TOSA: TRAINABLE ONLINE SPARSE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Processing long sequences is expensive because dense self-attention has quadratic complexity. Existing sparse attention methods, e.g., Native Sparse Attention, can skip score computation, but prediction and indexing introduce additional overhead. Online block skipping avoids a separate selection stage during inference, but high sparsity often reduces model accuracy. To address this issue, we propose **ToSA, a trainable online sparse attention method for long-sequence modeling**. ToSA reuses online-softmax statistics in a fused Triton kernel to identify low-contribution KV blocks after the attention scores are computed. For each skipped block, it bypasses value loading, softmax computation, and the subsequent PV accumulation. Its sparse backward kernel replays the block decisions recorded in the forward pass, enabling end-to-end training. We further propose a three-stage schedule that begins with dense training, gradually increases sparsity, and ends with training at the desired sparsity. We evaluate ToSA with Qwen3.5-2B on long-context and long-video understanding. Compared with the dense fine-tuning baseline, ToSA improves the LongBench score **from 38.18 to 39.62** and the LongVideoBench score **from 58.19 to 60.02**. In terms of efficiency, ToSA achieves up to a $\mathbf{1.77\times}$ **attention-kernel speedup at a 128K sequence length**. Together, these results demonstrate that ToSA substantially accelerates long-sequence modeling while preserving long-sequence understanding.

1 INTRODUCTION

Attention is a major computational bottleneck for large language models and multimodal large models with long input sequences. Dense self-attention evaluates every query–key pair, and even IO-aware kernels such as FlashAttention-2 (FA2) (Dao, 2024) still compute all attention interactions, so the prefill cost grows quadratically with sequence length. Existing sparse attention methods such as NSA (Yuan et al., 2025), MoBA (Lu et al., 2026), SeerAttention (Gao et al., 2026), and DSA (Liu et al., 2025) use different mechanisms to select important tokens or attention blocks, so that only the corresponding attention interactions need to be computed. However, these selection mechanisms require additional prediction, routing, or indexing, which adds overhead.

Recent studies explore online block-skipping methods that identify and skip low-contribution blocks during attention computation. SpargeAttention (Zhang et al., 2025) applies an online softmax-aware filter to skip additional PV computations, but this online filtering relies on a compressed attention map computed in an earlier stage. BLASST (Yuan et al., 2026) instead reuses statistics from online softmax to identify negligible attention blocks, skipping softmax computation, value block loading, and subsequent matrix multiplication. However, BLASST primarily targets training-free inference, and its released implementation does not include a backward kernel. Without sparsity-aware training, the accuracy of such methods drops as sparsity increases. We therefore make online block skipping trainable, so that the model can adapt to high sparsity while keeping the low overhead of online selection.

In this paper, we propose **ToSA, a trainable online sparse attention mechanism for efficient long-sequence modeling** (Figure 1). Built on FA2, ToSA reuses the local maximum of each attention block and the running maximum maintained by online softmax to estimate its contribution. As shown in Figure 2, when the estimated contribution falls below a threshold, ToSA skips the subsequent softmax, value loading, and PV computation inside the Triton kernel. To make online block skipping trainable, we develop a sparse backward that reuses the block decisions recorded during the forward pass. It recomputes attention quantities only for retained blocks and accumulates their

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

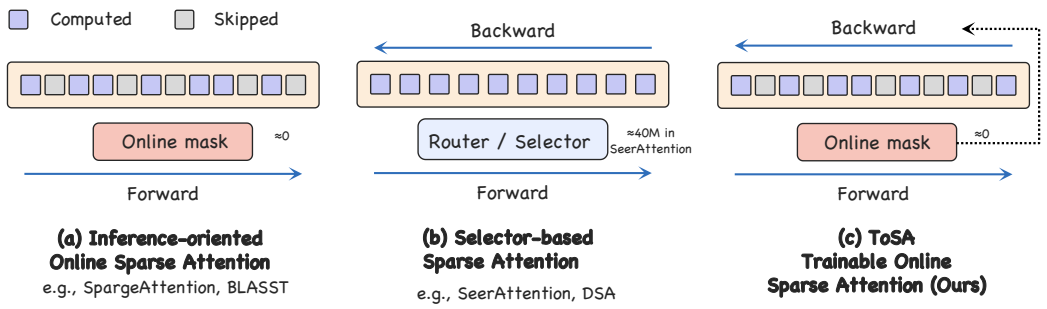


Figure 1: Framework comparison between ToSA and representative sparse attention paradigms. (a) Inference-oriented online sparse attention performs block selection during attention computation but is primarily designed for inference. (b) Selector-based sparse attention relies on an auxiliary router or selector to determine the sparse computation pattern. (c) ToSA performs online block selection without a separate predictor and reuses the resulting mask in sparse backward computation, enabling end-to-end training. The numbers shown below each paradigm denote the approximate number of additional learnable parameters introduced by the sparsity-selection mechanism, excluding the backbone model parameters. For example, SeerAttention introduces approximately 40M auxiliary parameters for its selector, whereas online selection in BLASST and ToSA does not require an additional learnable selector and therefore introduces approximately zero such parameters. BLASST Yuan et al. (2026) primarily targets training-free inference and also discusses sparsity-aware training; however, at the time of our experiments, its released implementation did not include the corresponding backward kernel. Therefore, we categorize it here as an inference-oriented online sparse attention method.

contributions to the query, key, and value gradients. Unlike inference-oriented online sparse attention, ToSA trains the model with the same block-skipping rule used at inference. To help the model better adapt to high sparsity, we further develop a three-stage training schedule. Training starts with dense attention; the sparsification threshold is then gradually increased and finally kept at its target value. This schedule reduces the performance degradation caused by directly applying high sparsity.

We evaluate ToSA on InfiniteBench and LongBench for long-context understanding, and on LVBench, Video-MME, and LongVideoBench for long-video understanding. Compared with the dense fine-tuning baseline, ToSA increases the LongBench score from 38.18 to 39.62 and the LongVideoBench score from 58.19 to 60.02, while remaining comparable or better on the other benchmarks. We measure the efficiency of ToSA on NVIDIA A100 GPUs. At the kernel level, ToSA achieves up to $1.77\times$ speedup over FA2 at a 128K sequence length.

Our main contributions are summarized as follows:

- We propose ToSA, a trainable online sparse attention method for efficient long-sequence modeling. ToSA integrates block skipping into online attention computation without an auxiliary predictor or a precomputed sparse mask, and provides a sparse backward kernel that replays the forward block decisions for end-to-end training.
- We introduce a three-stage training schedule for adapting pretrained dense models to online sparsity. It progressively increases sparsity during fine-tuning, reducing the performance degradation caused by directly applying high sparsity.
- We evaluate ToSA on long-context and long-video understanding benchmarks. ToSA performs comparably to or better than the dense fine-tuning baseline, with up to a $1.77\times$ attention-kernel speedup on 128K sequences.

