SPARGEATTN: TRAINING-FREE SPARSE ATTENTION ACCELERATING ANY MODEL INFERENCE

Jintao Zhang[1]*, Chendong Xiang[1]*, Haofeng Huang[2]*, Jia Wei[1], Haocheng Xi[3], Jun Zhu[1], Jianfei Chen[1]†
[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center,
Tsinghua-Bosch Joint ML Center, THBI Lab, Tsinghua University
[2]Institute for Interdisciplinary Information Sciences, Tsinghua University
[3]EECS, University of California, Berkeley
{zhang-jt24@mails., dcszj@, jianfeic@}tsinghua.edu.cn

ABSTRACT

An efficient attention implementation is essential for large models due to its quadratic time complexity. Fortunately, attention commonly exhibits sparsity, i.e., many values in the attention map are near zero, allowing for the omission of corresponding computations. Many studies have utilized the sparse pattern to accelerate attention. However, most existing works focus on optimizing attention within specific models by exploiting certain sparse patterns of the attention map. **A universal sparse attention that guarantees both the speedup and end-to-end performance of diverse models remains elusive.** In this paper, we propose SpargeAttn, a universal sparse and quantized attention for any model. Our method uses a two-stage online filter: in the first stage, we rapidly and accurately predict the attention map, enabling the skip of some matrix multiplications in attention. In the second stage, we design an online softmax-aware filter that incurs no extra overhead and further skips some matrix multiplications. Experiments show that our method significantly accelerates diverse models, including language, image, and video generation, without sacrificing end-to-end metrics.

1 INTRODUCTION

Limitation of Existing Work. (L1. Universality) Though existing sparse attention methods Zhang et al. (2023); Xiao et al. (2024a); Fu et al. (2024); Zhu et al. (2024); Xiao et al. (2025; 2024b); Ribar et al. (2024); Singhania et al. (2024); Jiang et al. (2024); FlexPrefill (2025); Gao et al. (2024); Kitaev et al. (2020); Pagliardini et al. (2023) already demonstrate promising results, their universality remains limited. They are typically developed for specific tasks, like language modeling, using patterns such as sliding windows or attention sink. However, the attention pattern varies significantly across tasks (see examples in Fig. 5), making these methods hard to generalize. (L2. Usability) Moreover, it is difficult to implement both *accurate* and *efficient* sparse attention for any input. This is because *accuracy* demands precise prediction of the sparse regions in the attention map, while *efficiency* requires the overhead of this prediction to be minimal. However, current methods are difficult to effectively satisfy both of the requirements simultaneously.

Goal: We aim to design a training-free sparse attention operator that accelerates all models without metrics loss. **Our approach:** We develop SpargeAttn, a *training-free* sparse attention that can be adopted *universally* on various tasks, including language modeling and text-to-image/video. We propose three main techniques to improve the universality, accuracy, and efficiency. First, we propose a universal sparse mask prediction algorithm, which constructs the sparse mask by compressing each block of Q, K to a single token. Importantly, we compress *selectively* based on the *similarity* of tokens within the block, so the algorithm can accurately predict sparse masks universally across tasks. Second, we propose a sparse online softmax algorithm at the GPU warp level, which further omits some PV products by leveraging the difference between global maximum values and local maximum values in online softmax. Third, we integrate this sparse approach into the 8-bit quantized SageAttention framework for further acceleration.

*Equal contribution.
†Corresponding author.

Result. We evaluate `SpargeAttn` on a variety of generative tasks, including language modeling and text-to-image/video, with comprehensive performance metrics on the model quality. `SpargeAttn` can robustly retain model end-to-end performance while existing sparse attention baselines incur degradation. Moreover, `SpargeAttn` is 2.5x to 5x faster than existing dense and sparse attention models.

