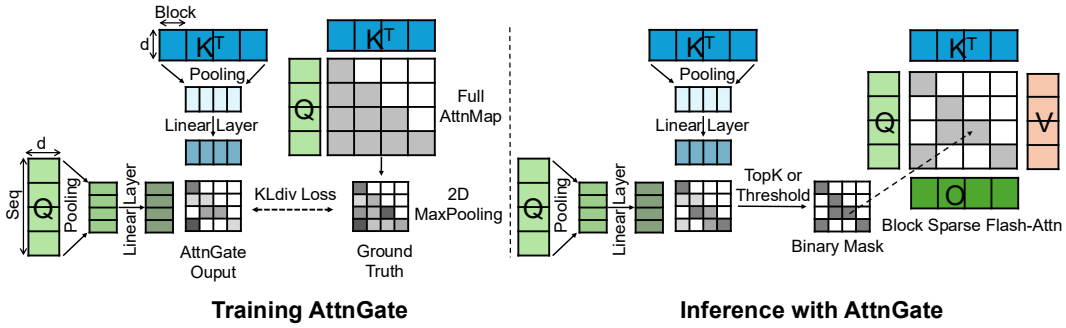



Figure 1: **Overall of SeerAttention.** The AttnGate in SeerAttention first pools the Q and K tensors in sequence dimension and passes through learnable linear layers. The AttnGate output are trained to mimic the 2D maxpooled results of the pre-trained model. During inference, it applies TopK or Thresholding on the AttnGate output to locate activate blocks.

Mixture of Experts (MoE) models [45, 17] suggesting that sparsity should ideally be learned directly from data within the model itself. This approach would allow models to adaptively harness sparsity, improving efficiency while maintaining accuracy.

**Long-Context LLM Optimizations.** The ability to process long contexts is crucial for large language models (LLMs) as it enables them to retain and utilize more extensive information. However, it comes with substantial computational and memory costs. Various research efforts have explored different strategies to optimize long-context processing. One major direction is improving prefill efficiency, where techniques such as prompt compression [27, 39, 8] and sparse attention [28, 18, 1, 62]. Another approach focuses on optimizing the decoding phase by introducing sparse loading mechanisms [56, 6]. Additionally, several methods aim to compress the KV cache, including KV cache sharing [3, 5], KV eviction policies [63, 34, 19], and KV quantization [37, 25, 14, 61].

## 3 SeerAttention

SeerAttention adopts a fully learning-based approach to adaptively identify attention sparsity in LLMs and leverages the learned sparsity for efficient inference. To ensure efficiency on modern hardware like GPUs, we focus on learning block sparsity, which can seamlessly integrate with the tiling computation scheme of FlashAttention [13, 12]. Figure 1 illustrates the overall diagram of SeerAttention, which augments conventional attention with a learnable gating module, termed *Attention Gate* (AttnGate). The AttnGate modules contain learnable parameters (linear layers) and are distilled to mimic the 2D-Maxpooled results of the attention maps. At inference time, the AttnGate can predict the block-level sparsity for the subsequent attention computation with a block-sparse FlashAttention kernel, which significantly enhances performance by reducing I/O and computation overhead.

### 3.1 Attention Gate Design

The AttnGate module is designed to learn block-wise information with minimal overhead. It takes the original matrices $\mathbf{Q}$ and $\mathbf{K}$ as inputs and downsamples them using pooling operations along the sequence dimension. As shown in Figure 1, for a given attention head, the sizes of the pooled $\mathbf{Q}$ and $\mathbf{K}$ become $[seq/B, d]$, where $B$ is the kernel and stride size of the pooling operation (non-overlapped blocks). The downsampled $\mathbf{Q}$ and $\mathbf{K}$ are then processed through a linear layer and multiplied together, similar to the standard attention operation. This results in a matrix of size $[seq/B, seq/B]$, where each element corresponds to one block in the original full attention map. With a typical block size of 64, the output of the AttnGate module is only $\frac{1}{4096}$ the size of the original attention map, making it super efficient to compute. To its simplest form, the AttnGate output soft score can be expressed as:

$$\mathbf{Q}_c = \mathrm{RoPE}\Big(W_q \ \mathrm{concat}_{i=1}^{m_q} P_i^{(q)}(\mathbf{Q}_{nope})\Big), \tag{1a}$$

$$\mathbf{K}_c = \mathrm{RoPE}\Big(W_k \ \mathrm{concat}_{j=1}^{m_k} P_j^{(k)}(\mathbf{K}_{nope})\Big), \tag{1b}$$

$$\mathbf{O} = \mathrm{softmax}(\mathbf{Q}_c \, \mathbf{K}_c^{\top} / \sqrt{d}). \tag{1c}$$

where $P_i^{(q)}$ and $P_j^{(k)}$ represents different pooling operations for $\mathbf{Q}$ and $\mathbf{K}$, and $d$ is the hidden size of the tensors similar to attention computation. The detailed algorithm will be explained as follows.

**Pooling Method Selection.** Pooling operations downsample tensors and may lead to information loss. To better preserve the characteristics of the attention tensors, SeerAttention allows different pooling methods to be composed for $\mathbf{Q}$ and $\mathbf{K}$. Specifically, we consider average, max, and min pooling. When applying multiple pooling methods to either $\mathbf{Q}$ or $\mathbf{K}$, the resulting pooled tensors are concatenated along the hidden dimension before being fed into the subsequent linear layer. Figure 2 presents the test perplexity on the PG19 [42] dataset for the top 15 pooling combinations using the LLaMA-3.1-8B model. We observe that applying AvgPooling on $\mathbf{Q}$ and a combination of Max, Min, and AvgPooling on $\mathbf{K}$ yields the best perplexity across different sparsity ratios. This trend may relate to prior findings in LLM quantization, where $\mathbf{K}$ tensors exhibit more outliers. Incorporating Max and Min pooling thus helps capture these extreme activations, leading to richer feature representations after pooling.