2 Related Work

Efficient Attention Kernels. A direct way to improve the efficiency of attention is to optimize its GPU execution without changing the attention results. FlashAttention (Dao et al., 2022) introduces an IO-aware attention algorithm that reduces data movement between GPU memory and on-chip memory through tiling. FlashAttention-2 (Dao, 2024) further improves work partitioning and parallelism to achieve higher hardware utilization. These methods provide highly efficient im-

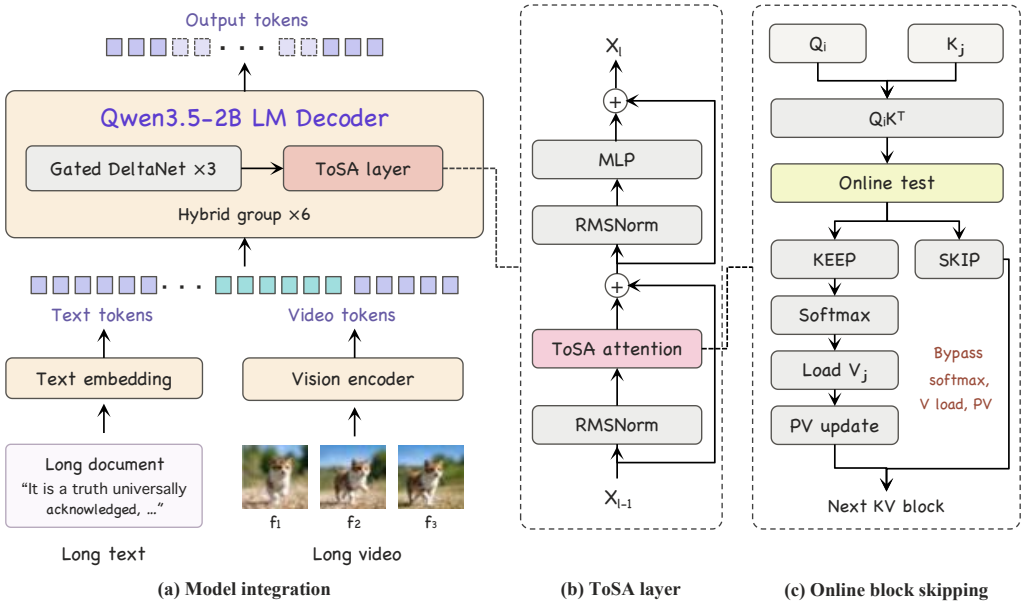


Figure 2: Overview of ToSA. (a) ToSA is applied to the full-attention layers of a long-sequence model for text and video inputs. (b) It replaces standard attention within each full-attention layer while leaving the remaining layer structure unchanged. (c) An online test is performed for each KV block after score computation; retained blocks execute softmax, value loading, and PV computation, while skipped blocks bypass these operations.

plementations of exact dense attention and have become common baselines for long-sequence models. However, they still compute all query-key interactions. Therefore, their computation remains quadratic with sequence length, which becomes increasingly expensive for long-sequence modeling.

Sparse and Memory-Efficient Attention. Sparse attention reduces computation by evaluating only a subset of query-key interactions. Methods such as MInference (Jiang et al., 2024), FlexPrefill (Lai et al., 2025), XAttention (Xu et al., 2025b), SampleAttention (Zhu et al., 2025), and HashAttention (Desai et al., 2024) use different forms of pattern estimation, scoring, or index construction to identify important attention regions. While effective, these selection procedures introduce additional overhead before sparse computation. A related line of work reduces memory cost by pruning, compressing, or selectively accessing the KV cache. H_2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2024), Quest (Tang et al., 2024), SnapKV (Li et al., 2024), TidalDecode (Yang et al., 2025), RazorAttention (Tang et al., 2025), and RocketKV (Behnam et al., 2025) reduce the number of KV entries that need to be stored or accessed during long-context inference. These methods mainly target memory footprint and memory traffic, especially during decoding.

Online and Trainable Sparse Attention. Another line of work integrates sparsity decisions more closely with the attention computation. SpargeAttention (Zhang et al., 2025) uses a two-stage filtering mechanism, including an online softmax-aware filter that skips low-contribution computation. BLASST (Yuan et al., 2026) further reuses statistics already available during online softmax to identify negligible attention blocks. It requires no separate proxy score or pre-computation pass and can directly skip softmax operations, value loading, and the subsequent matrix multiplication. This makes online sparsification particularly attractive for practical long-context attention. Most online sparse attention methods, however, are designed primarily as inference-time replacements for densely trained attention. Meanwhile, Native Sparse Attention (NSA) (Yuan et al., 2025) demonstrates that sparse attention can be trained end-to-end through a hardware-aware architecture that combines compressed representations, selected blocks, and local attention. Its sparsity mechanism, however, relies on a dedicated sparse attention architecture and selection procedure. In contrast, ToSA focuses on making online block skipping itself trainable.

Algorithm 1 ToSA Forward and Sparse Backward Computation

(a) Forward: Online Selection and Recording

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, threshold τ, scale s
Ensure: $\mathbf{O}, \mathbf{L}$, selection mask $\mathbf{M}$

$\mathbf{M} \leftarrow \mathbf{0}$
for each query block $\mathbf{Q}_i$ **do**
  $\mathbf{m}_i \leftarrow -\infty,\ \boldsymbol{\ell}_i \leftarrow 0,\ \mathbf{A}_i \leftarrow 0$
  for each valid KV block $(\mathbf{K}_j, \mathbf{V}_j)$ **do**
    $\mathbf{S}_{ij} \leftarrow s\,\mathbf{Q}_i\mathbf{K}_j^\top$
    $\mathbf{S}_{ij} \leftarrow \mathrm{CausalMask}(\mathbf{S}_{ij})$
    $\widetilde{\mathbf{m}}_{ij} \leftarrow \mathrm{rowmax}(\mathbf{S}_{ij})$
    $\widehat{\mathbf{m}}_{ij} \leftarrow \max(\mathbf{m}_i, \widetilde{\mathbf{m}}_{ij})$
    $\rho_{ij} \leftarrow \max_r(\widetilde{m}_{ij,r} - \widehat{m}_{ij,r})$
    /* record block selection for backward */
    if $\tau = 0$ **then**
      $M_{ij} \leftarrow 1$
    else
      $M_{ij} \leftarrow \mathbb{I}[\rho_{ij} \geq \log\tau]$
    end if
    if $M_{ij} = 0$ **then**
      continue
    end if
    /* update the retained block */
    $\mathbf{E}_{ij} \leftarrow \exp(\mathbf{S}_{ij} - \widehat{\mathbf{m}}_{ij})$
    $\boldsymbol{\alpha}_i \leftarrow \exp(\mathbf{m}_i - \widehat{\mathbf{m}}_{ij})$
    $\boldsymbol{\ell}_i \leftarrow \boldsymbol{\alpha}_i \odot \boldsymbol{\ell}_i + \mathrm{rowsum}(\mathbf{E}_{ij})$
    $\mathbf{A}_i \leftarrow \boldsymbol{\alpha}_i \odot \mathbf{A}_i + \mathbf{E}_{ij}\mathbf{V}_j$
    $\mathbf{m}_i \leftarrow \widehat{\mathbf{m}}_{ij}$
  end for
  $\mathbf{O}_i \leftarrow \mathbf{A}_i \oslash \boldsymbol{\ell}_i$
  $\mathbf{L}_i \leftarrow \mathbf{m}_i + \log\boldsymbol{\ell}_i$
end for
Save $\mathbf{L}$ and $\mathbf{M}$ for sparse backward
Return: $\mathbf{O}, \mathbf{L}, \mathbf{M}$