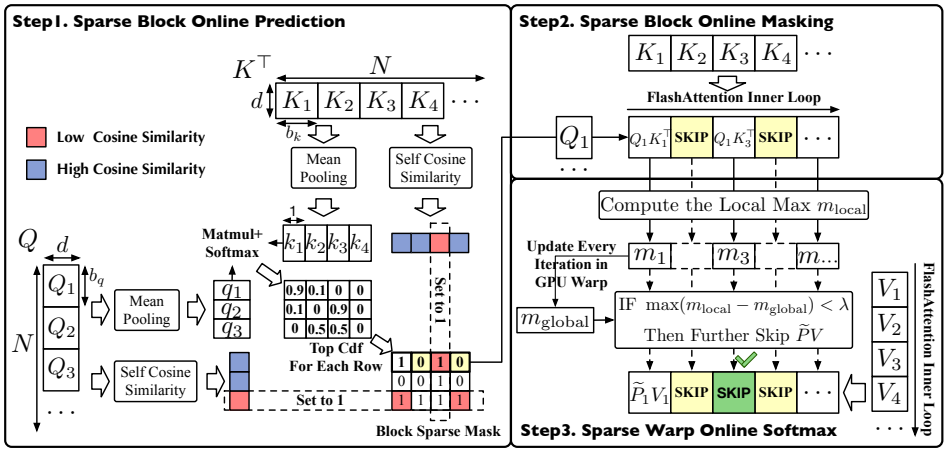

Figure 1: Workflow of `SpargeAttn`.

2 SPARGEATTN

2.1 SPARSE FLASHATTENTION

`SpargeAttn` adopts the tiling strategy of FlashAttention Dao (2024), and skip computing the blocks that are filtered out. Consider an attention operation $S = QK^\top/\sqrt{d},\ P = \sigma(S),\ O = PV$, where $\sigma(S)_{ij} = \exp(S_{ij})/\sum_k \exp(S_{ik})$ is the softmax operation. Let N be the sequence length and d be the dimensionality of each head; the matrices Q, K, and V each have dimensions $N \times d$, while the matrix S and P is $N \times N$. FlashAttention proposes to tile Q, K, and V from the token dimension into blocks $\{Q_i\}, \{K_i\}, \{V_i\}$ with block sizes b_q, b_k, b_k, respectively. Then, it uses online softmax Milakov & Gimelshein (2018) to progressively compute each block of O, i.e., O_i:

$$S_{ij} = Q_i K_j^\top/\sqrt{d},\ \ (m_{ij}, \widetilde{P}_{ij}) = \tilde{\sigma}(m_{i,j-1}, S_{ij}),\ \ l_{ij} = \exp(m_{i,j-1} - m_{ij}) l_{i,j-1} + \text{rowsum}(\widetilde{P}_{ij})$$
$$O_{ij} = \text{diag}\left(\exp(m_{i,j-1} - m_{ij}) O_{i,j-1} + \widetilde{P}_{ij} V_j \right. \tag{1}$$

where m_{ij} and l_{ij} are $b_q \times 1$ vectors, which are initialized to $-\infty$ and 0 respectively. The $\tilde{\sigma}()$ is an operator similar to softmax.: $m_{ij} = \max\{m_{i,j-1}, \text{rowmax}(S_{ij})\},\ \widetilde{P}_{i,j} = \exp(S_{ij} - m_{ij})$. Finally, the output O_i can be computed by $O_i = \text{diag}(l_{ij})^{-1} O_{ij}$. Implementing sparse FlashAttention is intuitive. By *skipping* certain block matrix multiplications of $Q_i K_j^\top$ and $\widetilde{P}_{ij} V_j$, we can accelerate the attention computation. We formulate sparse attention in the following definitions.

Definition 1 (Block Masks)**.** Let M_g and M_{pv} be binary masks of dimensions $\lceil N/b_q \rceil \times \lceil N/b_k \rceil$, where each value is either 0 or 1. These masks determine which computations are skipped in the sparse attention mechanism.

Definition 2 (Sparse FlashAttention)**.** The computation rules for sparse FlashAttention based on the masks are defined as follows:

$$Q_i K_j^\top, \widetilde{P}_{ij} V_j \ \text{ are skipped if } \ M_g[i,j] = 0. \qquad \widetilde{P}_{ij} V_j \ \text{ is skipped if } \ M_{pv}[i,j] = 0. \tag{2}$$

2.2 SELECTIVE TOKEN COMPRESSION FOR SPARSE PREDICTION

Key idea. Although attention maps vary across models, we observe that various models exhibit a common trait: Most closer tokens in the query and key matrices of the attention show high similarity (See Fig. 6). Consequently, we first compress blocks exhibiting high self-similarity within Q and K into tokens. Then, we swiftly compute a compressed attention map $\hat{P}$ using the compressed Q and K. Finally, we selectively compute $\{Q_i K_j^\top, \widetilde{P}_{ij} V_j\}$ for those pairs (i, j) where $\{\hat{P}[i, j]\}$ accumulates a

high score in $\hat{P}$. Importantly, compressing only the token blocks with high self-similarity is crucial, as omitting computations for non-self-similar blocks can result in the loss of critical information (ablated in Appendix A.4).