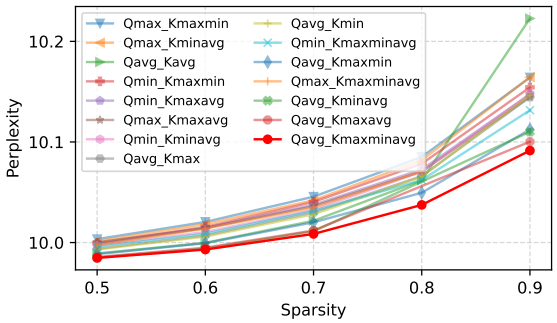



Figure 2: Test perplexity of different pooling combinations on PG19. The best configuration applies AvgPooling on $\mathbf{Q}$ and a combination of **Max**, **Min**, and **AvgPooling** on $\mathbf{K}$.

**Length Extrapolation of AttnGate using Positional Encoding.** Recent state-of-the-art LLMs typically employ positional encoding (PE) such as RoPE [46] to encode positional information. If the AttnGate relies solely on the original RoPE in the model, i.e., feeding the AttnGate with $\mathbf{Q}_{rope}$ and $\mathbf{K}_{rope}$, the positial information can possibly be damaged because of the pooling operation. This compromises the length extrapolate ability of AttnGate during distillation. To address this issue, we re-apply block-level PE in AttnGate with input wihtout PE, $\mathbf{Q}_{nope}$ and $\mathbf{K}_{nope}$, (shown in Equation 9). To represent the block-level information, the RoPE in AttnGate uses a reduced $\theta' = \theta/B$, where $\theta$ is the original RoPE theta of the LLM.

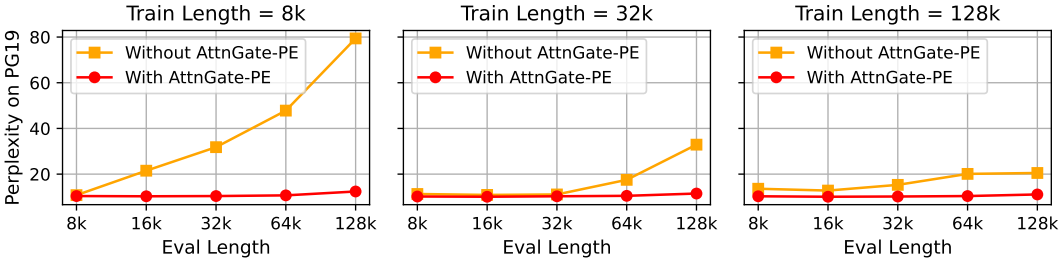



Figure 3: **Perplexity Comparison Between two PE setting in AttnGate on PG19 dataset.** The block-level RoPE in AttnGate allows it to effectively learn the block-level positional information, resulting better test performance for different context length. Without the AttnGate PE, it fails to deliver reasonable results with data longer than training length.

Figure 3 presents the test perplexity results with and without the block-level RoPE design in AttnGate. The results indicate that without this block-level RoPE design, AttnGate fails to perform adequately on evaluation data longer than 8k when trained with 8k length data. Similarly, when trained with 32k

length data, it does not perform well on 128k length data. However, with the additional block-level RoPE, AttnGate can extrapolate to different context lengths, significantly enhancing the model performance and training efficiency.

### 3.2 AttnGate Training

While the introduced SeerAttention architecture is straightforward, training presents challenges. Jointly training the gate and model from scratch, as in MoE, is costly and difficult. Fortunately, unlike MoE, where gating network must learn expert selection from scratch, the AttnGate in SeerAttention has a ground truth from standard attention for distillation.

**Obtaining the Ground Truth.** We use the 2D-MaxPooled attention map from full attention as ground truth to distill AttnGate, as illustrated in Figure 1. Semantically, it means that only when all the attention score in a block is small, the 2D-MaxPooled results will be small, which aligned with the block-sparse definition. However, obtaining the max-pooled attention map for training is non-trivial especially in long-context scenarios due to quadratic memory consumption of the intermediate $\mathbf{QK}^T$ results. To address this challenge, we customize an efficient kernel that directly outputs the MaxPooled attention map ground truth by modifying FlashAttention kernel but largely reuses its original computation flow. The detailed design are explained in A.1.

**Loss Function.** The Kullback-Leibler divergence loss [29] is use to distill the AttnGate. Since AttnGate uses softmax in output similar to full attention computation, the row summation of gating score will always be 1. KL-divergence loss allows the training process to focus on mimicking the attention distribution instead of absolute magnitude like Mean-square-error loss. The overall distillation process can be expressed as:

$$\mathbf{gt} = \mathrm{MaxPool2D}\Big(\mathrm{softmax}\big(\tfrac{\mathbf{Q}_{rope}\mathbf{K}_{rope}^T}{\sqrt{d}}\big)\Big), \tag{2}$$

$$\mathbf{o} = \mathrm{AttnGate}(\mathbf{Q}_{nope}, \mathbf{K}_{nope}), \tag{3}$$

$$\mathbf{loss} = D_{KL}\big(\mathbf{gt} \parallel \mathbf{o}\big). \tag{4}$$

### 3.3 Inference with SeerAttention

After self-distillation training process, SeerAttention can utilizes the trained AttnGate to generate a gating score for each block within the full attention mechanism. These scores are then used to select the final activated sparse blocks. Further combined with our backend Block-sparse FlashAttention kernel, SeerAttention can achieve significant speedup for long-context prefilling while maintaining high accuracy.