(b) Backward: Sparse Replay of Recorded Blocks

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}, \mathbf{L}, \mathbf{M}, d\mathbf{O}, s$
Ensure: $d\mathbf{Q}, d\mathbf{K}, d\mathbf{V}$

$\boldsymbol{\Delta} \leftarrow \mathrm{rowsum}(\mathbf{O} \odot d\mathbf{O})$
$d\mathbf{Q}, d\mathbf{K}, d\mathbf{V} \leftarrow \mathbf{0}$
/* replay retained blocks for $d\mathbf{Q}$ */
for each query block i **do**
  for each causally valid KV block j **do**
    if $M_{ij} = 0$ **then**
      continue
    end if
    $\mathbf{S}_{ij} \leftarrow s\,\mathbf{Q}_i\mathbf{K}_j^\top$
    $\mathbf{S}_{ij} \leftarrow \mathrm{CausalMask}(\mathbf{S}_{ij})$
    $\mathbf{P}_{ij} \leftarrow \exp(\mathbf{S}_{ij} - \mathbf{L}_i)$
    $d\mathbf{S}_{ij} \leftarrow \mathbf{P}_{ij} \odot (d\mathbf{O}_i\mathbf{V}_j^\top - \boldsymbol{\Delta}_i)$
    $d\mathbf{Q}_i \leftarrow d\mathbf{Q}_i + s\,d\mathbf{S}_{ij}\mathbf{K}_j$
  end for
end for
/* replay retained blocks for $d\mathbf{K}, d\mathbf{V}$ */
for each KV block j **do**
  for each compatible query block i **do**
    if $M_{ij} = 0$ **then**
      continue
    end if
    $\mathbf{S}_{ij} \leftarrow \mathrm{CausalMask}(s\,\mathbf{Q}_i\mathbf{K}_j^\top)$
    $\mathbf{P}_{ij} \leftarrow \exp(\mathbf{S}_{ij} - \mathbf{L}_i)$
    $d\mathbf{S}_{ij} \leftarrow \mathbf{P}_{ij} \odot (d\mathbf{O}_i\mathbf{V}_j^\top - \boldsymbol{\Delta}_i)$
    $d\mathbf{K}_j \leftarrow d\mathbf{K}_j + s\,d\mathbf{S}_{ij}^\top\mathbf{Q}_i$
    $d\mathbf{V}_j \leftarrow d\mathbf{V}_j + \mathbf{P}_{ij}^\top d\mathbf{O}_i$
  end for
end for
Reduce $d\mathbf{K}, d\mathbf{V}$ over query heads sharing each KV head
Return: $d\mathbf{Q}, d\mathbf{K}, d\mathbf{V}$

3 METHOD

ToSA integrates block selection into online softmax and records the selection decisions for sparse backward computation. As summarized in Algorithm 1, the forward pass evaluates each causally valid KV block after computing its attention scores. Retained blocks update the attention output, while skipped blocks bypass the remaining softmax and value computations. The backward pass replays the recorded block decisions. We further introduce sparsity progressively when fine-tuning a pretrained model.

3.1 ONLINE BLOCK SELECTION

Consider a query block Q_i as the forward kernel visits causally valid KV blocks (K_j, V_j). For each block, ToSA first computes the masked attention scores $S_{ij} = sQ_iK_j^\top$, where $s = 1/\sqrt{d}$ and d is the head dimension. At this point, the kernel has accessed K_j but has not loaded V_j. It uses the scores to decide whether the block should contribute to the attention output.

For query row r, let $\widetilde{m}_{ij,r}$ be its maximum score in the current block and $m_{i,r}$ its running maximum over previously retained blocks. The candidate maximum is $\widehat{m}_{ij,r} = \max(m_{i,r}, \widetilde{m}_{ij,r})$. We retain

the KV block if at least one valid query row satisfies

$$M_{ij} = \mathbb{I}\Big[\max_r \left(\widetilde{m}_{ij,r} - \widehat{m}_{ij,r}\right) \geq \log \tau\Big], \tag{1}$$

where $\tau \in (0, 1]$ controls the selection threshold. We define $\tau = 0$ to retain every causally valid block. Intuitively, a block is skipped when its largest scores are small relative to the running maxima for all query rows. The decision relies on statistics already available after the $QK^\top$ computation and requires no separate predictor or precomputed sparse mask.

If $M_{ij} = 0$, the kernel proceeds to the next KV block without loading V_j, computing the exponentiated attention weights, or performing the PV product. If $M_{ij} = 1$, it rescales the running online-softmax states and adds the current block's contribution to the normalizer and output accumulator, as shown in Algorithm 1. After the final KV block, the kernel normalizes the accumulator to obtain O_i. It saves the log-sum-exp L_i and block decisions M_{ij}, which the backward pass uses to recompute weights for the retained blocks.

3.2 Sparse Backward Computation

The backward pass reuses the block decisions M_{ij} made in the current forward pass. It does not differentiate through the discrete selection rule: gradients are computed only for blocks with $M_{ij} = 1$. For each retained block, the kernel recomputes its causally masked scores and uses the saved log-sum-exp L_i to recover the normalized attention weights $P_{ij} = \exp(S_{ij} - L_i)$. Thus, the forward pass does not need to store the attention weights of individual blocks. Conditioned on the recorded mask, this computes the gradient of the retained-block attention used in the forward pass, maintaining consistency between the sparse forward and backward computations.

Given the output gradient dO_i, multiplication by $V_j^\top$ propagates it to the attention weights. The row-wise term $\Delta_i = \mathrm{rowsum}(O_i \odot dO_i)$ accounts for the coupling between weights introduced by softmax normalization. Together, they give the score gradient

$$dS_{ij} = P_{ij} \odot \left(dO_i V_j^\top - \Delta_i\right). \tag{2}$$

For each retained block, dS_{ij} contributes to dQ_i and dK_j through multiplication with K_j and Q_i, respectively. The gradient dO_i is propagated through P_{ij} to obtain the contribution to dV_j.