Prediction. As shown in `Step1` in Fig. 1, we first compute a mean cosine similarity across tokens for each block of Q and K. Next, we compress each block into a single token by calculating a mean across tokens. Then, we compute a compressed $QK^\top$ using the compressed Q and K. Finally, to prevent interference from non-self-similar blocks, i.e., the block similarity less than a hyper-parameter θ, we set the corresponding values in S to $-\infty$, and then compute a softmax:

$$q = \{q_i\} = \{\text{mean}(Q_i, \text{axis} = 0)\},\ k = \{k_j\} = \{\text{mean}(K_j, \text{axis} = 0)\},\ s_{qi} = \text{CosSim}(Q_i)$$
$$s_{kj} = \text{CosSim}(K_j),\ \hat{S}[i] = q_i k^\top;\ \ \hat{S}[:, j] = -\infty,\ \text{If } s_{kj} < \theta,\ \hat{P}[i] = \text{Softmax}(\hat{S}[i])$$

where $Q_i \in \mathbb{R}^{b_q \times d}, q_i \in \mathbb{R}^{1\times d}, K_j \in \mathbb{R}^{b_k \times d}, k_j \in \mathbb{R}^{1\times d}$ and $\text{CosSim}(X) = \frac{XX^\top}{|\max(XX^\top)|}$.

For each row of $\hat{P}$, i.e., $\hat{P}[i, :]$, we select $M_g[i, :]$ as the positions of the top values whose cumulative sum reaches τ, where τ is a hyper-parameter. Finally, we need to ensure calculations involving non-self-similar blocks of Q or K are not omitted:

$$M_g[i, :] = 1,\ \text{If } s_{qi} < \theta; \qquad M_g[:, j] = 1,\ \text{If } s_{kj} < \theta \tag{3}$$

2.3 Sparse Warp Online Softmax

Key idea. We can further identify the small enough values in the attention map during the online softmax process. If all values in $\widetilde{P}_{ij}$ are close enough to zero, the $\widetilde{P}_{ij}V_j$ can be omitted.

To identify which $\widetilde{P}_{ij} = \exp(S_{ij} - m_{i,j})$ (See Sec. 2.1) contains values small enough to be omitted, we note that in every inner loop of FlashAttention, the O_{ij} will be scaled by $\exp(m_{i,j-1} - m_{ij})$ and then plus the $\widetilde{P}_{ij}V_j$:

$$m_{\text{local}} = \text{rowmax}(S_{ij}),\ \ m_{ij} = \max\{m_{i,j-1}, m_{\text{local}}\}$$
$$O_{ij} = \text{diag}\left(\exp(m_{i,j-1} - m_{ij})\right) O_{i,j-1} + \widetilde{P}_{ij}V_j$$

If $\text{rowmax}(S_{ij}) < m_{ij}$, then $m_{ij} = m_{i,j-1}$. Consequently, $O_{ij} = O_{i,j-1} + \widetilde{P}_{ij}V_j$. Furthermore, if $\text{rowmax}(S_{ij}) \ll m_{ij}$ holds ture, then all values in $\widetilde{P}_{ij} = \exp(S_{ij} - m_{ij})$ are close to 0. This results in all values in $\widetilde{P}_{ij}V_j$ being close to 0. This condition implies that $\widetilde{P}_{ij}V_j$ is negligible when $\text{rowmax}(S_{ij})$ is significantly smaller than m_{ij}:

$$O_{ij} \approx O_{i,j-1}, \quad \text{if } \max\left(\exp(S_{ij} - m_{ij})\right) \to 0$$
$$\max\left(\exp(S_{ij} - m_{ij})\right) \to 0 \Leftrightarrow \max(m_{\text{local}} - m_{ij}) < \lambda$$

2.4 Combined with SageAttention

Since quantization operations and sparse operations are orthogonal, sparse computation can be directly applied to SageAttention Zhang et al. (2025). Specifically, first, we need to add one judgment at the beginning of the inner loop of SageAttention to decide whether to skip the whole inner loop once. Second, we add another judgment before the updating of O_{ij} in the inner loop of SageAttention to decide whether to skip the computation of $\widetilde{P}_{ij}V_j$.

