

TTT-Sparse: Test-Time Training of Sparse Attention for Long-Context Understanding

Xuehao Wang^{1,*}, Huayu Deng^{2,*}, Linhao Zhong¹, Yanbin Wei^{4,5},
Hengyu Zhao¹, Mingyu Liu¹, Hao Chen^{1,†}, Chunhua Shen^{1,3,†}

¹State Key Lab of CAD & CG, Zhejiang University, ²Shanghai Jiao Tong University,

³Ant Group, ⁴Southern University of Science and Technology,

⁵Hong Kong University of Science and Technology

Correspondence: xuehaowangfi@gmail.com, haochen.cad@zju.edu.cn, chunhuashen@zju.edu.cn

Abstract

Pretrained LLMs excel across diverse tasks, yet standard attention remains a major bottleneck for long-context understanding. Recent test-time training (TTT) methods alleviate this by encoding context into *internal* model parameters, but they typically require specialized architectures, meta-learned initialization, or continual post-training, preventing plug-and-play use on off-the-shelf LLMs. We propose test-time training sparse attention (**TTT-Sparse**), a plug-and-play alternative that performs test-time training over *external* memory while keeping the internal parameters frozen. For each input, TTT-Sparse constructs contextual chunk key states from the KV cache and learns a lightweight query adapter to induce a task-specific sparse attention pattern, which is used during generation to attend only to the most relevant context chunks and suppress interference from irrelevant context. Extensive experiments on long-context benchmarks show that TTT-Sparse improves long-context understanding while reducing generation-time attention cost.

1 Introduction

Long-context understanding has emerged as a crucial capability of large language models, enabling applications such as multi-turn dialogue (Maharana et al., 2024), reasoning over large code repositories (Jimenez et al., 2024), and long-horizon agentic systems (Chan et al., 2025). However, training LLMs to robustly handle long contexts remains challenging (Hsieh et al., 2024) due to the quadratic cost of standard attention (Vaswani et al., 2017) and the scarcity of high-quality long-context corpora (Zhuang et al., 2025).

Prior work (Behrouz et al., 2025; Sun et al., 2025; Feng et al., 2026) explores test-time training (TTT) to improve long-context understanding

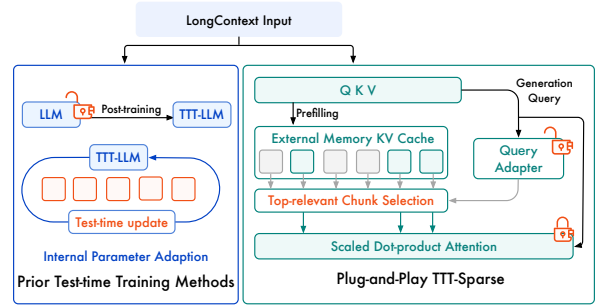


Figure 1: **Comparison between prior TTT methods and TTT-Sparse.** While existing methods adapt the *internal* parameters of LLMs to encode long-context information, TTT-Sparse keeps the off-the-shelf LLM frozen and trains only a lightweight query adapter, which guides top-relevant chunk selection over the *external* memory to select relevant KV states during generation.

by adapting *internal* parameters of LLMs on each input context. However, these methods require pretrained initialization (Behrouz et al., 2025; Sun et al., 2025) or additional continual training (Feng et al., 2026), limiting their plug-and-play applicability to off-the-shelf LLMs.

Different from these methods, we focus on the *external* memory of LLMs, *i.e.*, the prefilled KV cache of the input context. In long-context settings, task-relevant evidence is often sparsely distributed among many irrelevant or redundant cached states. Dense attention over the entire KV cache can therefore introduce distraction, positional bias, and evidence dilution (Jin et al., 2025; Jiang et al., 2024b). While sparse attention offers a natural way to filter irrelevant context, existing methods usually rely on fixed structures (Xiao et al., 2024), heuristic KV-cache pruning (Li et al., 2024), or native sparse patterns (Yuan et al., 2025) over frozen representations. Such static or heuristic patterns may fail to capture the input-specific contextual structure and task-specific selection preferences required for generation.

To enable input-specific KV-cache sparsification without updating the backbone LLM, we pro-

*Equal contribution.

†Correspondence should be addressed to HC and CS.

pose **Test-Time Training Sparse Attention (TTT-Sparse)**, a plug-and-play method that learns sparse attention patterns at test time. We consider a long-context input consisting of a long document segment and a short task-specific observation segment, such as a question or instruction. As shown in Figure 1, the key insight is to use the observation tokens to infer which parts of the *external* memory (*i.e.* prefilled KV cache) are likely to be relevant for subsequent generation. After prefilling, TTT-Sparse partitions the external memory into consecutive chunks and aggregates a **contextual chunk key state** for each chunk. The aggregation weights combine intra-document importance, which captures the document’s internal structure, with observation-conditioned importance, which reflects task-specific preferences. TTT-Sparse then learns a lightweight **query adapter** on the current input to introduce task-specific chunk-selection preference into the query state space. The adapter is optimized with a self-supervised chunk-level target derived from the frozen model’s observation-to-document attention, enabling adapted query states to rank document chunks according to the observation-derived task-specific preference. During generation, adapted queries select the top-relevant document chunks, and the frozen LLM performs standard token-level attention over the selected chunk tokens, the observation tokens, and the generated prefix. Extensive experiments demonstrate that TTT-Sparse improves long-context understanding while reducing generation-time computation cost.

The contributions of this paper are three-fold: *a)* We introduce TTT-Sparse, a new test-time training method for long-context understanding that adapts sparse attention patterns over the KV cache rather than updating internal model parameters. *b)* We propose contextual chunk key states and a lightweight query adapter to capture intra-document contextual structure and task-specific preference for sparse chunk selection. *c)* Extensive experiments on long-context benchmarks demonstrate the effectiveness and efficiency of the proposed TTT-Sparse method.

2 Preliminary

Multi-head attention (MHA). The MHA block in each Transformer layer consists of H attention heads. For simplicity, we describe one head and omit the layer and head indices. Each

head is parameterized by three projection matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d_h}$, where d is the hidden state dimension and d_h is the head dimension. Given the token representation $\mathbf{x}_i \in \mathbb{R}^d$ of the i -th token, the corresponding query, key, and value states of each attention head are calculated as

$$\mathbf{q}_i = \mathbf{x}_i \mathbf{W}_q, \mathbf{k}_i = \mathbf{x}_i \mathbf{W}_k, \mathbf{v}_i = \mathbf{x}_i \mathbf{W}_v. \quad (1)$$

Causal scaled dot-product attention computes the output $\mathbf{o}_i$ as

$$\mathbf{o}_i = \sum_{j \leq i} \frac{z_{i,j}}{\sum_{j' \leq i} z_{i,j'}} \mathbf{v}_j, \quad z_{i,j} = \exp\left(\frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d_h}}\right). \quad (2)$$

The causal constraint $j \leq i$ enforces that each token only attends to itself and the preceding context.