As shown in Algorithm 1, computing dQ_i fixes query block i and accumulates contributions from its retained KV blocks. Computing dK_j and dV_j instead fixes KV block j and accumulates contributions from the query blocks that retained it. The latter traversal recomputes the required attention quantities for each retained pair. Under grouped-query attention, contributions from query heads sharing a KV head are summed into the corresponding key and value gradients.

3.3 Progressive Sparsity Training

We introduce sparsity gradually during fine-tuning. Training begins with $\tau = 0$, retaining all causally valid blocks. We then increase τ toward its target value and continue training at that value. At every stage, the backward pass uses the decisions recorded by the corresponding forward pass. The target threshold may differ across attention layers; the schedule and layer-specific values are given in the experimental setup.

4 Experiments

4.1 Experimental Setup

Implementation and training data. All of our models are based on Qwen3.5-2B. We replace its six full-attention layers with ToSA while keeping the remaining model architecture unchanged. We construct separate instruction-tuning sets for long-video and long-context training. For the long-video setting, we follow the data construction protocol of VideoNSA Song et al. (2026) and use approximately half of its long-video instruction data derived from LLaVA-Video-178K. For the long-context setting, we use approximately 100K training samples from ChatQA2 Xu et al. (2025a). The two settings allow us to evaluate ToSA on different types of long-sequence inputs. The video

Table 1: Kernel latency under different sparsity levels on an NVIDIA A100-SXM4-80GB GPU. We benchmark with $B = 1$, $H = 2$, $H_Q = 8$, $D = 256$, and $T = 131{,}072$ (128K). Speedup is measured relative to FA2.

Metric	FA2	ToSA								
	Dense	10%	20%	30%	40%	50%	60%	70%	80%	90%
Latency (ms)	1411.0	1342.2	1273.6	1208.3	1139.5	1069.9	1001.3	932.9	863.8	**798.7**
Speedup	1.00×	1.05×	1.11×	1.17×	1.24×	1.32×	1.41×	1.51×	1.63×	**1.77**×

and text settings use maximum sequence lengths of 65,536 and 18,432, respectively. For the six full-attention layers equipped with ToSA, we use layer-specific sparsity thresholds: $\tau = 0.95$ for Layers 3, 7, and 23, and $\tau = 0.85$ for Layers 11, 15, and 19. These values are used as the target thresholds of the progressive sparsity schedule described in Section 3. Additional implementation details are provided in Appendix A.

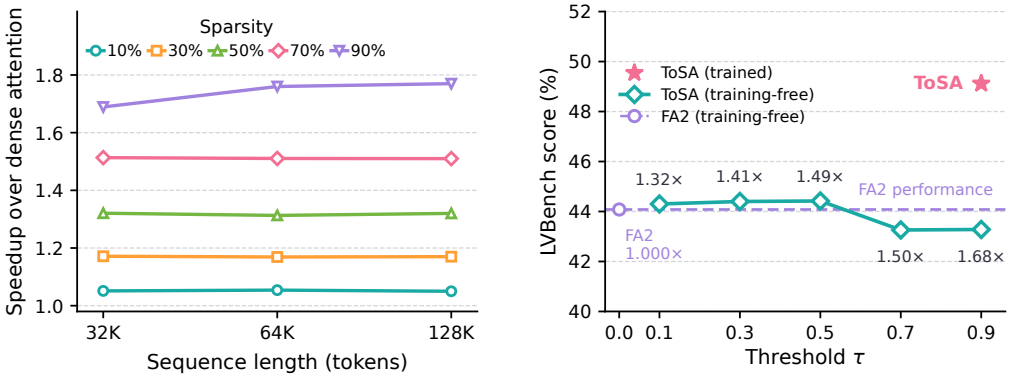
Training recipe. We use a three-stage sparsification schedule during fine-tuning. We first keep $\tau = 0$ for the initial 500 optimization steps, so that training starts with dense attention. We then progressively increase τ from 0 to its layer-specific target value over a transition period corresponding to 10% of the total optimization steps. For the remainder of training, τ is kept fixed at its target value.

Benchmarks. For long-video understanding, we evaluate on LongVideoBench Wu et al. (2024), LVBench Wang et al. (2025a), and Video-MME Fu et al. (2025). For long-context understanding, we evaluate on LongBench Bai et al. (2024) and a subset of InfiniteBench Zhang et al. (2024). Following the LongBench evaluation setting of AHN Fang et al. (2025), we use six tasks with average input lengths above 8K tokens: DuReader, HotpotQA, MuSiQue, NarrativeQA, QMSum, and TriviaQA. Their average input lengths range from approximately 8K to 18K tokens and cover question answering, multi-hop reasoning, and summarization. On InfiniteBench, we evaluate four representative tasks from the 128K subset: English free-form QA (En.QA), Chinese free-form QA (Zh.QA), English multiple-choice long-book QA (En.MC), and long-context code debugging (Code.Debug). Following AHN Fang et al. (2025), we additionally report the average over En.QA and Zh.QA for direct comparison, alongside the average over all four tasks. We follow the official evaluation protocol and metric of each benchmark. Detailed evaluation configurations are provided in Appendix A.2. For all video benchmarks, we use the same video preprocessing configuration. Videos are sampled at 1 FPS with at most 256 frames. We set the minimum number of input pixels to 65,536 and the maximum number of input pixels to 737,280. For the main benchmark evaluations, we use the same layer-wise configuration: $\tau = 0.95$ for Layers 3, 7, and 23, and $\tau = 0.85$ for Layers 11, 15, and 19. Training-free analyses and ablations use the thresholds specified in their respective descriptions.

Baselines. Our primary controlled baseline is Dense-SFT, which uses the same Qwen3.5-2B backbone, training data, and optimization settings as ToSA, while retaining dense FA2 attention. This comparison isolates the effect of replacing dense attention with ToSA. We also use the original Qwen3.5-2B checkpoint as a training-free reference in our analysis. In addition, we report results from representative efficient long-video and long-context methods for reference. Since these methods use different backbone models and training recipes, their results are not treated as strictly controlled comparisons. We consider BLASST (Yuan et al., 2026) as an inference-time baseline. Since its released implementation primarily targets Hopper and Blackwell GPUs, whereas our experiments are conducted on NVIDIA A100 GPUs, we reimplement BLASST in Triton for our hardware setup and use this implementation as the inference-time baseline.

4.2 Kernel Efficiency

We first examine whether online block skipping yields actual GPU runtime savings. We benchmark ToSA against FA2 on an NVIDIA A100-SXM4-80GB GPU, using the attention-head configuration of the full-attention layers in Qwen3.5-2B. At a sequence length of 128K, the latency of ToSA decreases as block sparsity increases. At 90% block sparsity, latency falls from 1411.0 ms with FA2 to 798.7 ms with ToSA, giving a $1.77\times$ kernel speedup (Table 1). These measurements establish the



(a) Sequence-length scaling.