3 Experiment

3.1 Setup

Models, Datasets, metrics. For details about the models, datasets, and metrics, please refer to Appendix. A.2.

Speed and sparsity metric. We use TOPS (tera operations per second) to evaluate the speed of sparse attention methods. Specifically, **TOPS** = $O(attn)/t$, where $O(attn)$ represents the total number of operations in a standard attention computation, and t is the latency of attention operation,

Table 1: End-to-end metrics across text, image, and video generation models. The speed and sparsity are the average for each layer in the model in real generation tasks described in Sec. 3.1. The speed and sparsity of `Llama3.1` are measured in the Needle-in-a-Haystack task with a 128K sequence length.

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**WikiText (Ppl.)** ↓	**Longbench** ↑	**InfiniteBench** ↑	**NIAH** ↑
`Llama3.1` (128K)	Full-Attention	156.9	6.013	38.682	0.6594	0.907
	Minference (0.5)	140.1	10.631	28.860	0.5152	0.832
	FlexPrefill (0.5)	240.6	6.476	38.334	0.6460	0.858
	Minference (0.3)	115.7	6.705	34.074	0.6532	0.870
	FlexPrefill (0.42)	206.9	6.067	38.334	0.6581	0.878
	`SpargeAttn` (0.54)	**708.1**	**6.020**	**39.058**	**0.6638**	**0.909**

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**CLIPSIM** ↑	**CLIP-T** ↑	**VQA-a** ↑	**VQA-t** ↑	**FScore** ↑
`CogvideoX` (17K)	Full-Attention	166.0	0.1819	0.9976	80.384	75.946	5.342
	Minference (0.5)	264.6	0.1728	0.9959	70.486	62.410	2.808
	FlexPrefill (0.6)	175.3	0.1523	0.9926	1.5171	4.5034	1.652
	Minference (0.3)	196.9	0.1754	0.9964	77.326	63.525	3.742
	FlexPrefill (0.45)	142.0	0.1564	0.9917	7.7259	8.8426	2.089
	`SpargeAttn` (0.46)	**507.9**	**0.1798**	**0.9974**	**78.276**	**74.846**	**5.030**

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**FID** ↓	**CLIP** ↑	**IR** ↑
`Flux` (4.5K)	Full-Attention	158.2	166.103	31.217	0.8701
	Minference (0.5)	151.8	180.650	30.235	0.4084
	FlexPrefill (0.48)	47.7	443.928	18.3377	-2.2657
	Minference (0.3)	118.9	170.221	31.001	0.7701
	FlexPrefill (0.41)	40.9	405.043	19.5591	-2.2362
	`SpargeAttn` (0.38)	**280.3**	**163.982**	**31.448**	**0.9207**

including the time spent predicting the sparse region of the attention map. We define **Sparsity** as the proportion of the Matmul of Q_iK_j plus $P_i^jV_j$ that are skipped relative to the total number of Q_iK_j plus $P_i^jV_j$ in a full attention.

Baselines and Hyperparameters. Currently, sparse attention methods applicable across different model types are limited. We choose block-sparse MInference Jiang et al. (2024) and FlexPrefill Flex-Prefill (2025) as our baselines. To vary the *sparsity* of these baselines, we use 30% and 70% for MInference, and use $\gamma = 0.95$ and 0.99 for FlexPrefill according to their paper. Hyperparameter determination for `SpargeAttn` are detailed in Appendix. A.3.

3.2 Quality and Efficiency Evaluation

End-to-end metrics. We assess the end-to-end metrics of various models using `SpargeAttn` compared to using full attention and baselines. As shown in Table 1, our method incurs almost no end-to-end metric loss across various models compared to Full-Attention and surpasses baselines with various sparsity levels in terms of end-to-end accuracy. Fig. 3 and 4 show some visible comparison examples on different image/video generation models, showing that `SpargeAttn` incurs no performance loss and outperforms baselines. More results are shown in Appenedix A.5.