Long-context input formulation. A long-context input with length T typically consists of a document segment and an observation segment. Let $\mathcal{T} = \{1, \dots, T\}$ denote the input positions. We define the document segment $\mathcal{D}$ and observation segment $\mathcal{O}$ as $\mathcal{D} = \{i \in \mathcal{T} \mid s_D \leq i \leq e_D\}$ and $\mathcal{O} = \{i \in \mathcal{T} \mid s_O \leq i \leq e_O\}$, respectively. The document segment provides long-context evidence, while the observation segment specifies the task through instructions, questions, answer options, or response templates. The model is expected to generate the answer based on evidence from the document.

Chunk-level sparse attention. Standard causal attention computes each output over all preceding tokens, incurring quadratic complexity with respect to the context length. To reduce this cost, sparse attention methods restrict each query to a selected subset of historical tokens. In this work, we perform sparse selection at the chunk level over the document segment. Formally, we partition $\mathcal{D}$ into C consecutive chunks indexed by $\mathcal{C} = \{1, \dots, C\}$. For each chunk $c \in \mathcal{C}$, $\mathcal{T}_c = \{s_c, \dots, e_c\} \subseteq \mathcal{D}$ denotes its token positions, with corresponding query, key, and value states $\{\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i\}_{i \in \mathcal{T}_c}$. Let $\mathcal{K}_i \subseteq \mathcal{C}$ denote the selected document chunks for the i -th token. In practice, non-document tokens such as observation tokens and the generated prefix are retained by default, while sparse selection is applied to the document chunks. Thus, the set of attended token positions is defined as $\mathcal{S}_i = \mathcal{R}_i \cup \bigcup_{c \in \mathcal{K}_i} \mathcal{T}_c$, where $\mathcal{R}_i$ denotes the positions of retained non-document tokens. The sparse attention output is

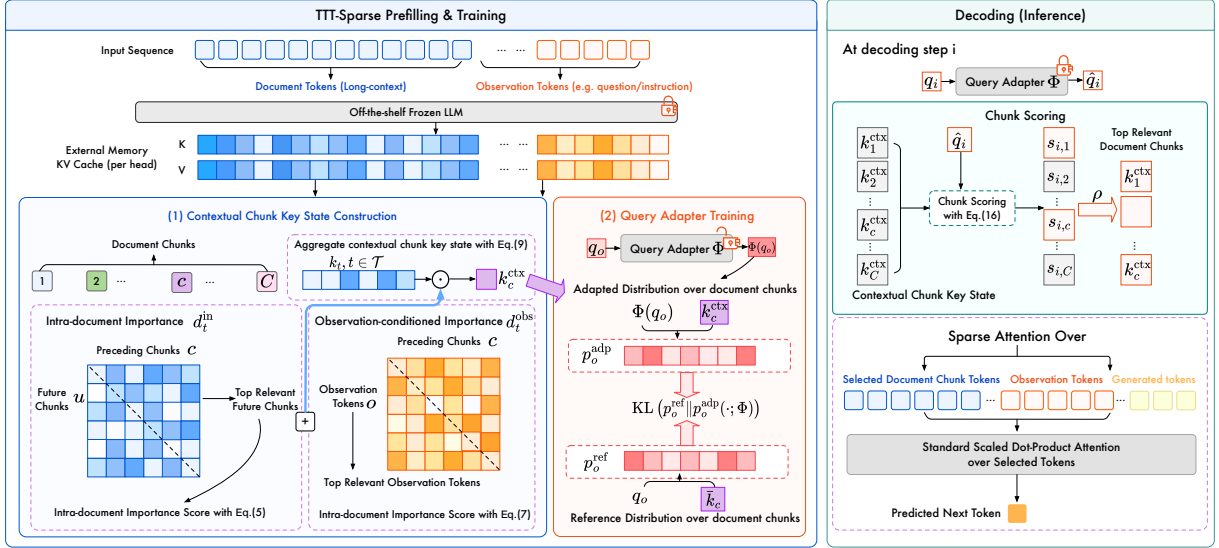


Figure 2: **TTT-Sparse framework.** *Left.* After prefilling, (1) TTT-Sparse partitions the document KV cache into chunks and builds contextual chunk key states for each attention head. (2) It then trains a lightweight query adapter with attention patterns of observation tokens as supervision, adapting query states toward task-specific preferences. *Right.* During generation, adapted queries select relevant chunks, and the model performs standard attention over original KV states of selected chunks together with observation and generated prefix.

formulated as

$$\tilde{o}_i = \sum_{t \in \mathcal{S}_i} \frac{z_{i,t}}{\sum_{t' \in \mathcal{S}_i} z_{i,t'}} v_t, \quad (3)$$

where $z_{i,t} = \exp\left(\frac{q_i k_t^\top}{\sqrt{d_h}}\right)$. When $|\mathcal{S}_i| \ll |\mathcal{D}|$, sparse attention methods substantially reduce the computational costs.

3 Methodology

3.1 Overview

TTT-Sparse learns an input-specific sparse attention pattern over the long-context document segment at test time, while keeping the pretrained LLM frozen. The method consists of two components: *contextual chunk key states* (Sec. 3.2) and a *query adapter* (Sec. 3.3). After prefilling, we represent each document chunk by a contextual chunk key state, computed as a weighted sum of its original token key states using contextual importance scores derived from intra-document importance and observation-conditioned importance. We then optimize a lightweight *query adapter* using the attention distributions of the observation tokens over document chunks as supervision, enabling the adapted query states to capture task-specific chunk preferences. During generation (Sec. 3.4), the adapted query states score document chunks using the contextual chunk key states and select the most relevant subset. Standard attention is then

computed over the original key and value states of the selected chunks, observation tokens, and generated prefix. The overall procedure is outlined in Figure 2.

3.2 Contextual Chunk Key State

To select document chunks efficiently, TTT-Sparse represents each chunk with a contextual chunk key state, which is a weighted sum of the token key states within the corresponding chunk. We compute the weights from two types of contextual importance scores: (i) intra-document importance score, which captures contextual dependencies within the document, (ii) observation-conditioned importance score, which injects task preference from observation tokens such as questions or instructions.

Intra-document importance score. Since attention scores reflect syntactic and contextual relations (Vig and Belinkov, 2019), we estimate the intra-document importance score of each token based on causal attention scores within the document. To avoid the high cost of computing token-level attention between all chunk pairs, we first estimate coarse chunk-level attention scores and then use the subsequent chunks with the largest chunk-level attention scores to compute token-level importance scores for each chunk.