**Generating Binary Block Mask.** SeerAttention provides the flexibility to convert the floating-point gating scores $\mathbf{o}$ into a final binary block mask using either the TopK or Thresholding methods. If using the TopK method, the $k$ blocks with the highest scores in each row are selected.

$$b_{ij} = \begin{cases} 1 & \text{if } j \in \mathrm{TopK}(\mathbf{o}_i, k).\mathrm{index}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Alternatively, users can activate blocks with score exceeding a threshold. This can further saving the need of sorting the AttnGate output score.

$$b = \mathbf{o} > threshold \tag{6}$$

Notably, once AttnGate is trained, for the inference stges, we can adjust the TopK ratio or threshold-based at test time to achieve various trade-offs.

**Block Sparse Flash-Attn Kernel.** In designing the Block Sparse Flash-Attention kernel, the block size of AttnGate is aligned with the tiling size used in Flash-Attention, typically 64 or 128. By doing so, we can create a customized block-sparse Flash-Attention kernel that leverages the binary block mask generated by AttnGate to selectively skip the I/O and computation for unactivated blocks. This approach is highly efficient on modern GPUs, as it optimizes the processing of sparse data at the block level rather than dealing with fine-grained element-wise level, leading to significant performance gains.

# 4 Experiments

In this section, we evaluate both the accuracy and efficiency of SeerAttention. In our current experiments, block-size $B$ for the AttnGate and sparse kernel is fixed at 64 and AttnGate solely applies in the prefill stage.

**Models, Baselines and Tasks.** We apply SeerAttention to the pre-trained models Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct [15], as well as Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct and Qwen2.5-32B-Instruct [55] in the experiments. We compare SeerAttention with three state-of-the-art sparse attention methods, MoA [18], MInference [28], and DuoAttention [54]. It shoule be noted that Llama-3.1-8B-Instruct is the only model that all the other methods provide official support/configuration. Thus, we only compare with them using Llama-3.1-8B-Instruct model. MoA uses an offline search scheme to apply static sparse patterns across different attention heads. In our experiment, we adopt their official implementation with "KV Sparsity" in 0.5 which means "Attention Sparsity" in 0.35. MInference dynamically generates sparse indices using heuristic methods for each head based on pre-defined sparse patterns. We used their official configuration for Llama-3.1-8B-Instruct model, where all attention heads choose the "Vertical-Slash" sparsity pattern. DuoAttention differentiates some attention heads as streaming heads [53] while keep the rest as dense heads. In the following experiment, we adopted their official setup for Llama-3.1-8B-Instruct model with 50% head as streaming heads. We evaluate the performance using on two long context benchmarks: LongBench [4] and RULER [26], and 4 short-context task from Open LLM Leaderboard [50]: HellaSwag [60], MMLU [24], ARC-challenge [9], GSM8K [10]. For long-context benchmark, we follow a similar practice in Star Attention [2] and SCBench [33] that only applies sparsity in context rather than question in SeerAttention. All the evaluation were run on A100 GPUs.

**Distillation Training Setup.** We use the RedPajama [11] dataset for AttnGate distillation, which are chunked into 64k with BOS and EOS tokens. Our training employs a learning rate of 1e-3 with cosine decay. We set the global batch size to 16 and conduct training for only 500 steps, leveraging DeepSpeed [43] stage 2 optimization on A100 GPUs. As only AttnGate parameters are learned and updated, the distillation process can be completed within around 40 A100 hours for 7B or 8B models. To prevent the quadratic memory explosion that occurs when saving the intermediate attention map for ground truth generation, we customized a FlashAttention kernel. This kernel directly outputs the 2D max-pooled ground truth on top of the original attention computation. Further details about this kernel can be found in A.1.

## 4.1 Accuracy of Evaluation

**LongBench Evaluation.** LongBench is a long-context understanding benchmark. We compare with those of MoA, MInference, and DuoAttention using the Llama-3.1-8B-Instruct model. DuoAttention

Table 1: LongBench Results on Llama and Qwen models.

| Model | Method | 0-4k | 4-8k | 8k+ | Avg. Acc. | Avg. Sparsity |
|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | Full Attention | 55.32 | 53.98 | 52.9 | 54.07 | 0.0 |
| | MInference | 55.23 | 53.78 | 52.18 | 53.73 | 0.31 |
| | MoA | 50.74 | 49.84 | 51.89 | 50.82 | 0.35 |
| | DuoAttention | 53.77 | 52.17 | 51.27 | 52.40 | 0.5* |
| | **SeerAttention** | **55.43** | **54.49** | **52.69** | **54.20** | 0.50 |
| Llama-3.1-70B-Instruct | Full Attention | 58.32 | 57.29 | 57.32 | 57.64 | 0.0 |
| | SeerAttention | 57.83 | 56.07 | 55.61 | 56.50 | 0.62 |
| Qwen2.5-7B | Full Attention | 53.72 | 50.52 | 48.21 | 50.81 | 0.0 |
| | SeerAttention | 53.94 | 50.78 | 48.73 | 51.80 | 0.55 |
| Qwen2.5-14B | Full Attention | 54.64 | 53.16 | 50.68 | 52.83 | 0.0 |
| | SeerAttention | 54.55 | 52.84 | 51.21 | 52.86 | 0.55 |
| Qwen2.5-32B | Full Attention | 56.29 | 52.17 | 51.83 | 53.43 | 0.00 |
| | SeerAttention | 56.43 | 51.93 | 52.01 | 53.45 | 0.56 |

* 50% streaming heads, the real sparsity <50%

uses 50% of the heads as streaming heads and 50% as dense heads. For streaming heads, the attention only occurs in the attention sink and recent tokens. As a result, it is not less than 50% sparsity overall. In this benchmark test, SeerAttention employs a threshold of 2e-3 for all AttnGates. With the same threshold, different attention gates can exhibit varying sparsity ratios, and longer context data tends to be sparser. This approach allows for a more adaptive allocation of sparsity. As demonstrated in Table 1, SeerAttention consistently outperforms other methods across various test lengths. Notably, in the 0-4k and 4-8k tests, our score surpasses even the dense baseline. This may be attributed to AttnGate filtering out noisy attention in certain cases. Furthermore, SeerAttention achieves the highest average score and the highest average sparsity across all tests. For other models except Llama-3.1-8B-Instruct, SeerAttention demonstrates similar accuracy performance compared to dense baseline with $> 50\%$ averaged sparsity. The 8k+ split typically has an sparsity around 70%.