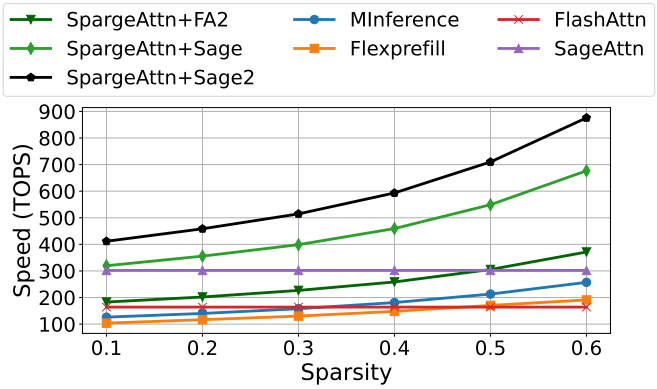

Figure 2: Kernel speed comparison under varying sparsity on RTX4090. Input tensors have a sequence length of 22K and a head dimension of 128. *SpargeAttn+FA2/Sage/Sage2* means deploying our method on FlashAttention2, SageAttention or SageAttention2 Zhang et al. (2024a).

Attention speed. Table 1 shows that our method achieves faster speeds compared to Full-Attention and surpasses baselines with various sparsity levels in terms of attention speed. Fig. 2 illustrates the kernel speeds of various methods across different sparsity. Tabel 2 demonstrates that the overhead of dynamic sparse block prediction of `SpargeAttn` is minimal, particularly for long sequences.

End-to-end speedup. Table 3 shows the end-to-end latency on `CogvideoX`, `Mochi`, and `Llama3.1` using `SpargeAttn`. Notably, `SpargeAttn` achieves 1.83x speedup on `Mochi`.

Table 2: Overhead (ms) of sparse block prediction in `SpargeAttn`.

Seq Len	Prediction	Full Attention	Overhead
8k	**0.251**	6.649	3.78%
16k	**0.487**	26.83	1.82%
32k	**0.972**	106.68	0.911%
128k	**8.764**	1696.2	0.516%

Table 3: End-to-end generation latency using `SpargeAttn`.

Model	GPU	Original	`SageAttn`	`SpargeAttn`
`CogvideoX`	RTX4090	87 s	68 s	**53 s**
`Mochi`	L40	1897 s	1544 s	**1037 s**
`Llama3.1` (24K)	RTX4090	4.01 s	3.53 s	**2.6 s**
`Llama3.1` (128K)	L40	52 s	42s	**29.98 s**

REFERENCES

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

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2023.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

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.

FlexPrefill. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations*, 2025.

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.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7514–7528, 2021.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1): S63–S63, 1977.

Huiqiang Jiang, YUCHENG LI, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Gregory Kamradt. Llmtest needle in a haystack-pressure testing llms. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`, 2023.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. Evalcrafter: Benchmarking and evaluating large video generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22139–22149, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Maxim Milakov and Natalia Gimelshein. Online normalizer calculation for softmax. *arXiv preprint arXiv:1805.02867*, 2018.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Fast attention over long sequences with dynamic sparse flash attention. *Advances in Neural Information Processing Systems*, 36:59808–59831, 2023.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient LLM inference. In *Forty-first International Conference on Machine Learning*, 2024.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Stability AI. Introducing stable diffusion 3.5. `https://stability.ai/news/introducing-stable-diffusion-3-5`, 2023.

Genmo Team. Mochi 1. `https://github.com/genmoai/models`, 2024.

Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring video quality assessment on user generated contents from aesthetic and technical perspectives. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20144–20154, 2023.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

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

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. In *The International Conference on Learning Representations*, 2025.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36, 2024.

Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. Sageattention2: Efficient attention with thorough outlier smoothing and per-thread int4 quantization, 2024a. URL `https://arxiv.org/abs/2411.10958`.