Specifically, we first define the coarse chunk

key state of the c -th chunk as the mean key state $\bar{\mathbf{k}}_c = \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} \mathbf{k}_t$. The chunk query state is formulated as a norm-weighted average of token query states $\bar{\mathbf{q}}_c = \frac{\sum_{j \in \mathcal{T}_c} (\|\mathbf{q}_j\|_2^2 \mathbf{q}_j)}{\sum_{j' \in \mathcal{T}_c} \|\mathbf{q}_{j'}\|_2^2}$. The norm weighting assigns larger weights to tokens whose queries induce larger attention scores and sharper attention distributions (Govindarajan et al., 2026), yielding a chunk query state that better reflects representative and important tokens within the chunk. We then compute the coarse causal importance score from each future chunk u to the preceding chunk $c < u$ as

$$s_{u \rightarrow c} = \frac{\exp(a_{u,c})}{\sum_{c'=1}^{u-1} \exp(a_{u,c'})}, \quad (4)$$

where $c < u$ and $a_{u,c} = \frac{\bar{\mathbf{q}}_u \bar{\mathbf{k}}_c^\top}{\sqrt{d_h}}$. For the c -th chunk, the top- M future chunks with the largest coarse scores, denoted as $\mathcal{M}_c^{\text{in}} = \arg \text{TopM}\{s_{u \rightarrow c}\}_{u > c}$, are selected to calculate the fine-grained intra-document importance score of the token at position $t \in \mathcal{T}_c$ as

$$d_t^{\text{in}} = \sum_{u \in \mathcal{M}_c^{\text{in}}} s_{u \rightarrow c} \frac{\exp(\bar{\mathbf{q}}_u \mathbf{k}_t^\top / \sqrt{d_h})}{\sum_{t' \in \mathcal{T}_c} \exp(\bar{\mathbf{q}}_u \mathbf{k}_{t'}^\top / \sqrt{d_h})}. \quad (5)$$

Observation-conditioned importance score. The intra-document importance score provides a task-agnostic contextual prior by identifying tokens that receive high attention scores from subsequent document chunks. However, this causal signal remains incomplete, as tail chunks receive less supervision from future chunks, and intra-document importance may not align with the evidence required by the current task. Inspired by Li et al. (2024), we therefore introduce an observation-conditioned importance score from the observation tokens to inject task-conditioned signals into sparse attention.

For each observation token at position $o \in \mathcal{O}$, we compute its importance score for the c -th document chunk as

$$s_{o \rightarrow c} = \frac{\exp(a_{o,c})}{\sum_{c'=1}^C \exp(a_{o,c'})}, \quad (6)$$

where $a_{o,c} = \frac{\mathbf{q}_o \bar{\mathbf{k}}_c^\top}{\sqrt{d_h}}$. Similarly, we select the top- M observation tokens $\mathcal{M}_c^{\text{obs}}$ that assign the largest importance score to the c -th chunk and then use these tokens to compute the observation-conditioned im-

portance score of the token at position $t \in \mathcal{T}_c$ as

$$d_t^{\text{obs}} = \sum_{o \in \mathcal{M}_c^{\text{obs}}} s_{o \rightarrow c} \frac{\exp(\mathbf{q}_o \mathbf{k}_t^\top / \sqrt{d_h})}{\sum_{t' \in \mathcal{T}_c} \exp(\mathbf{q}_o \mathbf{k}_{t'}^\top / \sqrt{d_h})}. \quad (7)$$

Contextual chunk key state construction. The intra-document and observation-conditioned importance scores are combined to obtain the final token-level contextual importance scores within each chunk. For each token at position $t \in \mathcal{T}_c$, we compute the contextual importance score d_t as

$$d_t = d_t^{\text{in}} + d_t^{\text{obs}}. \quad (8)$$

Finally, the contextual chunk key state $\mathbf{k}_c^{\text{ctx}}$ of the c -th chunk is computed as a weighted sum of the original key states, which is formulated as

$$\mathbf{k}_c^{\text{ctx}} = \sum_{t \in \mathcal{T}_c} w_t \mathbf{k}_t, \quad (9)$$

where $w_t = \frac{d_t}{\sum_{t' \in \mathcal{T}_c} d_{t'}}$. This aggregation preserves the original key state space while emphasizing tokens supported by both intra-document importance and task-specific preference.

3.3 Query Adapter

After constructing contextual chunk key states, we further learn a lightweight query adapter to introduce the task-specific selection preference into the query state space. As observation tokens commonly include questions or instructions, their attention distributions reveal task-specific preferences over document chunks, thereby providing a natural supervision signal for learning the query adapter. Moreover, instead of imposing this preference as a fixed bias shared by all generation steps, the query adapter applies a residual transformation to the query state. In this way, each generated token preserves its own query-dependent behavior, while its chunk scoring is calibrated toward the sparse selection pattern inferred from observation tokens.

Specifically, the query adapter Φ of each attention head is parameterized by two low-rank matrices $\mathbf{A} \in \mathbb{R}^{d_h \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_h}$, with $r \ll d_h$. Given a query state $\mathbf{q}_t$, the query adapter applies a low-rank residual transformation as

$$\Phi(\mathbf{q}_t) = \mathbf{q}_t + \frac{\alpha}{r} \mathbf{q}_t \mathbf{A} \mathbf{B}, \quad (10)$$

where α controls the strength of the adapter. Accordingly, the resulting attention score between the

Algorithm 1 The training procedure of the query adapter for each layer and attention head.

// Layer/head indices ℓ, h are omitted for clarity

Input: Original KV cache; contextual chunk key states; observation tokens $\mathcal{O}$; rank r ; scale α ; learning rate η ; update steps N .

Output: Query adapter for the current attention head.

- 1: Freeze all pretrained LLM parameters.
- 2: Initialize Φ , with $\mathbf{A}$ randomly initialized and $\mathbf{B} \leftarrow \mathbf{0}$.
- 3: **for** step $n = 1, \dots, N$ **do**
- 4: **for** observation token $o \in \mathcal{O}$ **do**
- 5: Compute reference chunk-level distribution
 $p_o^{\text{ref}}(c) \leftarrow \text{softmax}_c(\mathbf{q}_o \bar{\mathbf{k}}_c^\top / \sqrt{d_h})$.
- 6: Compute adapted chunk-level distribution
 $p_o^{\text{adp}}(c; \Phi) \leftarrow \text{softmax}_c(\Phi(\mathbf{q}_o) \mathbf{k}_c^{\text{ctx}^\top} / \sqrt{d_h})$.
- 7: **end for**
- 8: Compute head-wise KL loss
 $\mathcal{L}_{\text{adp}} \leftarrow \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \text{KL}(p_o^{\text{ref}} \| p_o^{\text{adp}})$.
- 9: Update adapter parameters
 $\mathbf{A}, \mathbf{B} \leftarrow \mathbf{A}, \mathbf{B} - \eta \nabla_{\mathbf{A}, \mathbf{B}} \mathcal{L}_{\text{adp}}$.
- 10: **end for**

adapted query state and the contextual chunk key state can be decomposed as

$$\frac{\Phi(\mathbf{q}_t) \mathbf{k}_c^{\text{ctx}^\top}}{\sqrt{d_h}} = \frac{\mathbf{q}_t \mathbf{k}_c^{\text{ctx}^\top}}{\sqrt{d_h}} + \frac{\alpha}{r} \cdot \frac{\mathbf{q}_t \mathbf{A} \mathbf{B} \mathbf{k}_c^{\text{ctx}^\top}}{\sqrt{d_h}}. \quad (11)$$

The second term is a learned residual score that adjusts the original chunk score in a task-specific and query-dependent manner.