Table 2: RULER Benchmark Results on Llama-3.1-8B-Instruct Model.

| Methods | 4k | 8k | 16k | 32k | 64k | 128k | Average Accuracy | Average Speedup |
|---|---|---|---|---|---|---|---|---|
| Full Attention | 95.53 | 92.37 | 92.01 | 87.63 | 84.39 | 76.26 | 88.01 | 1.00 |
| MInference | 95.53 | 92.64 | 91.37 | 85.71 | 83.24 | 67.02 | 85.92 | 0.83 |
| DuoAttention | **95.64** | 92.08 | 90.71 | 84.75 | 83.24 | **75.32** | 86.96 | 1.09 |
| **SeerAttention** | 95.53 | **92.71** | **92.02** | **88.49** | **83.48** | 73.37 | **87.60** | **1.41** |

**RULER Evaluation.** RULER is a long-context LLM evaluation benchmark consisting of 13 challenging sub-tasks. It generates tests with data sizes ranging from 4k to 128k. In this experiment, SeerAttention employs a threshold of 5e-4, which allows it to automatically adapt sparsity from approximately 10% for 4k data to around 85% for 128k data. Due to out-of-memory (OOM) issues in some tests, MoA was excluded from this benchmark. Table 2 provides detailed accuracy results across different evaluation lengths. It is evident that SeerAttention achieves the best accuracy in most tests (8k-64k). For the 128k test, DuoAttention has less than 50% sparsity, whereas SeerAttention maintains an sparsity higher than 80%, which accounts for the slightly lower performance. SeerAttention also attains the highest average accuracy compared to other models (only 0.41% lower than the dense baseline) while delivering the highest average end-to-end speedup ($1.41\times$) in prefilling time.

Table 3: Short Context Tests on Llama-3.1-8B-Instruct Model

| | MMLU | HellaS. | ARC-c | GSM-8K |
|---|---|---|---|---|
| Full Attention | 68.1 | 80.1 | 60.7 | 75.7 |
| SeerAttention | 67.9 | 79.8 | 60.2 | 75.6 |
| Avg Sparsity | 3.4 | 50.4 | 26 | 52.1 |
| Avg Seqlens | 118 | 840 | 395 | 872 |

**Short Context Test.** For short context input, attention contributes a smaller proportion in the total runtime. Consequently, sparse attention does not significantly enhance latency performance. Nevertheless, we evaluate SeerAttention accuracy performance under a very high threshold 3e-2 to achieve high sparsity. The results, as shown in 3, indicate that SeerAttention exhibits negligible accuracy loss. For instance, with an average sequence length of 872 in the GSM-8K task, SeerAttention achieves only 0.1% degradation in accuracy with 52% averaged sparsity.

### 4.2 Efficiency Evaluation

We evaluate the efficiency of SeerAttention using our implementation of CUDA kernels. We evaluate the kernel-level as well as end-to-end speedup using a Llama-3.1-8B-Instruct on a single A100 GPU. Results are compared to FlashAttention-2 (dense baseline), MoA, MInference and DuoAttention.

#### 4.2.1 Kernel evaluation

**Negligible Overhead incurred by AttnGate.** 4 shows the kernel-level latency breakdown of SeerAttention. It demonstrates that the overhead introduced by the AttnGate during inference is minimal. For instance, at a context length of 32K and a sparsity of 0.5, the AttnGate contributes

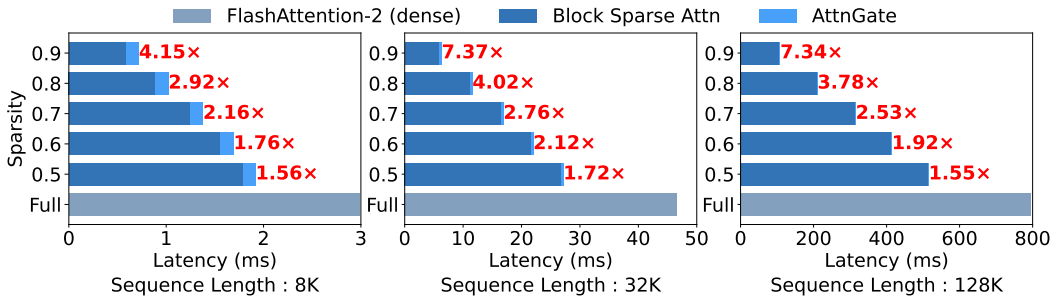



Figure 4: **SeerAttention Speedup over FlashAttention-2 at the Kernel Level.** The latency overhead from AttnGates is minimal. Our block-sparse attention kernel achieves highly linear speedup over dense configurations, delivering a 7.3× speedup with 90% sparsity on sequences of 128k. The AttnGate overhead amost diminishes in 128k context length.

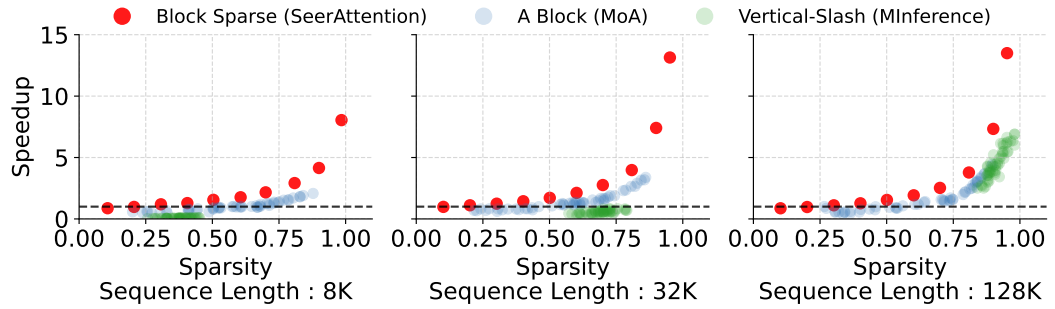



Figure 5: Kernel-level Speedup Comparison Between Different Works. SeerAttention translates sparsity to speedup more effectively.

only 1% to the total latency of an attention layer. In the cases of 128K sequence length, the relative overhead almost diminishes.