Jintao Zhang, Jia Wei, Pengle Zhang, Jianfei Chen, and Jun Zhu. Sageattention: Accurate 8-bit attention for plug-and-play inference acceleration. In *International Conference on Learning Representations*, 2025.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞Bench: Extending long context evaluation beyond 100K tokens. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024b.

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.

Tianchen Zhao, Tongcheng Fang, Haofeng Huang, Enshu Liu, Rui Wan, Widyadewi Soedarmadji, Shiyao Li, Zinan Lin, Guohao Dai, Shengen Yan, Huazhong Yang, et al. Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. In *International Conference on Learning Representations*, 2025.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

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

A Appendix

A.1 Visual Examples

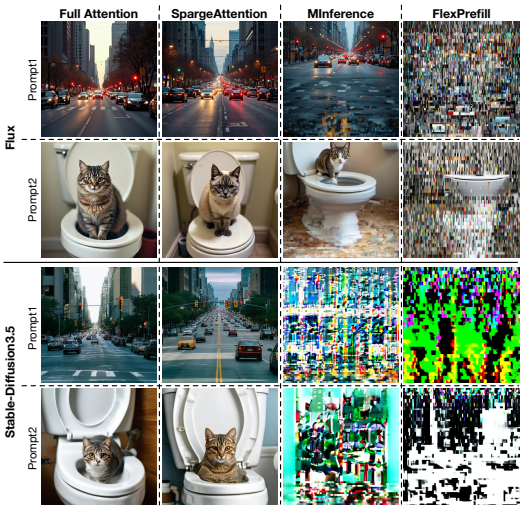

Figure 3: Comparison examples on `Flux` and `Stable-Diffusion3.5`. The sparsity of `SpargeAttn`, MInference and FlexPrefill is 0.38, 0.3, and 0.4 on `Flux` and 0.31, 0.3, and 0.35 on `Stable-Diffusion3.5`.

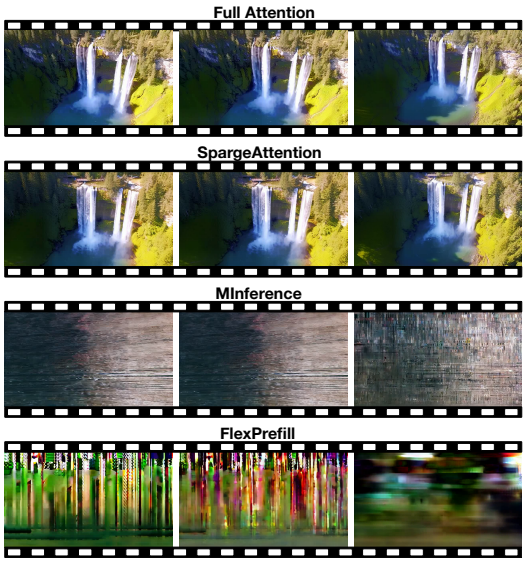

Figure 4: Comparison examples on `Mochi`. The sparsity of `SpargeAttn`, MInference and FlexPrefill is 0.47, 0.3, and 0.4.

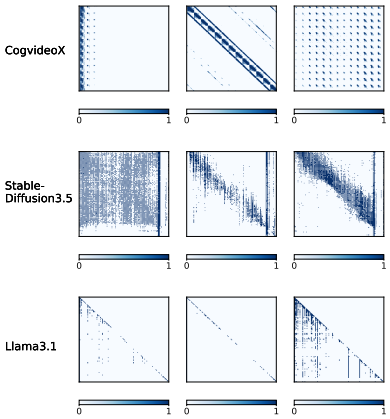

Figure 5: Some sampled patterns of attention map P in various models.

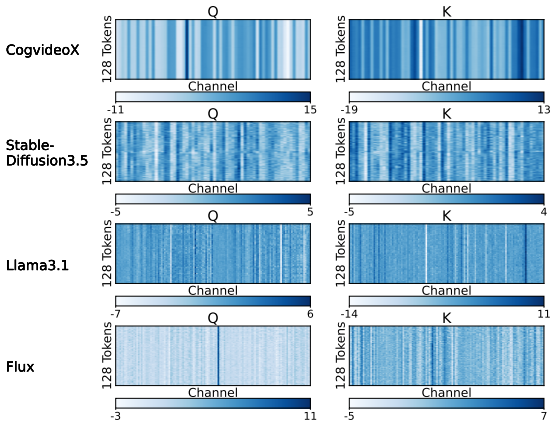

Figure 6: Exemplary patterns of the query and key in the attention of various models.