To train the query adapter, we use attention distributions from observation tokens to document chunks as supervision for learning the task-specific chunk preferences in the query state space. For each observation token at position $o \in \mathcal{O}$, we define the reference distribution over document chunks as

$$p_o^{\text{ref}}(c) = \frac{\exp(\mathbf{q}_o \bar{\mathbf{k}}_c^\top / \sqrt{d_h})}{\sum_{c'=1}^C \exp(\mathbf{q}_o \bar{\mathbf{k}}_{c'}^\top / \sqrt{d_h})}, \quad (12)$$

where $c \in \mathcal{C}$ and $\bar{\mathbf{k}}_c = \frac{1}{|\mathcal{T}_c|} \sum_{t' \in \mathcal{T}_c} \mathbf{k}_{t'}$. We then compute the adapted distribution over document chunks using the adapted query state and contextual chunk key states:

$$p_o^{\text{adp}}(c; \Phi) = \frac{\exp(\Phi(\mathbf{q}_o) \mathbf{k}_c^{\text{ctx}^\top} / \sqrt{d_h})}{\sum_{c'=1}^C \exp(\Phi(\mathbf{q}_o) \mathbf{k}_{c'}^{\text{ctx}^\top} / \sqrt{d_h})}. \quad (13)$$

Since minimizing the KL divergence between the two distributions better preserves the relative importance (Kim et al., 2022) and aligns naturally

with chunk selection, the loss function of the query adapter is formulated as

$$\mathcal{L}_{\text{adp}} = \frac{1}{|\mathcal{O}|} \sum_{o \in \mathcal{O}} \text{KL}(p_o^{\text{ref}} \| p_o^{\text{adp}}(\cdot; \Phi)). \quad (14)$$

Algorithm 1 summarizes the overall learning procedure, where each attention head learns an independent adapter while the pretrained LLM remains frozen. During training, only the low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are updated, which keeps the adaptation lightweight. We initialize $\mathbf{B}$ to $\mathbf{0}$, ensuring that the adapter starts from the original model behavior and gradually learns task-specific preference.

3.4 Sparse Generation

After test-time training, each attention head obtains contextual chunk key states $\{\mathbf{k}_c^{\text{ctx}}\}_{c \in \mathcal{C}}$ and a query adapter Φ . During autoregressive generation, these learned components are only used for sparse chunk selection, while the final attention output is calculated over the original full-precision KV cache of the selected chunks, observation tokens, and generated prefix.

For each generation step i , let $\mathbf{q}_i$ denote the current query state. We first apply the learned query adapter to the query state as

$$\hat{\mathbf{q}}_i = \Phi(\mathbf{q}_i). \quad (15)$$

The adapted query is then used to score the c -th document chunk against the corresponding contextual chunk key state as

$$s_{i,c} = \frac{\hat{\mathbf{q}}_i \mathbf{k}_c^{\text{ctx}^\top}}{\sqrt{d_h}}. \quad (16)$$

Given a keep ratio ρ , we select the top-ranked chunks as

$$\mathcal{K}_i = \arg \text{TopK}\{s_{i,c}\}_{c \in \mathcal{C}}, \quad (17)$$

where $K = \lceil \rho |\mathcal{C}| \rceil$. The set of attended token positions at step i is defined as

$$\mathcal{S}_i = \mathcal{R}_i \cup \bigcup_{c \in \mathcal{K}_i} \mathcal{T}_c, \quad (18)$$

where $\mathcal{R}_i$ denotes the positions of non-document tokens, including observation tokens and generated prefix. We gather the original key and value states indexed by $\mathcal{S}_i$ from the original KV cache and compute standard scaled dot-product attention as

$$\tilde{\mathbf{o}}_i = \sum_{t \in \mathcal{S}_i} \frac{\exp(\mathbf{q}_i \mathbf{k}_t^\top / \sqrt{d_h})}{\sum_{t' \in \mathcal{S}_i} \exp(\mathbf{q}_i \mathbf{k}_{t'}^\top / \sqrt{d_h})} \mathbf{v}_t. \quad (19)$$

	Method	Code Repo	Long ICL	Long Struct	Long Dialog	Multi-Doc QA	Single-Doc QA	Avg.
Qwen3-8B	Full Attention	32.00	40.74	30.30	28.21	31.20	34.86	33.80
	Random Sparse	36.00	32.10	18.18	33.33	26.40	30.86	29.80
	SnapKV	34.00	38.27	30.30	20.51	32.00	36.00	33.60
	Quest	38.00	35.80	27.27	23.08	31.20	36.00	33.40
	TTT-Sparse	38.00	40.74	33.33	33.33	31.20	36.00	35.39
Qwen3-14B	Full Attention	40.00	45.68	21.21	20.51	31.20	34.29	34.00
	Random Sparse	28.00	18.52	18.18	28.21	31.20	32.57	28.23
	SnapKV	44.00	45.68	30.30	12.82	34.40	36.00	35.79
	Quest	46.00	48.15	30.30	12.82	32.00	36.57	35.98
	TTT-Sparse	40.00	49.38	30.30	23.08	34.40	37.14	37.18

Table 1: Accuracy (%) on 6 tasks of the LongBench-v2 benchmark. YaRN is applied to Qwen models to support 120K context length. The best results are shown in **bold**.

Notably, inspired by Mao et al. 2026, to prevent positional information from biasing selection, we use pre-RoPE query and key states for test-time training and sparse chunk selection, encouraging the learned sparse pattern to be driven by semantic relevance. However, the final standard scaled dot-product attention is calculated with the post-RoPE query state, the post-RoPE key states from the original KV cache, and the original value states.

4 Experiments

4.1 Experimental Setup

Benchmarks. We empirically evaluate the long-context performance of the proposed method on two long-context benchmark datasets, including *LongBench-v2* (Bai et al., 2025) and *RULER* (Hsieh et al., 2024). The *LongBench-v2* benchmark contains multiple-choice questions with contexts up to 2M words, covering diverse real-world scenarios such as code repository understanding, long in-context learning, single-document QA, multi-document QA, long-dialogue history understanding, and long structured data understanding. *RULER* benchmark provides synthetic long-context evaluation, including needle-in-a-haystack (NIAH), aggregation, and question answering (QA) tasks, which respectively measure evidence retrieval, distributed information aggregation, and question answering over long contexts. Additional details of benchmarks can be found in Appendix B.2.

Baselines. We compare the proposed method TTT-Sparse with (i) full attention, (ii) random sparse, which randomly selects chunks during generation, (iii) SnapKV (Li et al., 2024), which selects the set of important KV for each attention head based on attention patterns from the observation tokens, and (iv) Quest (Tang et al., 2024), which dynamically selects query-relevant chunks

for sparse attention during generation.

Models. We evaluate both baselines and the proposed TTT-Sparse using open-source models from the Qwen3 family (Yang et al., 2025), including Qwen3-8B and Qwen3-14B. For both LongBench-v2 and RULER, we use the same decoding configuration with temperature 0.7, top- p 0.8, and top- k 20. On LongBench-v2, we enable YaRN positional scaling to support contexts up to 120K tokens, while on RULER, we disable YaRN and use the default context configuration of each model.