**Block-sparse FlashAttention Kernel Speedup.** Figure 4 also shows that our kernel exibits linear speedup over various sparsity levels. At a sequence length of 128K with 90% sparsity, SeerAttention achieves a speedup of 7.3× compared with FlashAttention-2 (full attention) on a single A100 GPU. This demonstrates the effectiveness of the block-level sparsity employed by SeerAttention, which is highly efficient on GPUs and translates into high speedup.

**Kernel-level Comparison with Related Works.** We compare the kernel-level speedup of SeerAttention with MoA and MInference. MInference uses offline calibration to identify a pre-defined sparse pattern for each layer. For Llama-3.1-8B-Instruct model, MInference consistently uses "Vertical-slash" pattern across all layers. During runtime, MInference will dynamically generate non-zero indices based on their approximation algorithm. On the other hand, MoA uses "A-shape" blocks as their sparse pattern and calibrate the shape parameters offline under given sparsity constraint. DuoAttention is omitted in kernel-level comparison as it's a combination between streaming and dense head, whose performance is a mixture results of block sparse attention and dense FlashAttention.

Figure 5 shows the sparsity v.s. speedup plots of different methods on 8k, 32k, 128k sequences length, where the speedup baseline is FlashAttention-2. The kernel-level sparsity statistics were collected from PG19 datasets. For MoA, we generated the sparse configurations under their 0.5 overall "KV-sparsity" constraints, which corresponds to an average of 0.35 sparsity in attention. The results demonstrates that the block-sparse attention kernel used in SeerAttention outperforms both MoA and MInference in most cases.

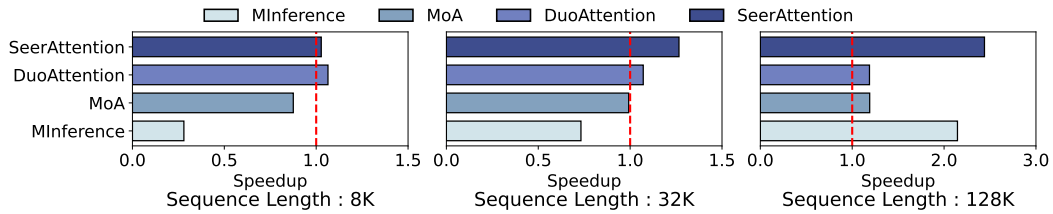



Figure 6: **Comparing Prefilling Time Speedup on RULER Test Setting.** SeerAttention outperforms related works in most long-context data scenarios ($\geq$ 16k). For longer context data, the attention mechanism constitutes a larger proportion of the total runtime, allowing sparse methods to achieve better speedup. Overall, SeerAttention achieves the highest average speedup ($1.41\times$) while maintaining the best average accuracy under this RULER benchmark setting.

#### 4.2.2 End-to-end Speedup Comparison.

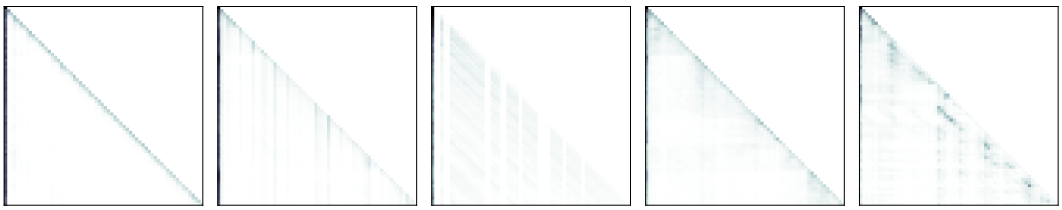
To assess the end-to-end speedup of our method, we measured the average prefilling time, or time-to-first-token (TTFT), using the Llama-3.1-8B-Instruct model on the RULER test discussed above. Since attention takes up more runtime with longer contexts, all methods generally achieve better speedup with longer context lengths. It should be noted that SeerAttention uses an identical threshold across all tests in RULER, automatically adjusting to higher sparsity for longer contexts (ranging from approximately 10% sparsity for 4k to around 85% sparsity for 128k). This approach results in an end-to-end prefilling speedup of up to $2.43\times$ on 128k length. On the other hand, MInference experiences a slowdown with data sizes less than 64k due to significant overhead in searching for sparse indices during runtime. It is feasible for SeerAttention to adjust to higher sparsity to achieve greater speedup in shorter contexts as a tradeoff. Nevertheless, SeerAttention delivered the highest average accuracy (87.6) and the greatest average speedup ($1.41\times$) in this RULER benchmark setting.

### 4.3 Training Cost and Parameter Overhead.

The additional training cost of SeerAttention is modest. On LLaMA-3-8B-Instruct, it requires about 40 A100 GPU hours, similar to or lower than DuoAttention. In comparison, MInference calibrates faster but shows lower accuracy and slower inference. SeerAttention introduces about 101 M trainable parameters, roughly 1.3 % of the model, and adds less than 5 % memory and latency overhead compared with FlashAttention, supported by our customized distillation kernels. The gating module also scales well to larger models, such as 503 MB for LLaMA-3.1-70B and 252 MB for DeepSeek-R1-Distill-Qwen-32B, showing that the method remains lightweight and scalable for large deployments.