A.2 Models, Datasets, and Metrics

Models. We validate the effectiveness of `SpargeAttn` across diverse representative models from language, image, and video generation. Specifically, we conduct experiments on `Llama3.1` (8B) Dubey et al. (2024) for text-to-text, `CogvideoX` (2B) and `Mochi` Team (2024) for text-to-video, `Flux` Black Forest Labs (2023)(.1-dev) and `Stable-Diffusion3.5` (large) Stability AI (2023) for text-to-image.

Datasets. The Text-to-text model is evaluated on four zero-shot tasks: WikiText Merity et al. (2017) to assess the model's prediction confidence, Longbench Bai et al. (2024) and En.MC of InfiniteBench Zhang et al. (2024b) for a comprehensive assessment of long context understanding capabilities, and the Needle-in-a-Haystack task Kamradt (2023) to assess the model's retrieval ability.

Text-to-video models are evaluated using the open-sora Zheng et al. (2024) prompt sets. Text-to-image models are assessed on COCO annotations Lin et al. (2014).

End-to-end metrics. For `Llama3.1`, we use perplexity (ppl.) Jelinek et al. (1977) for WikiText, Longbench score Bai et al. (2024), and retrival accuracy for the Needle-in-A-Haystack task Kamradt (2023). For text-to-video models, following Zhao et al. (2025), we evaluate the quality of generated videos on five metrics: CLIPSIM and CLIP-Temp (CLIP-T) Liu et al. (2024) to measure the text-video alignment; VQA-a and VQA-t to assess the video aesthetic and technical quality, and Flow-score (FScore) for temporal consistency Wu et al. (2023). For text-to-image models, generated images are compared with the images in the COCO dataset in three aspects: FID Heusel et al. (2017) for fidelity evaluation, *Clipscore* (CLIP) Hessel et al. (2021) for text-image alignment, and *ImageReward* (IR) Xu et al. (2024) for human preference.

A.3 Hyper-parameters Determination for Model Layer

Based on the method description in Sec. 2.2 and 2.3, our method incorporates three hyper-parameters: $\tau \in (0, 1)$, $\theta \in (-1, 1)$, and $\lambda < 0$. The parameter determination process for each attention layer in any model is straightforward. We aim to identify a set of hyperparameters that not only maximize attention sparsity but also constrain the attention error across five different model inputs. To evaluate attention accuracy, we employ a strict error metric, the Relative L1 distance, defined as $L1 = \sum |O - O'| / \sum |O|$. The process begins by setting two L1 error thresholds l_1 and l_2, e.g., $l_1 = 0.05, l_2 = 0.06$. We first conduct a grid search for τ and θ to identify the optimal pair that maximizes sparsity while ensuring $L1 < l_1$. Subsequently, we perform another grid search for λ to find the optimal value that further maximizes sparsity while maintaining $L1 < l_2$.

In our experiments, we use ($l_1 = 0.08, l_2 = 0.09$) for `Llama3.1`, ($l_1 = 0.05, l_2 = 0.06$) for `CogvideoX` and `Mochi`, and ($l_1 = 0.07, l_2 = 0.08$) for `Stable-Diffusion3.5` and `Flux`.

A.4 Ablation Study of Self-Similarity Judge

To investigate the impact of the self-similarity judge on attention performance, we conduct an ablation study by removing the self-similarity judge, using five distinct prompts and pre-searched hyperparameters with $l_1 = 0.05, l_2 = 0.06$ on both `CogvideoX` and `Mochi` models. In most cases, the presence of highly localized patterns results in a minimal number of non-self-similar blocks, leading to only minor differences in precision and sparsity when averaging across all tensor cases. To obtain more meaningful and interpretable insights, we specifically analyze cases where the precision difference is statistically significant.

To this end, we apply a threshold-based selection criterion, retaining only those cases where the absolute difference between $L1^{sim-judge}$ (precision error with the self-similarity judge) and $L1^{no-judge}$ (precision error without the self-similarity judge) exceeds 0.05. This criterion results in approximately 2% of the tensor cases being retained for further analysis. We employ precision (L1 error) and sparsity as evaluation metrics to assess the influence of the self-similarity judge on the attention output. The results are summarized in Table 4.