Implementation Details. For all methods, the keep ratios ρ are set to 0.01 and 0.2 on LongBench-v2 and RULER, respectively. For TTT-Sparse and Quest, we set the chunk size to 512 on LongBench-v2 following the setup of Bai et al. (2025), and use the chunk size as 4 on RULER. The rank r of the query adapter is 32, and the number of update steps is 8. The learning rates η are 0.1 for Qwen3-8B and 0.05 for Qwen3-14B. More implementation details are provided in Appendix B.3.

4.2 Results

The results on the LongBench-v2 benchmark are shown in Table 1. As can be seen, the proposed TTT-Sparse method achieves the best average performance on both Qwen3-8B and Qwen3-14B. Compared with sparse baselines such as Random Sparse, SnapKV, and Quest, TTT-Sparse obtains consistently higher average accuracy, indicating that learning sparse attention patterns at test time is more effective than random or heuristic sparse selection. Notably, TTT-Sparse matches or outperforms full attention on every LongBench-v2 task across both model scales, suggesting that it can reduce irrelevant long-context interference while preserving task-relevant evidence.

Furthermore, to assess whether TTT-Sparse

	Length	Method	NIAH	Aggregation	QA
Qwen3-8B	16k	Full Attention	99.72	<u>82.23</u>	<u>63.00</u>
		Random Sparse	0.03	70.98	37.00
		SnapKV	98.88	78.10	61.50
		Quest	<u>99.84</u>	81.20	62.50
		TTT-Sparse	99.97	87.50	69.50
	32k	Full Attention	99.44	<u>80.37</u>	62.00
		Random Sparse	0.00	67.91	35.00
		SnapKV	97.91	79.35	<u>62.50</u>
		Quest	99.16	80.22	62.00
		TTT-Sparse	<u>99.25</u>	87.69	67.50
Qwen3-14B	16k	Full Attention	100.00	<u>89.40</u>	<u>71.00</u>
		Random Sparse	0.16	81.02	44.50
		SnapKV	99.88	87.94	70.50
		Quest	100.00	86.73	70.50
		TTT-Sparse	100.00	92.30	73.00
	32k	Full Attention	100.00	<u>86.56</u>	67.50
		Random Sparse	0.53	76.84	45.50
		SnapKV	99.50	84.55	68.00
		Quest	100.00	83.10	<u>69.00</u>
		TTT-Sparse	<u>99.84</u>	87.60	69.50

Table 2: Performance on 3 tasks of the RULER across various context lengths. The best results are shown in **bold** and the second-best results are underlined.

Method	Ministral-3-8B	Ministral-3-14B
Full Attention	34.99	38.77
SnapKV	34.79	38.17
Quest	35.19	39.36
TTT-Sparse	37.18	41.15

Table 3: Accuracy (%) of the LongBench-v2 benchmark on the Ministral-3 family.

causes the model to lose fine-grained information within long contexts, we evaluate it on the RULER benchmark at various context lengths. The results are shown in Table 2. As can be seen, TTT-Sparse achieves the best performance on aggregation and QA tasks across both model scales and context lengths, while maintaining near-perfect retrieval accuracy on the NIAH task. These results suggest that TTT-Sparse preserves fine-grained retrieval ability while improving reasoning over long contexts.

4.3 Ablation Study

Extension to Additional Models. To assess whether the proposed method TTT-Sparse generalizes beyond the Qwen3 family, we further evaluate it on *Ministral-3-8B* and *Ministral-3-14B* models. As shown in Table 3, TTT-Sparse consistently outperforms full attention and the sparse baselines at both model scales, supporting its generalization to other model families.

Effectiveness of query adapter and contextual chunk key state. We conduct an ablation study

Method	Qwen3-8B	Qwen3-14B
Full Attention	33.80	34.00
TTT-Sparse w/o Φ and k_c^{ctx}	32.41	35.79
TTT-Sparse w/o Φ	34.00	35.98
TTT-Sparse	35.39	37.18

Table 4: Ablation study on the key components of TTT-Sparse on LongBench-v2 benchmark.

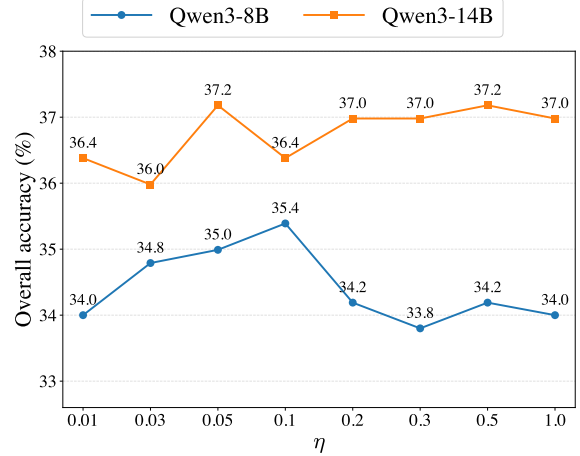


Figure 3: Sensitivity analysis of learning rate η . We report the average accuracy (%) of LongBench-v2 on Qwen3 family using TTT-Sparse across various query adapter learning rates η .

to evaluate the contribution of each component in TTT-Sparse, as shown in Table 4. *TTT-Sparse w/o Φ and k_c^{ctx}* removes both the query adapter and contextual chunk key states, where the original query state directly scores each chunk using the mean chunk key state during generation. *TTT-Sparse w/o Φ* removes only the query adapter and uses the original query state to score the contextual chunk key states for chunk selection. The results show that directly scoring mean chunk key states provides limited and unstable gains, as *TTT-Sparse w/o Φ and k_c^{ctx}* underperforms full attention on Qwen3-8B. *TTT-Sparse w/o Φ* improves performance by introducing contextual chunk key states, indicating that the proposed chunk key state better captures informative tokens within each chunk. Finally, TTT-Sparse achieves the best performance on both models, improving over the contextual chunk key state variant by 1.39% and 1.20% on Qwen3-8B and Qwen3-14B, respectively. These results demonstrate that both contextual chunk key states and the test-time optimized query adapter contribute to effective sparse attention selection.

Impact of query adapter learning rate. We conduct sensitivity analysis to evaluate the impact of the learning rate η used for query adapter op-

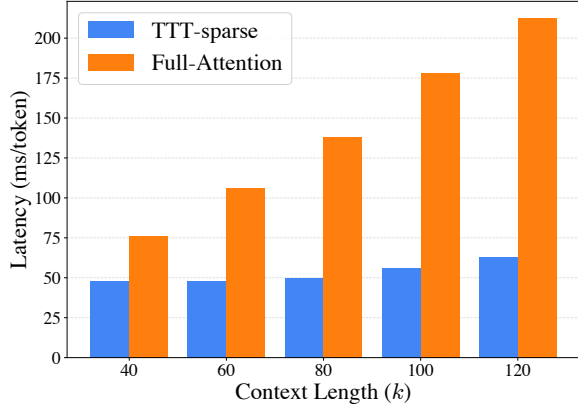


Figure 4: Comparison of generation latency (ms/token) using Qwen3-14B across different context lengths. TTT-Sparse substantially reduces generation latency under long contexts.