(b) Training-free trade-off vs our ToSA.

Figure 3: (a) Sequence-length scaling. Kernel speedup of ToSA over FA2 at sequence lengths of 32K, 64K, and 128K tokens, where each curve uses a fixed uniform sparsity. (b) Training-free trade-off versus trained ToSA. We apply ToSA directly to the original Qwen3.5-2B checkpoint without fine-tuning and evaluate its LVBench performance under different uniform thresholds. The star denotes our trained ToSA using the layer-wise sparsity thresholds adopted in the main experiments. Annotations indicate kernel speedups relative to FA2, and the dashed line denotes the training-free FA2 performance.

runtime benefit of block skipping; the remaining question is how much sparsity an existing model can tolerate without losing task accuracy.

4.3 Training-Free Accuracy–Efficiency Trade-off

To study this question, we apply ToSA directly to the original Qwen3.5-2B checkpoint without additional fine-tuning. We use a uniform threshold τ across all ToSA layers and evaluate LVBench accuracy and attention-kernel speedup. As shown in Figure 3(b), increasing τ from 0.1 to 0.5 raises kernel speedup from $1.32\times$ to $1.49\times$, while the LVBench score remains close to the FA2 baseline. At $\tau = 0.7$, speedup changes only slightly to $1.50\times$, but accuracy drops below the baseline. Increasing τ to 0.9 further improves speedup to $1.68\times$, while accuracy remains below the baseline. These results motivate adapting the model to sparse computation through fine-tuning. This raises a central question: *Can we preserve task performance at high sparsity?*

4.4 Task Performance

To address the accuracy loss observed under aggressive training-free sparsification, we introduce ToSA into supervised fine-tuning of Qwen3.5-2B. Its sparse backward pass uses the block-selection mask from the forward pass to compute gradients for retained blocks, as detailed in Section 3. We first use a dense warmup and then progressively increase the sparsity threshold during training, allowing the model to adapt gradually to block skipping. For the main task evaluations, we use the layer-wise target thresholds specified in Section 4.1 during both sparse training and inference. The uniform-threshold configuration is evaluated separately in Table 6.

Table 5: Training-schedule ablation with layer-wise τ.

Warmup	Prog. τ	LVBench	LongBench
✗	✗	47.94	38.10
✓	✗	48.11	39.17
✗	✓	48.29	39.26
✓	✓	**49.12**	**39.62**

Table 6: Sparse-training ablation with uniform $\tau = 0.9$.

Train.	Infer.	LVBench	LongBench
✗	✗	48.53	38.18
✗	✓	47.72	38.07
✓	✗	48.97	38.49
✓	✓	**49.00**	**39.28**

We compare the resulting model with the controlled Dense-SFT baseline. On LongVideoBench, Video-MME, and LVBench, ToSA scores 60.02, 63.90, and 49.12, respectively, compared with

Table 2: Performance comparison on long-video understanding benchmarks. The best results are highlighted in bold, and the second-best results are underlined. † Results are taken from the unified Qwen2.5-VL-7B evaluation reported by VecAttention. ‡ BLASST is reimplemented in Triton on Qwen3.5-2B and evaluated on NVIDIA A100 GPUs. "–" denotes results not reported under the corresponding evaluation setting.

Method	Params.	LongVideoBench	Video-MME	LVBench
General MLLMs				
InternVL3.5 (Wang et al., 2025b)	2B	57.4	58.4	–
Qwen3-VL (Bai et al., 2025)	2B	55.6	61.9	47.4
NEO-ov (Diao et al., 2026)	2B	56.8	60.4	43.3
Penguin-VL (Zhang et al., 2026)	2B	59.5	57.4	–
Token Compression Methods				
VideoChat-Flash (Li et al., 2026)	2B	58.3	57.0	44.9
Sparse Attention Methods				
XAttention (Xu et al., 2025b)†	7B	58.5	63.0	–
VecAttention (Liu et al., 2026)†	7B	59.4	**64.8**	–
VideoNSA (Song et al., 2026)	7B	<u>60.0</u>	–	–
BLASST (Yuan et al., 2026) (Triton reimpl.)‡	2B	51.38	55.31	43.25
Ours				
Dense-SFT	2B	58.19	<u>64.22</u>	<u>48.53</u>
ToSA	2B	**60.02**	63.90	**49.12**

Table 3: Performance on six LongBench tasks with average input lengths above 8K tokens. Avg. (5) averages HotpotQA through TriviaQA, while Avg. (6) averages all six tasks. A dash indicates an unreported result. † BLASST is reimplemented in Triton and evaluated on NVIDIA A100 GPUs. Backbone models and evaluation settings differ across studies.

Method	DuReader	Hotpot	MuSiQue	NarrQA	QMSum	TriviaQA	Avg. (5)	Avg. (6)
AHN-DN (Fang et al., 2025) (Qwen2.5-3B)	25.12	42.83	19.78	19.11	22.35	86.17	38.05	35.89
AHN-DN (Fang et al., 2025) (Qwen2.5-7B)	26.42	54.24	29.30	25.08	21.69	89.49	43.96	41.04
NOSA (Huang et al., 2025) (3B)	–	40.00	19.70	21.20	23.50	88.20	38.52	–
InfLLMv2 (Zhao et al., 2026) (3B)	–	30.10	14.70	19.70	23.20	87.20	34.98	–
FastKV (Jo et al., 2026) (Llama-3.2-3B)	–	47.58	21.93	24.02	23.24	89.22	41.20	–
BLASST (Yuan et al., 2026) (Triton reimpl.)†	28.09	46.22	29.10	17.73	21.48	84.63	39.83	37.88
Ours								
Dense-SFT (Qwen3.5-2B)	32.80	52.22	27.08	17.21	15.83	83.92	39.25	38.18
ToSA (Qwen3.5-2B)	33.94	50.18	31.31	17.75	19.07	85.46	40.75	39.62

58.19, 64.22, and 48.53 for Dense-SFT (Table 2). On the six LongBench tasks, ToSA achieves an average score of 39.62 versus 38.18 for Dense-SFT (Table 3). On the four selected InfiniteBench tasks, the average scores are 41.00 versus 37.65, respectively (Table 4). Across these evaluations, training with sparse attention preserves task performance relative to the dense counterpart.

4.5 Ablation Studies And Further Analysis

Training recipe. Table 5 isolates the contributions of dense warmup and progressive sparsification. All four variants are evaluated using the same layer-wise inference thresholds, so their differences reflect the training schedule rather than a change in the inference configuration. Starting from the variant with neither component, dense warmup alone improves LVBench from 47.94 to 48.11 and LongBench from 38.10 to 39.17. Combining both components yields the highest scores, 49.12 on LVBench and 39.62 on LongBench. These results suggest that gradually introducing sparsity is particularly useful, while the initial dense stage provides an additional benefit when used together with the progressive schedule.