### 4.4 Visualization of Learned Attention Maps.

The AttnGate module automatically learns diverse sparse patterns without any prior knowledge or heuristics. Figure 7 shows several example outputs from AttnGate, including (a) "A-shape," or streaming head (b) "Vertical," (c) "Slash" with empty vertical spaces, (d) block sparsity along the diagonal, and (e) random patterns. These patterns not only encompass but also extend beyond those observed in previous works such as MoA and MInference, showcasing the versatillty of our learning based methods.



(a) (b) (c) (d) (e)

Figure 7: Visualization of the AttnGate's outputs.

## 5 Conclusion and Future Work

This paper presents SeerAttention, a new attention mechanism that learns and leverages the intrinsic sparsity in attention to boost long-context LLMs. SeerAttention learns the attention sparsity from the LLM itself with a lightweight self-distillation approach. Our experiments demonstrate that SeerAttention outperforms previous approaches in terms of long context model accuracy and pre-filling latency. For future work, there are several promising directions to explore for improving and expanding the capabilities of SeerAttention. One key area is enhancing the training methodologies for SeerAttention, such as applying SeerAttention in long-context continued pre-training with more training tokens to achieve higher sparsity without sacrificing accuracy. Another important avenue is applying SeerAttention in the decoding stage, especially for long-CoT.

## Acknowledgments

We gratefully acknowledge Prof. Ang Li from University of Washington for his help with the paper revision and valuable feedback.

## References

[1] Shantanu Acharya, Fei Jia, and Boris Ginsburg. Star attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*, 2024.

[2] Shantanu Acharya, Fei Jia, and Boris Ginsburg. Star attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*, 2024.

[3] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[4] 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. *arXiv preprint arXiv:2308.14508*, 2023.

[5] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. Reducing transformer key-value cache size with cross-layer attention. *arXiv preprint arXiv:2405.12981*, 2024.

[6] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm generation. *arXiv preprint arXiv:2410.16179*, 2024.

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. Learning to compress prompt in natural language formats. *arXiv preprint arXiv:2402.18700*, 2024.

[9] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[11] Together Computer. Redpajama: an open dataset for training large language models, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

[12] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. 2023. URL `https://arxiv.org/abs/2307.08691`.

[13] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[14] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*, 2024.

[15] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[16] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

[17] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23 (120):1–39, 2022.

[18] Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. Moa: Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*, 2024.

[19] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[20] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

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

[22] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023.

[23] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[24] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[25] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[26] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[27] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023.

[28] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*, 2024.

[29] James M Joyce. Kullback-leibler divergence. In *International encyclopedia of statistical science*, pages 720–722. Springer, 2011.

[30] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[31] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[32] Heejun Lee, Geon Park, Youngwan Lee, Jina Kim, Wonyoung Jeong, Myeongjae Jeon, and Sung Ju Hwang. Hip attention: Sparse sub-quadratic attention with hierarchical attention pruning. *arXiv preprint arXiv:2406.09827*, 2024.

[33] Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. *arXiv preprint arXiv:2412.10319*, 2024.

[34] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[35] Liu Liu, Zheng Qu, Zhaodong Chen, Yufei Ding, and Yuan Xie. Transformer acceleration with dynamic sparse attention. *arXiv preprint arXiv:2110.11299*, 2021.

[36] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pages 22137–22176. PMLR, 2023.

[37] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[38] 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, Yuxin Wu, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[39] Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36, 2024.

[40] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[41] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=wHBfxhZu1u`.

[42] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

[43] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL `https://doi.org/10.1145/3394486.3406703`.

[44] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

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

[46] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[47] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[48] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

[49] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[50] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

[51] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[52] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE, 2021.

[53] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[54] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

[55] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[56] Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*, 2024.

[57] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

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

[59] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33: 17283–17297, 2020.

[60] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[61] Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *arXiv preprint arXiv:2405.03917*, 2024.

[62] Xuechen Zhang, Xiangyu Chang, Mingchen Li, Amit Roy-Chowdhury, Jiasi Chen, and Samet Oymak. Selective attention: Enhancing transformer through principled context control. *arXiv preprint arXiv:2411.12892*, 2024.

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

[64] Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, et al. Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *arXiv preprint arXiv:2406.15486*, 2024.

# A Appendix

## A.1 Training SeerAttention with Customized GPU Kernel

In this appendix, we provide a detailed design and implementation of our efficient kernel, highlighting key modifications to FlashAttention and optimizations for long-context scenarios. We then evaluate the peak memory usage and additional latency overhead of our training kernel during the AttnGate training stage, showing that it incurs only minimal overhead in both memory and latency compared to training with FlashAttention-2.

**FlashAttenion with 2D-MaxPooling: A Customized training kernel.** In Section 3.2, we discussed the method for obtaining the ground truth attention map used to distill AttnGate. Specifically, we leverage the 2D-MaxPooled attention map from full attention as the ground truth, aligning with the block-sparse attention definition. However, directly computing this attention map is challenging due to the quadratic memory complexity and the fused operation nature of FlashAttention. To overcome this, we developed a customized kernel based on Triton [49] that efficiently extracts the 2D-MaxPooled attention map by modifying the FlashAttention kernel while largely preserving its computation flow. Figure 8 shows the pseudo code and diagram of this customized kernel.

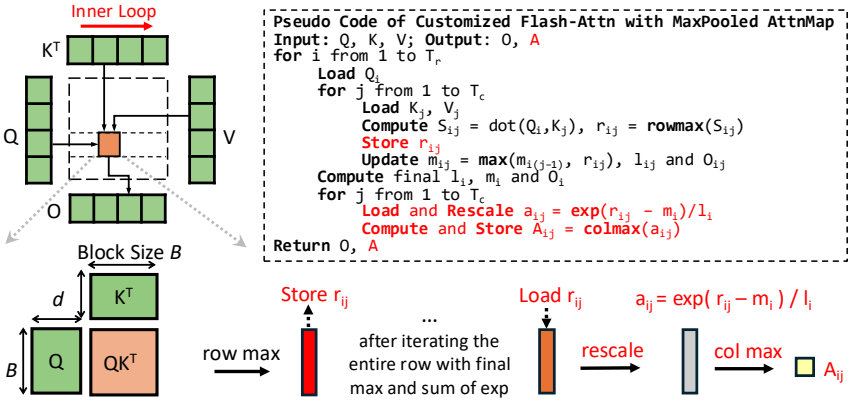



Figure 8: Efficient FlashAttention kernel with pooling of attention map.