Table 4: Impact of the self-similarity judge on the accuracy and sparsity of attention.

Method	**w/ judge**		**w/o judge**		**filter w/ judge**		**filter w/o judge**	
	`CogvideoX`	`Mochi`	`CogvideoX`	`Mochi`	`CogvideoX`	`Mochi`	`CogvideoX`	`Mochi`
L1 error↓	0.0316	0.0343	0.0325	0.0365	0.0843	0.0555	0.214	0.154
Sparsity ↑	0.199	0.301	0.203	0.305	0.242	0.371	0.275	0.392

The findings demonstrate that the self-similarity judge effectively mitigates extreme precision loss while introducing only a marginal reduction in sparsity. We also evaluate the end-to-end accuracy result in Table 5.

Table 5: Abalation of self-similarity judge.

Method	**VQA-a** ↑	**VQA-t** ↑	**FScore** ↑
W/o. self-sim Judge	34.664	44.722	1.138
With self-sim Judge	54.179	67.219	1.807

A.5 Additional Experiments

Table 6 shows results on `Mochi` and `Stable-Diffusion3.5`. Figure 7 shows the visual results on Needle-in-a-Haystack of `Llama3.1`.

Table 6: Results on `Mochi` and `Stable-Diffusion3.5`. ✗ indicates an inability to generate results for evaluation.

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**CLIPSIM** ↑	**CLIP-T** ↑	**VQA-a** ↑	**VQA-t** ↑	**FScore** ↑
`Mochi` (22K)	Full-Attention	164.2	0.1725	0.9990	56.472	67.663	1.681
	Minference (0.5)	202.4	0.1629	0.9891	6.668	50.839	0.653
	FlexPrefill (0.48)	191.3	0.1667	0.9898	0.582	0.0043	✗
	Minference (0.3)	147.7	0.1682	0.9889	14.541	42.956	0.833
	FlexPrefill (0.4)	171.7	0.1677	0.9909	2.941	0.7413	✗
	`SpargeAttn` (0.47)	**582.4**	**0.1720**	**0.9990**	**54.179**	**67.219**	**1.807**

Model (seq_len)	**Attention** (Sparsity)	**Speed (TOPS)**↑	**FID** ↓	**CLIP** ↑	**IR** ↑
`Stable-Diffusion3.5` (4.5K)	Full-Attention	164.2	166.101	32.007	0.9699
	Minference (0.5)	186.4	348.930	18.3024	-2.2678
	FlexPrefill (0.37)	23.1	350.497	18.447	-2.2774
	Minference (0.3)	150.3	337.530	18.099	-2.2647
	FlexPrefill (0.35)	22.7	348.612	18.147	-2.2756
	`SpargeAttn` (0.31)	**293.0**	**166.193**	**32.114**	**0.9727**

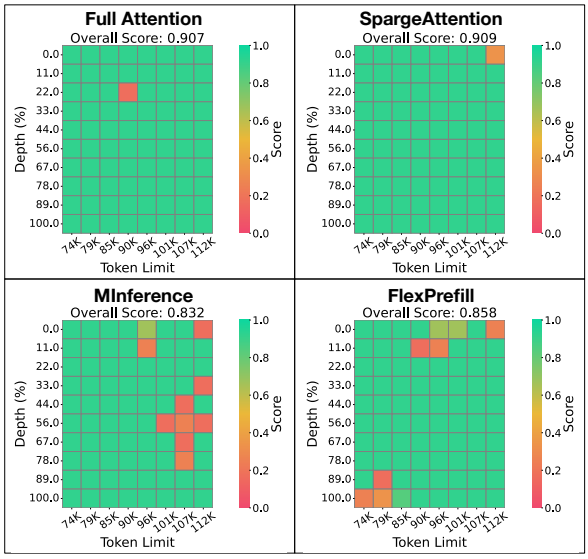

Figure 7: A Needle-in-a-Haystack comparison example on `Llama3.1`. The sparsity of `SpargeAttn`, MInference, and FlexPrefill is 0.5, 0.5, and 0.54.