Sparse training and inference. Table 6 disentangles the effects of sparse training and sparse inference. For this ablation, sparse inference uses a uniform threshold of $\tau = 0.9$ across all six full-attention layers, instead of the layer-wise thresholds used in Table 5. Applying sparse inference directly to the densely trained model reduces performance from 48.53 to 47.72 on LVBench and from

Table 4: Performance on four representative tasks from the 128K subset of InfiniteBench. QA Avg. is the average of En.QA and Zh.QA, while Avg. is the average over all four displayed tasks. A dash indicates an unreported or unavailable result. § BLASST is reimplemented in Triton and evaluated on NVIDIA A100 GPUs. Backbone models and evaluation settings differ across studies.

Method	**En.QA**	**Zh.QA**	**En.MC**	**Code.Debug**	**QA Avg.**	**Avg.**
AHN-DN (Qwen2.5-3B) (Fang et al., 2025)	10.61	16.41	–	–	13.51	–
AHN-GDN (Qwen2.5-7B) (Fang et al., 2025)	13.37	20.48	–	–	16.93	–
MiniCPM-2B-128K (Hu et al., 2024)	16.33	23.06	29.69	22.08	19.70	22.79
MInference (Jiang et al., 2024)	8.50	12.14	65.94	22.08	10.32	27.17
FlexPrefill (Lai et al., 2025)	12.60	11.63	57.21	22.84	12.12	26.07
BLASST (Yuan et al., 2026) (Qwen3.5-2B)§	16.49	16.98	60.43	40.89	16.74	33.70
Ours						
Dense-SFT (Qwen3.5-2B)	16.40	22.88	64.38	46.95	19.64	37.65
ToSA (Qwen3.5-2B)	15.83	28.24	71.18	48.73	22.04	41.00

38.18 to 38.07 on LongBench. Sparse training without sparse inference yields 48.97 on LVBench and 38.49 on LongBench, resulting in modest improvements even under dense inference. In contrast, when sparse training and inference are used together, performance increases to 49.00 on LVBench and 39.28 on LongBench. These results suggest that training with the same block-skipping path used at inference helps the model adapt to sparse computation and mitigates the degradation caused by applying sparsity only at inference time.

Efficiency across sequence lengths. To better understand how ToSA scales with context length, we evaluate its kernel efficiency at sequence lengths of 32K, 64K, and 128K tokens under different block sparsity levels. Figure 3(a) shows that speedups remain largely stable across these lengths at lower sparsity levels, while a modest increase is observed at 90% sparsity, from $1.69\times$ at 32K to $1.76\times$ at 64K and $1.77\times$ at 128K. At each sequence length, higher sparsity consistently yields greater speedup. At 128K tokens, increasing sparsity from 10% to 90% improves speedup from $1.05\times$ to $1.77\times$. For figure 5, each curve shows the fraction of skipped KV blocks for each query position. The dashed blue line marks the average skip ratio for the corresponding example. In the examples shown, skip ratios generally increase toward later query positions and are higher at larger thresholds and longer sequence lengths. These results show that the efficiency gains depend on both context length and the selection threshold, with larger gains observed under more aggressive sparsification.

Figure 6 visualizes the block-level selection patterns in Layer 7 under different thresholds and sequence lengths. At 32,768 tokens, increasing τ from 0.1 to 0.5 and 0.9 raises sparsity from 1.5% to 29.6% and 62.9%, respectively. Skipping also expands from a few localized regions to much of the causally valid attention region. At a fixed threshold, the longer sequences shown exhibit higher sparsity: at $\tau = 0.5$, the skipped fraction increases from 29.6% at 32,768 tokens to 61.9% at 125,312 tokens. Retained blocks remain distributed across the valid region, reflecting nonuniform selection rather than a fixed local window. Together with Figure 3(a), these results illustrate two complementary aspects of efficiency: the sparsity achieved under a given threshold and the kernel speedup obtained at a given sparsity level.

5 Conclusion

We introduced ToSA, a trainable online sparse attention mechanism for efficient long-sequence modeling. By skipping low-contribution KV blocks during attention computation, ToSA reduces attention cost without requiring a separate sparsity selection stage and can be directly integrated into sparsity-aware training. Our Triton kernel achieves up to $1.77\times$ speedup over FlashAttention-2 on an NVIDIA A100 GPU. Across long-video and long-context benchmarks, ToSA largely preserves the performance of its dense counterpart. Overall, our results show that trainable online sparse attention provides a practical way to improve the efficiency of long-sequence processing while maintaining model capability.

Reproducibility Statement

We provide the model configuration, training settings, evaluation protocols, and attention-kernel parameters in Section 4 and the appendix. These details are intended to support reproduction of the reported accuracy and efficiency results.

AI Use Statement

AI tools were used to assist with language editing and the organization of the manuscript. The authors reviewed the resulting text and are responsible for the technical content, experiments, and conclusions.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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.

Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. Rocketkv: Accelerating long-context llm inference via two-stage kv cache compression. *arXiv preprint arXiv:2502.14051*, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

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.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Matei Zaharia, Joseph E Gonzalez, and Ion Stoica. Hashattention: Semantic sparsity for faster inference. *arXiv preprint arXiv:2412.14468*, 2024.

Haiwen Diao, Jiahao Wang, Penghao Wu, Yuhao Dong, Yuwei Niu, Yue Zhu, Zhongang Cai, Weichen Fan, Linjun Dai, Silei Wu, et al. From pixels to words–towards native one-vision models at scale. *arXiv preprint arXiv:2605.28820*, 2026.

Yunhao Fang, Weihao Yu, Shu Zhong, Qinghao Ye, Xuehan Xiong, and Lai Wei. Artificial hippocampus networks for efficient long-context modeling. *arXiv preprint arXiv:2510.07318*, 2025.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118. IEEE, 2025.

Yizhao Gao, Zhichen Zeng, DaYou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden So, Ting Cao, Fan Yang, et al. Seerattention: Self-distilled attention gating for efficient long-context prefilling. *Advances in Neural Information Processing Systems*, 38:55846–55869, 2026.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Yuxiang Huang, Pengjie Wang, Jicheng Han, Weilin Zhao, Zhou Su, Ao Sun, Hongya Lyu, Hengyu Zhao, Yudong Wang, Chaojun Xiao, et al. Nosa: Native and offloadable sparse attention. *arXiv preprint arXiv:2510.13602*, 2025.

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. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. Fastkv: Decoupling of context reduction and kv cache compression for prefill-decoding acceleration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 32167–32186, 2026.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

Xinhao Li, Yi Wang, Jiashuo Yu, Xiangyu Zeng, Yuhan Zhu, Haian Huang, Jianfei Gao, Kunchang Li, Yinan He, Chenting Wang, et al. Videochat-flash: Hierarchical compression for long-context video modeling. In *International Conference on Learning Representations*, volume 2026, pp. 109089–109117, 2026.