Normally, the softmax function ensures numerical stability by subtracting the maximum value before applying the exponential operation. FlashAttention computes the local row max of each block, and gradually updates the global maximum through iteration:

$$\begin{aligned} S_{ij} &= Q_i K_j^T; \\ r_{ij} &= \text{rowmax}(S_{ij}); \\ m_{ij} &= \max(m_{i(j-1)}, r_{ij}). \end{aligned} \tag{7}$$

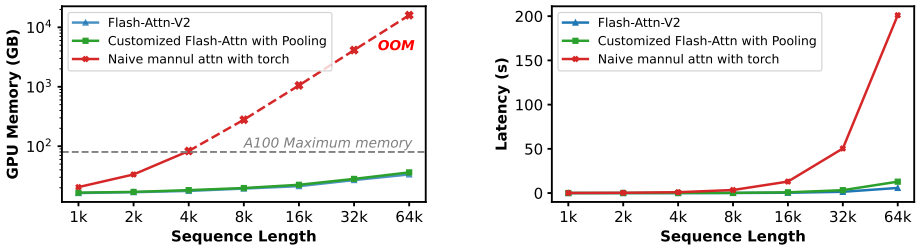
where $r_{ij}$ is typically treated as a temporary result. However, we store it in HBM and rescale it later with the final global max $m_i$ and sum of exp $l_i$ after the iteration:

$$a_{ij} = \exp(r_{ij} - m_i)/l_i \tag{8}$$

This $a_{ij}$ represents the correct row max of the original attention block. With that, 2D-MaxPooling is achieved by applying a column max over $a_{ij}$. This introduces only minor overhead (storing and rescaling $r_{ij}$) but significantly improves the efficiency of obtaining the ground truth. The overhead of memory and latnecy analysis is in Figure 9.

**Performance of the Training Kernel.** We evaluate our customized FlashAttention kernel with 2D-MaxPooled attention map for scalable training of SeerAttention by comparing against with PyTorch naïve manual attention implementation and FlashAttention-2. As shown in Figure 9b, the PyTorch kernel runs out of memory (OOM) when the sequence length exceeds 4k, while our customized kernel costs similar peak memory usage compared to FlashAttention-2. Regarding latency, since PyTorch encounters OOM for sequences longer than 8K, the attention operations per head into a loop to assess

kernel-level latency. Figure 9b shows that the latency overhead introduced by the additional pooling operation is minimal compared to FlashAttention-2, while the PyTorch implementation suffers from a significant slowdown.




(a) Memory (b) Latency

Figure 9: Memory and latency of customized FlashAttention with max-pooling training kernel.

### A.2 Preliminary Experiments of Fine-tuning with SeerAttention

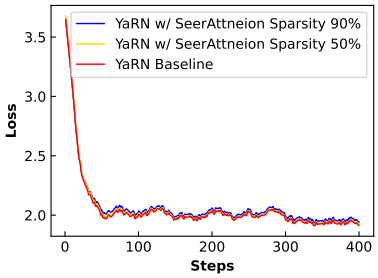



Figure 10: Fine-tuning Loss.

Figure 11: By incorporating SeerAttention with YaRN [41] to extend a Llama-3-8B model from 8k to 32k context length, the loss curves for 50% to 90% sparsity are nearly identical to the dense YaRN baseline.

Table 4: Perplexity of YaRN baseline, SeerAttention after YaRN and YaRN fine-tuning with SeerAttention.

| | YaRN | Post-training SeerAttention after YaRN | | | | | YaRN with SeerAttention | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Sparsity | 0.0 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| PG19 | 8.79 | 9.16 | 9.30 | 9.48 | 9.73 | 10.18 | 8.81 | 8.82 | 8.85 | 8.93 | 9.16 |
| Proof-pile | 2.46 | 2.53 | 2.57 | 2.61 | 2.68 | 2.85 | 2.47 | 2.47 | 2.48 | 2.51 | 2.60 |

In this preliminary experiment, we demonstrate that SeerAttention can be seamlessly integrated in Long-context extension fine-tuning stages. We follow YaRN [41] to extend the context size of a Llama-3-8B model from 8k to 32k. The loss function is the summation of original LM loss and AttnGate loss. To ensure stable training, the AttnGates are first initialized using the post-training self-distillation before fine-tuning the entire model. We integrate SeerAttention into YaRN and compare the performance against the YaRN dense baseline and the post-training time self-distillation of SeerAttention applied after YaRN. Figure 10 presents the loss curves of the YaRN dense baseline and SeerAttention at 50% and 90% sparsity. The curve at 50% sparsity nearly overlaps with the baseline, while the curve at 90% sparsity shows slightly higher loss. Table 4 displays the test perplexity on the PG19 and ProofPile datasets evaluated at a 32k context length. The YaRN dense baseline achieves perplexity scores of 8.79 and 2.46, respectively. Post-training SeerAttention results in increased perplexity. When applying SeerAttention during the YaRN extension fine-tuning, it maintains near-lossless performance at 50% sparsity (with scores of 8.81 and 2.47), and even at 90% sparsity, the loss remains minimal.