Method	Avg.	Method	Avg.
Post-RoPE	32.80	Uniform	33.00
Pre-RoPE	35.39	Norm-weighted	35.39

Table 5: Ablation of using pre-RoPE versus post-RoPE states.

Table 6: Ablation of averaging strategies for constructing chunk query states.

timization. As shown in Figure 3, TTT-Sparse consistently improves over the full attention baseline across a broad range of learning rates on both Qwen3-8B and Qwen3-14B. This indicates that the proposed method is not sensitive to the learning rate. In particular, Qwen3-8B achieves its best performance at a moderate learning rate of 0.1, while Qwen3-14B maintains stable gains under larger learning rates. These results suggest that query adapter training provides robust input-specific adaptation across different model scales.

Wall-clock time comparison. We compare the average generation latency of TTT-Sparse and full attention on Qwen3-14B across different context lengths with the keep ratio $\rho = 0.1$, undergoing 512 generate steps. As shown in Figure 4, the latency of full attention increases rapidly as the context length grows, while TTT-Sparse remains much more stable. This demonstrates that selecting a small set of relevant chunks during generation can substantially reduce the attention computation cost, especially under long-context settings. More wall-clock time comparisons under different hyperparameter settings are provided in Appendix C.

Ablation of using pre-RoPE states. We assess the effectiveness of using pre-RoPE query and key states for test-time training and sparse chunk selection. Since RoPE introduces relative positional in-

Model	Length	Full Attention	TTT-Sparse
Qwen3-8B	4K	72.0	74.5
	8K	68.0	72.0
Qwen3-14B	4K	76.0	77.5
	8K	72.0	73.0

Table 7: Accuracy (%) on short-context QA tasks of the RULER benchmark. The best results are shown in **bold**.

formation into standard attention, using pre-RoPE states reduces its influence on chunk selection. As shown in Table 5, using pre-RoPE states yields better performance than using post-RoPE states, suggesting that separating relevance estimation from positional information benefits chunk selection. Further analysis of the use of pre-RoPE states is provided in Appendix C.

Ablation of averaging methods to construct chunk query states. TTT-Sparse employs norm-weighted averaging to emphasize query states that capture stronger and more selective contextual dependencies. We compare it with the straightforward alternative of uniform averaging. As shown in Table 6, norm-weighted averaging outperforms uniform averaging, supporting its effectiveness in constructing more representative chunk query states.

Performance on short-context QA. To examine whether TTT-Sparse degrades short-context capabilities, we additionally evaluate it on the 4k and 8k RULER QA tasks. This benchmark includes a SQuAD-based (Rajpurkar et al., 2016, 2018) task that emphasizes reading comprehension, and a HotpotQA-based (Yang et al., 2018) task that requires multi-hop integration. As shown in Table 7, TTT-Sparse achieves better performance across various model sizes and context lengths compared to full attention on all tasks. These results suggest that TTT-Sparse does not degrade other capabilities of models, including QA or short-context performance. The detailed results on each QA task are provided in Table 14 of Appendix C.

Due to page limit, additional experiments and analysis are provided in Appendix C.

5 Related Work

Test-Time Training. Test-Time Training (TTT) is a paradigm that enables a model to adapt to each input by optimizing an auxiliary objective at inference time. TTT demonstrates strong effectiveness in computer vision (Sun et al., 2020; Liu et al., 2021; Gandelsman et al., 2022), and recent studies extend this paradigm to language modeling

and long-context understanding (Sun et al., 2025; Tandon et al., 2025; Feng et al., 2026). For long-context understanding, recent TTT methods store contextual information in fast weights or neural memory modules. TTT layers (Sun et al., 2025) treat hidden states as learnable models and update them on the test sequence, while Titans (Behrouz et al., 2025) introduces a memory module that learns to retain past information at test time. TTT-E2E (Tandon et al., 2025) trains model weights on the input context with next-token prediction, LaCT (Zhang et al., 2026) scales TTT with large-chunk fast weight updates, and In-Place TTT (Feng et al., 2026) repurposes MLP weights as inference-time fast weights under a new training objective. Despite their effectiveness, existing TTT methods for long-context LLMs adapt internal states or parameters of models, typically requiring specialized architectures, learned initialization of fast weights, or additional training on external datasets. In contrast, TTT-Sparse performs test-time training over the external KV cache by learning sparse attention patterns for chunk selection, enabling plug-and-play use with frozen pretrained LLMs.

Sparse Attention. Sparse attention methods improve the efficiency of long-context models by restricting attention to a subset of tokens. Early methods train models with predefined sparse patterns, such as sliding-window, global, or random attention (Beltagy et al., 2020; Zaheer et al., 2020). For long-context LLM inference, several methods exploit sparsity in the KV cache. StreamingLLM (Xiao et al., 2024) preserves attention sinks and local windows, H₂O (Zhang et al., 2023) retains heavy-hitter and recent tokens, and SnapKV (Li et al., 2024) compresses the prompt KV cache using attention patterns observed near the end of the prompt. MInference (Jiang et al., 2024a) accelerates prefilling with head-specific sparse patterns, while Quest (Tang et al., 2024) selects query-relevant KV pages during decoding. Different from these static or heuristic sparse attention methods, TTT-Sparse learns an input-specific sparse attention pattern at test time, adapting chunk selection to both document structure and task-specific preferences from observation tokens.

6 Conclusion

In this paper, we propose **TTT-Sparse**, a plug-and-play test-time training method for long-context understanding and generation efficiency that adapts

sparse attention patterns over the external memory (*i.e.*, KV cache) while keeping the pretrained model frozen. TTT-Sparse first constructs contextual chunk key states to represent document chunks based on intra-document importance and observation-conditioned importance. Lightweight query adapters are then optimized to inject task-specific selection preferences from observation tokens into the query state space. During generation, the learned components are used only to select relevant chunks, while the final attention is computed over the original KV states of the selected chunks, the observation tokens, and the generated prefix. Experiments on LongBench-v2 and RULER show that TTT-Sparse improves long-context understanding, preserves fine-grained retrieval, and reduces generation latency. Additional Ministral-3 results and sensitivity analyses support its effectiveness across model families and robustness across the evaluated settings. These results suggest that test-time training over external memory is an effective alternative to updating internal model parameters for long-context understanding.

Limitations

Despite the gains in long-context understanding and generation efficiency, TTT-Sparse preserves the full KV cache, reducing generation-time attention computation but not KV-cache memory. Moreover, supervision derived from the average chunk-level attention distribution of the frozen model may cause the query adapter to inherit ranking errors. Our analysis of selected chunks shows that TTT-Sparse recovers some missed evidence, but correction of all such errors is not guaranteed. The fixed chunk size, learning rate, and adapter scale α may also be suboptimal across different inputs and tasks, motivating future work on adaptive chunking and input-dependent adaptation hyperparameters.