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. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Anmin Liu, Ruixuan Yang, Huiqiang Jiang, Bin Lin, Minmin Sun, Yong Li, Chen Zhang, and Tao Xie. Vecattention: Vector-wise sparse attention for accelerating long context inference. *arXiv preprint arXiv:2603.29494*, 2026.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *Advances in Neural Information Processing Systems*, 38:17790–17815, 2026.

Enxin Song, Wenhao Chai, Shusheng Yang, Ethan Armand, Xiaojun Shan, Haiyang Xu, Jianwen Xie, and Zhuowen Tu. Videonsa: Native sparse attention scales video understanding. In *International Conference on Learning Representations*, volume 2026, pp. 25271–25310, 2026.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads. In *International Conference on Learning Representations*, volume 2025, pp. 16632–16646, 2025.

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.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22958–22967. IEEE, 2025a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa 2: Bridging the gap to proprietary llms in long context and rag capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 67244–67258, 2025a.

Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025b.

Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. In *International Conference on Learning Representations*, volume 2025, pp. 5823–5844, 2025.

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, et al. Blasst: Dynamic blocked attention sparsity via softmax thresholding. *Proceedings of Machine Learning and Systems*, 8:843–859, 2026.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.

Boqiang Zhang, Lei Ke, Ruihan Yang, Qi Gao, Tianyuan Qu, Rossell Chen, Dong Yu, et al. Penguin-vl: Exploring the efficiency limits of vlm with llm-based vision encoders. *arXiv preprint arXiv:2603.06569*, 2026.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. ∞Bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024.

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.

Weilin Zhao, Zihan Zhou, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, et al. Infllm-v2: Dense-sparse switchable attention for seamless short-to-long adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 69374–69390, 2026.

Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Xiao Chuanfu, Dahua Lin, and Chao Yang. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. *Proceedings of Machine Learning and Systems*, 7, 2025.

A Implementation Details

A.1 Training Details

We fine-tune Qwen3.5-2B separately for long-video and long-context tasks. The detailed training configurations are summarized in Table 7. Both settings are trained for one epoch on 8 NVIDIA A100-SXM4-80GB GPUs with an effective global batch size of 32. We use BF16 precision with TF32 enabled and a cosine learning-rate schedule.

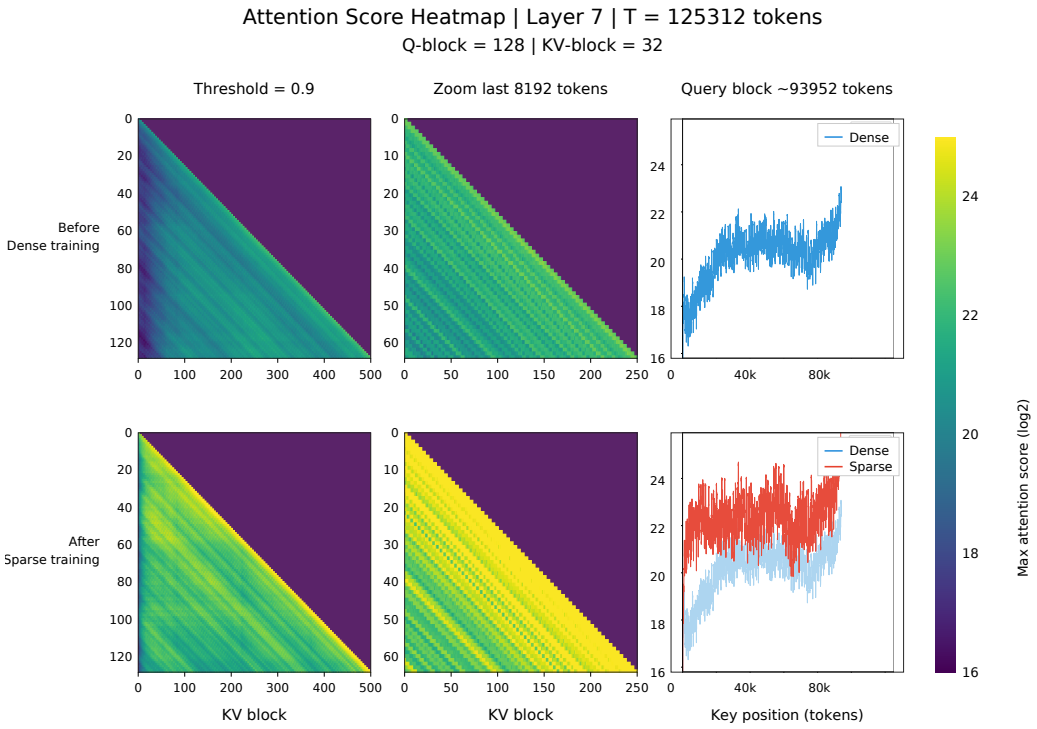


Figure 4: Attention score visualization for Layer 7 at sequence length $T = 125{,}312$ with threshold $\tau = 0.9$.

Table 7: Training configurations for long-video and long-text fine-tuning.

Configuration	**Video SFT**	**Text SFT**
Base model	Qwen3.5-2B	Qwen3.5-2B
GPUs	8× A100-80GB	8× A100-80GB
Precision	BF16	BF16
TF32	Enabled	Enabled
Batch size / GPU	1	1
Gradient accumulation	4	4
Effective batch size	32	32
Learning rate	1×10^{-6}	1×10^{-6}
Weight decay	0.01	0.05
LR scheduler	Cosine	Cosine
LR warmup ratio	0.1	0.1
Epochs	1	1
Maximum sequence length	65,536	18,432
Gradient clipping	1.0	1.0
Layer-wise τ	$\begin{cases} L3 : 0.95 \\ L7 : 0.95 \\ L11 : 0.85 \\ L15 : 0.85 \\ L19 : 0.85 \\ L23 : 0.95 \end{cases}$	$\begin{cases} L3 : 0.95 \\ L7 : 0.95 \\ L11 : 0.85 \\ L15 : 0.85 \\ L19 : 0.85 \\ L23 : 0.95 \end{cases}$

A.2 Evaluation Details

We evaluate ToSA on both long-video and long-context benchmarks. For long-video evaluation, we report results on LongVideoBench, LVBench, and Video-MME. For long-text evaluation, we use LongBench and InfiniteBench. Unless otherwise specified, all dense baselines use FlashAttention-