### A.3 Preliminary Results on Introducing Sparse Attention at Decoding Stage

**Adjusting AttnGate for decoding** The current AttnGate design mainly works for accelerating long-context prefill. However, applying the attention gate distillation idea is also a feasible direction. This is important to improve the efficiency of reasoning models that generate longer sequences during inference before producing an answer, aka test-time scaling. Adjusting current design to compatible for decoding cases requires removing the sequence level pooling of Query to adhere to the token-by-token generation fashion. Here is a modification of AttnGate design in (9):

$$\mathbf{Q}_c = \mathrm{RoPE}\Big(\mathbf{W}^{\mathbf{q}}_{\mathbf{gate}}\ \mathrm{reshape}(\mathbf{Q}_{nope}, [..., g \cdot d])\Big), \tag{9a}$$

$$\mathbf{K}_c = \mathrm{RoPE}\Big(\mathbf{W}^{\mathbf{k}}_{\mathbf{gate}}\ \mathrm{concat}[\mathrm{P}_{\max}(\mathbf{K}_{nope}), \mathrm{P}_{\min}(\mathbf{K}_{nope}), \mathrm{P}_{\mathrm{avg}}(\mathbf{K}_{nope})]\Big), \tag{9b}$$

$$\mathbf{S} = \mathrm{softmax}(\mathbf{Q}_c\,\mathbf{K}_c^{\top}/\sqrt{d_{gate}}). \tag{9c}$$

where, $\mathrm{P}_{\max}$, $\mathrm{P}_{\min}$, and $\mathrm{P}_{\mathrm{avg}}$ stand for Max, Min and Average Pooling in sequence dimension, and $g$ is the group size of GQA setting. $d$ and $d_{gate}$ are the hidden dimension of the original model and AttnGate for each head, respectively. $\mathbf{S}$ is the output score of each block from AttnGate. We aggregate Query heads within each group to share sparsity decisions to improve the decoding efficiency. Specifically, a linear layer in the $\mathbf{Q}$ branch reduces each subgroup of queries (e.g., 32 heads $\rightarrow$ 8 heads for $g$=4) to a single $\mathbf{Q}_c$ head while keeping $\mathbf{K}$ heads unchanged, enabling shared sparsity among grouped queries. To compress $\mathbf{K}$ along the sequence dimension, we apply non-overlapping block-level pooling that concatenates Max, Min, and Average pooling outputs before a linear projection. Additionally, AttnGate reapplies RoPE on pre-RoPE $\mathbf{Q}$ and $\mathbf{K}$, assigning each block the position of its first token, which we find improves the accuracy.

**Evaluation on Reasoning Tasks** We evaluate this design on three reasoning benchmarks: AIME24, MATH-500 [23], and GPQA-Diamond [44] using DeepSeek-R1-Distill-Qwen-14B [21]. We compare against full attention and Quest [48], which is a training-free sparse decoding method using query-aware KV cache selection. Both methods use a block size of 64 and apply sparsity to all layers for fair comparison. The maximum output length is fixed at 32,768 tokens across all settings. Accuracy is reported as average pass@1 over 64 (AIME24), 8 (MATH-500), and 16 (GPQA) samples. For AttnGate distillation, we use OpenR1-MATH-220k [16] dataset with 800 steps and global batch size 16 on AMD MI300x GPUs, employing DeepSpeed ZeRO-2, AdamW optimizer, and a 1e-3 learning rate with cosine decay.

Table 5: Performance comparison across different token budgets for SeerAttention-decoding and Quest using DeepSeek-R1-Distill-Qwen-14B model.

| **Method** | **Dataset** | **Token Budgets** | | | | **Full** |
|---|---|---|---|---|---|---|
| | | **2k** | **4k** | **6k** | **8k** | |
| SeerAttention-decoding | AIME24 | 55.78 | 66.35 | 67.50 | 66.82 | 67.50 |
| | MATH500 | 87.65 | 92.10 | 93.05 | 93.12 | 93.30 |
| | GPQA-Diamond | 51.26 | 56.79 | 56.41 | 57.48 | 57.80 |
| Quest | AIME24 | 25.83 | 46.67 | 53.75 | 60.00 | 67.50 |
| | MATH500 | 57.40 | 79.60 | 88.60 | 92.20 | 93.30 |
| | GPQA-Diamond | 35.35 | 50.25 | 52.53 | 54.55 | 57.80 |

Table Table 5 shows the superior performance of our design compared to Quest across multiple reasoning benchmarks and token budgets. On AIME24, we consistently outperforms Quest, achieving 55.78 vs. 25.83 accuracy at 2k tokens and maintaining a strong lead using more token budgets. Similarly, on MATH500 and GPQA-Diamond, SeerAttention-decoding shows better budget-accuracy tradeoff compared with Quest, reflecting better information retention under sparse decodin owing to its learned gate that shares sparsity decisions, enabling more coherent block selection.

## 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: In abstract and introduction, we provide a clear description of motivation, methods and contribution of the paper.

   Guidelines:

   - The answer NA 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 NA 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 provide discussion of limitations in the last section.

   Guidelines:

   - The answer NA 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: [NA]

Justification: All the results are real experiment results without theoretical assumption.

Guidelines:

- The answer NA 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: Training and testing details are listed in detail. The training data and testing benchmark are all public-available resources.

   Guidelines:

   - The answer NA 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: [No]

Justification: We did not provide code in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have listed detailed hyperparameters of the training setting.

   Guidelines:

   - The answer NA 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: The experiments results are mostly deterministic without an error bar.

   Guidelines:

   - The answer NA 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 provide training GPU settings and typical GPU hours information.

   Guidelines:

   - The answer NA 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: Our research does not involve human subjects, and all the training/testing dataset and models are commonly used open-source resources.

   Guidelines:

   - The answer NA 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: [NA]

    Justification: This work is mainly about model efficiency. We believe the societal impact is mostly related to the model functionality itself instead of how fast it can run.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: We use the common open-sourced models in our experiments.

    Guidelines:

    - The answer NA 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: This paper properly credits and cites the creators or original owners of the assets used including code, data, and models.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: We did not provide code in submissions.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: This research does not involve LLMs as any important, original, or non-standard components

Guidelines:

- The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.