Acknowledgements

AI was used for polishing the paper, such as correcting grammar and enhancing readability, and also provided assistance with partial code implementation. All outputs were reviewed and verified by the authors. This work was supported by the National Natural Science Foundation of China (No. 62576315).

References

- Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. [LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, Vienna, Austria. Association for Computational Linguistics.
- Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. 2025. [Rock-eKV: Accelerating long-context LLM inference via two-stage KV cache compression](#). In *Forty-second International Conference on Machine Learning*.
- Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2025. [Titans: Learning to memorize at test time](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 113506–113543. Curran Associates, Inc.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. [Mle-bench: Evaluating machine learning agents on machine learning engineering](#). In *International Conference on Learning Representations*, volume 2025, pages 50466–50494.
- Keqi Deng, Shaoshi Ling, Ruchao Fan, and Jinyu Li. 2026. [Unique: Universal top-k sparse attention for training-free inference and sparsity-aware training](#). *Preprint*, arXiv:2605.27740.
- Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. 2026. [In-place test-time training](#). In *The Fourteenth International Conference on Learning Representations*.
- Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. 2022. [Test-time training with masked autoencoders](#). In *Advances in Neural Information Processing Systems*.
- Hariprasath Govindarajan, Per Sidén, Jacob Roll, and Fredrik Lindsten. 2026. [QUEST: A robust attention formulation using query-modulated spherical attention](#). In *The Fourteenth International Conference on Learning Representations*.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesht, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. [Ruler: What’s the real context size of your long-context language models?](#) *Preprint*, arXiv:2404.06654.
- 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. 2024a. [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*.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024b. [LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. [SWE-bench: Can language models resolve real-world github issues?](#) In *The Twelfth International Conference on Learning Representations*.
- Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan Arik. 2025. Long-context llms meet rag: Overcoming challenges for long inputs in rag. In *International Conference on Learning Representations*, volume 2025, pages 37784–37822.
- Minsoo Kim, Sihwa Lee, Suk-Jin Hong, Du-Seong Chang, and Jungwook Choi. 2022. [Understanding and improving knowledge distillation for quantization aware training of large transformer encoders](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6713–6725, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. [SnapKV: LLM knows what you are looking for before generation](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Yuejiang Liu, Parth Kothari, Bastien Germain van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. [TTT++: When does self-supervised test-time training fail or thrive?](#) In *Advances in Neural Information Processing Systems*.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870.
- Weian Mao, Xi Lin, Wei Huang, Yuxin Xie, Tianfu Fu, Bohan Zhuang, Song Han, and Yukang Chen. 2026. [Triattention: Efficient long reasoning with trigonometric kv compression](#). *Preprint*, arXiv:2604.04921.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. [Know what you don’t know: Unanswerable questions for SQuAD](#). In *Proceedings of the 56th Annual*

- Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. 2025. [Learning to \(learn at test time\): RNNs with expressive hidden states](#). In *Forty-second International Conference on Machine Learning*.
- Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. [Test-time training with self-supervision for generalization under distribution shifts](#). In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9229–9248. PMLR.
- Arnub Tandon, Karan Dalal, Xinhao Li, Daniel Kocaja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. 2025. [End-to-end test-time training for long context](#). Preprint, arXiv:2512.23675.
- Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. [Quest: query-aware sparsity for efficient long-context llm inference](#). In *Proceedings of the 41st International Conference on Machine Learning, ICML’24*. JMLR.org.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.
- Jesse Vig and Yonatan Belinkov. 2019. [Analyzing the structure of attention in a transformer language model](#). In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. [Efficient streaming language models with attention sinks](#). In *The Twelfth International Conference on Learning Representations*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. [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)*, pages 23078–23097, Vienna, Austria. Association for Computational Linguistics.
- Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. [Big bird: Transformers for longer sequences](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.
- Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T. Freeman, and Hao Tan. 2026. [Test-time training done right](#). In *The Fourteenth International Conference on Learning Representations*.
- Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Re, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. [H2o: Heavy-hitter oracle for efficient generative inference of large language models](#). In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Yonghao Zhuang, Lanxiang Hu, Longfei Yun, Souvik Kundu, Zhengzhong Liu, Eric P. Xing, and Hao Zhang. 2025. [Scaling long context training data by long-distance referrals](#). In *The Thirteenth International Conference on Learning Representations*.

A Algorithm

We present the TTT-Sparse attention module applied to each layer and attention head, together with the overall TTT-Sparse procedure, in Algorithm 2 and Algorithm 3, respectively. For clarity, Algorithm 2 omits layer and head indices. All variables, including the query adapter, contextual chunk key states, and KV cache, correspond to the specific layer and attention head.

B Implementation Details

B.1 Alignment with Grouped Query Attention

The Qwen3 models used in our experiments adopt grouped query attention (GQA), where multiple query heads share one key-value head. Let H_q and H_{kv} denote the numbers of query heads and key-value heads, respectively, and let $\mathcal{G}(g)$ denote the set of query heads associated with the g -th key-value head. In the main text, we describe TTT-Sparse for a single attention head for notational simplicity. In implementation, each query head has its own query state, while several query heads share the same key and value states. Therefore, TTT-Sparse computes query adapters, adapted queries, and chunk scores separately for each query head, but performs sparse selection over the shared KV cache within each GQA group.

Considering the computational overhead of maintaining separate sparse attention patterns for query heads that share the same KV cache, we aggregate chunk scores within each GQA group and perform a single top- K selection for each key-value head. Specifically, let $s_{i,c}^{(h)} \in \mathbb{R}$ denote the score assigned by query head h to the c -th document chunk at generation step i , computed from the adapted query state and the corresponding contextual chunk key state. For the g -th key-value head, we define the group-level chunk score as

$$s_{i,c}^{(kv,g)} = \max_{h \in \mathcal{G}(g)} s_{i,c}^{(h)}. \quad (20)$$

We then select top-relevant chunks using the group-level scores:

$$\mathcal{K}_i^{(kv,g)} = \arg \text{TopK} \{s_{i,c}^{(kv,g)}\}_{c=1}^C, \quad (21)$$

where $K = \lceil \rho C \rceil$. The attended token positions for the g -th key-value head are

$$\mathcal{S}_i^{(kv,g)} = \mathcal{R}_i \cup \bigcup_{c \in \mathcal{K}_i^{(kv,g)}} \mathcal{T}_c. \quad (22)$$

Each query head $h \in \mathcal{G}(g)$ then computes standard attention with its own query state over the original key and value states of the shared KV head indexed by $\mathcal{S}_i^{(kv,g)}$. This aggregation preserves the most relevant chunks for each GQA group while keeping the sparse attention computation aligned with the original GQA implementation.