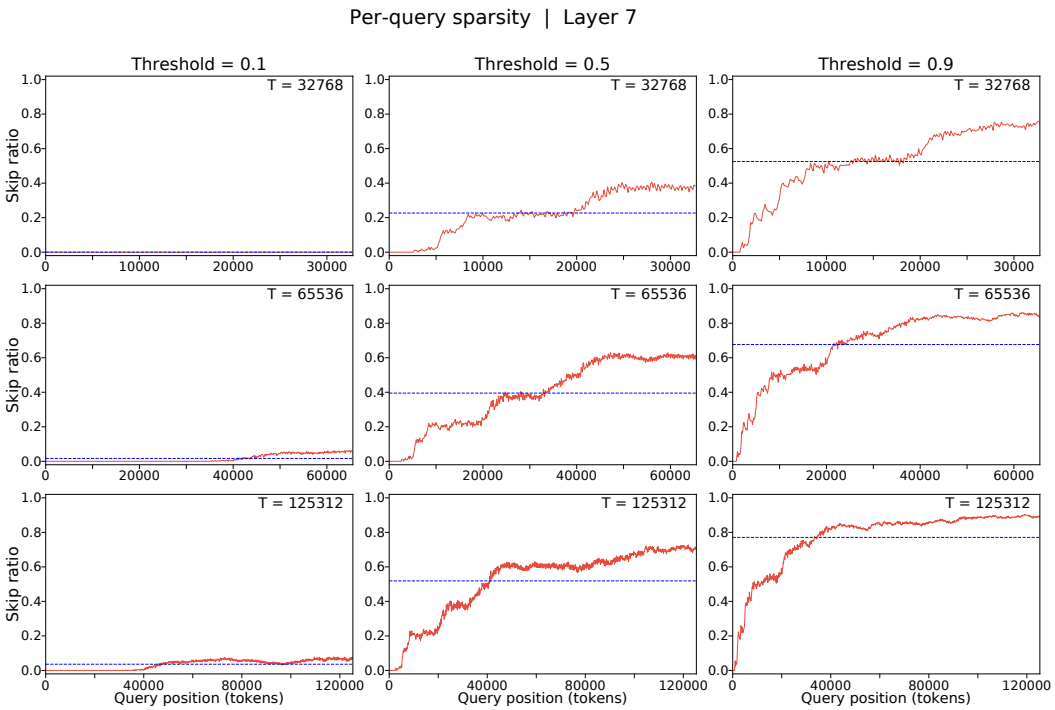


Figure 5: Per-query skip ratio in Layer 7 under different sparsity thresholds and sequence lengths. Figures 5 and 6 use different input samples; therefore, the achieved sparsity can differ even at the same threshold and sequence length.

Table 8: Evaluation configurations for long-video and long-text benchmarks. The same layer-wise sparsity configuration is used for all main benchmark evaluations.

Configuration	**Value**
Long-video evaluation	
Benchmarks	LongVideoBench, LVBench, Video-MME
Sampling rate	1 FPS
Maximum frames	256
Minimum pixels	65,536
Maximum pixels	737,280
Long-text evaluation	
LongBench context length	32K
InfiniteBench context length	128K

2 (FA2), while sparse inference uses the ToSA kernel. The detailed evaluation configurations are summarized in Table 8.

A.3 Kernel Configuration

We implement ToSA in Triton. The kernel configuration used in our experiments is summarized in Table 9. We use a query block size of 128 and a KV block size of 32, with 8 warps per program. All kernel-level experiments are conducted on an NVIDIA A100-SXM4-80GB GPU using BF16 precision. We use FlashAttention-2 as the dense attention baseline under the same hardware and precision settings.

Table 9: Kernel configuration used for ToSA and kernel-level benchmarking.

Configuration	Value
GPU	NVIDIA A100-SXM4-80GB
Head dimension	256
Number of query heads	8
Number of KV heads	2
GQA ratio	4
BLOCK_M	128
BLOCK_N	32
Number of warps	8

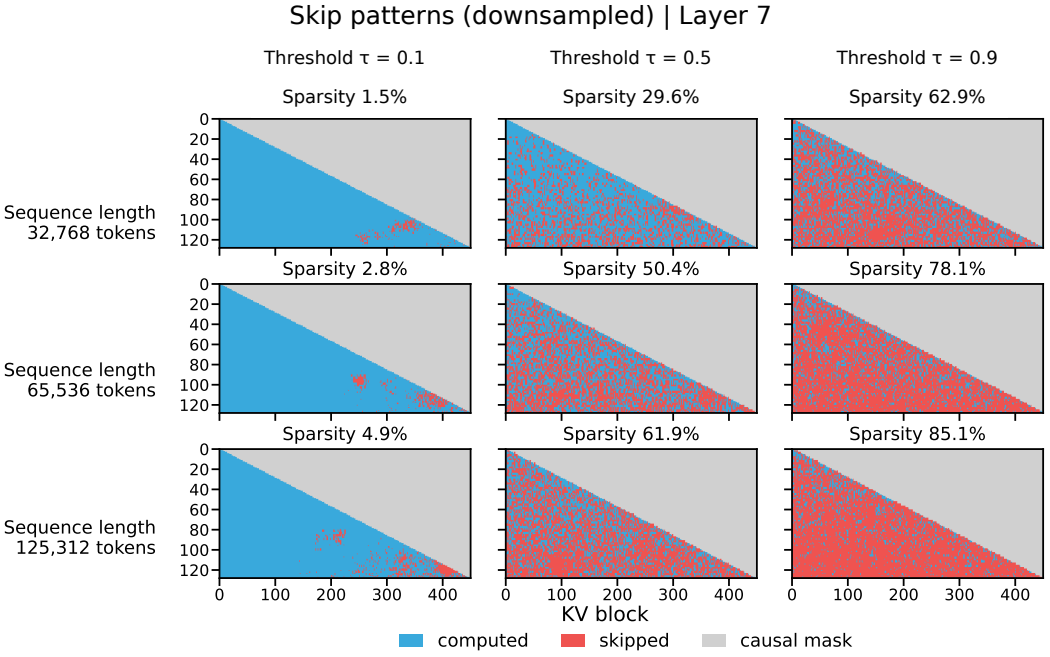


Figure 6: Visualization of skip patterns in Layer 7 under different sparsity thresholds and sequence lengths. Blue regions denote computed KV blocks, red regions denote skipped KV blocks, and gray regions denote the causal mask.

A.4 Additional Analysis of Sparse Behavior

We provide additional visualizations to better understand the sparse behavior learned by ToSA. As the threshold increases, ToSA skips more blocks. Higher sparsity also emerges more clearly at longer sequence lengths. Figure 4 compares the attention score distribution before and after sparse training.

Attention scores after sparse training. Figure 4 compares attention scores from the densely and sparsely fine-tuned models for the same long input and attention layer. Sparse fine-tuning produces a visibly different score pattern, including higher scores in the displayed query-block comparison. This shows that training with the sparse computation path changes the model's attention scores, consistent with the improved performance under sparse inference observed in our ablation. The top row shows the dense model before sparse training, while the bottom row shows the model after sparse training. From left to right, we show the full block-level heatmap, a zoomed-in view of the last 8192 tokens, and the maximum attention score of a representative query block. These visualizations show that sparse fine-tuning changes the attention-score distribution, complementing the task-level evidence of adaptation to sparse computation.