B.2 Details of benchmarks

LongBench-v2 LongBench-v2 (Bai et al., 2025) is a challenging benchmark for evaluating long-context understanding under heterogeneous input formats. It consists of 503 multiple-choice questions with contexts up to 2M words, covering diverse realistic scenarios such as code repository understanding, long in-context learning, long structured data understanding, long dialogue history understanding, multi-document QA, and single-document QA. Following the LongBench-v2 evaluation protocol, we keep a maximum context length of 120K tokens and truncate longer inputs by removing content from the middle while preserving the beginning and the end of the context. These tasks require models to identify and use relevant evidence scattered across long and noisy contexts. For example, in the code repository task, models need to reason over multi-file project structures, while in multi-document QA, they must integrate information distributed across several related documents. Thus, LongBench-v2 is suitable for evaluating whether TTT-Sparse and baselines can handle diverse long-context inputs.

RULER. RULER (Hsieh et al., 2024) is a synthetic benchmark for evaluating the effective context length of long-context LLMs. It constructs controllable long-context instances by embedding task-relevant evidence among distractor content. In our experiments, we report results on three task groups: Needle-in-a-Haystack (NIAH), aggregation, and question answering (QA). **NIAH** evaluates exact retrieval across single-needle, multi-key, multi-value, and multi-query variants. **Aggregation** consists of common words extraction (cwe) and frequent words extraction (fwe), requiring models to aggregate word-level statistics over the full context. **QA** consists of qa_1 and qa_2, which adapt SQuAD and HotpotQA examples by mixing answer-bearing passages with distractors. These tasks assess precise retrieval, distributed aggregation, and question-conditioned evidence use in long contexts.

Algorithm 2 TTT-Sparse Attention for each attention head.

// Layer/head indices ℓ, h are omitted for clarity

Input: Query state q_i , original KV cache $\{k_t, v_t\}$, contextual chunk key states $\{k_c^{\text{ctx}}\}_{c=1}^C$, query adapter Φ , non-document positions $\mathcal{R}_i$, keep ratio ρ .

Output: Attention output $\tilde{o}_i$.

1: Compute adapted query $\hat{q}_i \leftarrow \Phi(q_i)$.

// Chunk-level sparse selection

2: Compute chunk scores

$$s_{i,c} \leftarrow \hat{q}_i k_c^{\text{ctx}\top} / \sqrt{d_h}.$$

3: Select top-relevant chunks

$$\mathcal{K}_i \leftarrow \arg \text{TopK}\{s_{i,c}\}_{c=1}^C, \text{ where } K = \lceil \rho C \rceil.$$

4: Form attended positions

$$\mathcal{S}_i \leftarrow \mathcal{R}_i \cup \bigcup_{c \in \mathcal{K}_i} \mathcal{T}_c.$$

// Standard attention over selected original KV states

5: Compute attention output

$$\tilde{o}_i \leftarrow \sum_{t \in \mathcal{S}_i} \frac{\exp(q_i k_t^\top / \sqrt{d_h})}{\sum_{t' \in \mathcal{S}_i} \exp(q_i k_{t'}^\top / \sqrt{d_h})} v_t.$$

6: **return** $\tilde{o}_i$.

Benchmark	Model	M	Chunk size	Keep ratio	η
LongBench-v2	Qwen3-8B	64	512	0.01	0.10
	Qwen3-14B				0.05
RULER	Qwen3-8B	32	4	0.20	0.10
	Qwen3-14B				0.05

Table 8: Hyperparameter settings for TTT-Sparse.

B.3 Hyperparameters

Unless otherwise specified, TTT-Sparse uses a low-rank query adapter with update steps $N = 8$, rank $r = 32$, and adapter scale $\alpha = 0.1$. Table 8 reports the detailed hyperparameters, where M is the top- M budget used for constructing contextual chunk key states and ρ is the sparse keep ratio used during chunk selection.

For all ablation studies, we use the default hyperparameter setting of the corresponding benchmark, except when the ablated hyperparameter is explicitly swept for sensitivity analysis. Unless otherwise specified, all wall-clock comparisons are conducted using Qwen3-14B on a single NVIDIA H200 GPU with batch size 1, an 80K context length, 512 generated tokens, a chunk size of 512, a keep ratio of $\rho = 0.1$, 8 query adapter update steps, and the default eager-attention implementation for generation. We report the average over multiple runs. As we do not use hardware-specialized sparse-attention kernels, further speedups may be possible with hardware-aware implementations (Tang et al., 2024; Yuan et al., 2025).

Method	< 32K	32K–128K	> 128K
Full Attention	38.89	32.09	28.70
SnapKV	38.33	31.63	29.63
Quest	38.33	30.70	30.56
TTT-Sparse	41.11	32.56	31.48

Table 9: Detailed accuracy (%) on LongBench-v2 using Qwen3-8B, grouped by original input length before truncation. The best results are shown in **bold**.

C More Experimental Results

Detailed results grouped by original input length. To examine performance across different input lengths, we group all 503 LongBench-v2 examples according to their original token lengths before truncation. As shown in Table 9, TTT-Sparse achieves the highest accuracy in all three groups, indicating that its performance advantage is maintained across various input lengths. All processed inputs are capped at 120K tokens through middle truncation. Therefore, the $> 128K$ group refers to original input length rather than evaluation with more than 128K processed tokens.

Impact of chunk size. We explore the effects of chunk size for TTT-Sparse on LongBench-v2. As shown in Figure 5, TTT-Sparse outperforms the full attention and sparse baselines across different chunk sizes. This demonstrates the effectiveness of TTT-Sparse and suggests that the method is reasonably robust to the choice of chunk size.

Additional wall-clock time comparisons. We evaluate the wall-clock time of TTT-Sparse on

Algorithm 3 Overall Procedure of TTT-Sparse.

Input: Long-context input with document segment $\mathcal{D}$ and observation segment $\mathcal{O}$, keep ratio ρ , top- M budget M .

Output: Generated tokens.

```
// Test-time preparation
1: Prefill the input and obtain the original KV cache, query/key states, and document chunks  $\{\mathcal{T}_c\}_{c=1}^C$ .
2: for each layer  $\ell$  and attention head  $h$  do
3:   Construct contextual chunk key states  $\{\mathbf{k}_c^{\text{ctx},\ell,h}\}_{c=1}^C$  with Eq. 9.
4:   Optimize query adapter  $\Phi_{\ell,h}$  using Algorithm 1.
5: end for

// Autoregressive generation with TTT-Sparse Attention
6: for generation step  $i = 1, 2, \dots$  do
7:   Embed the current token and obtain hidden state  $\mathbf{h}_i^0$ .
8:   for layer  $\ell = 1, \dots, L$  do
9:     Compute query/key/value states from  $\mathbf{h}_i^{\ell-1}$ .
10:    for attention head  $h = 1, \dots, H$  do
11:      Execute TTT-Sparse Attention in Algorithm 2.
      // Using  $\Phi_{\ell,h}$ ,  $\{\mathbf{k}_c^{\text{ctx},\ell,h}\}$ , and the original KV cache of head  $h$ 
12:    end for
13:    Merge attention-head outputs.
14:    Apply the remaining frozen layer operations.
15:  end for
16:  Compute output logits and generate the next token.
17:  Update the original KV cache following standard autoregressive decoding.
18: end for
```

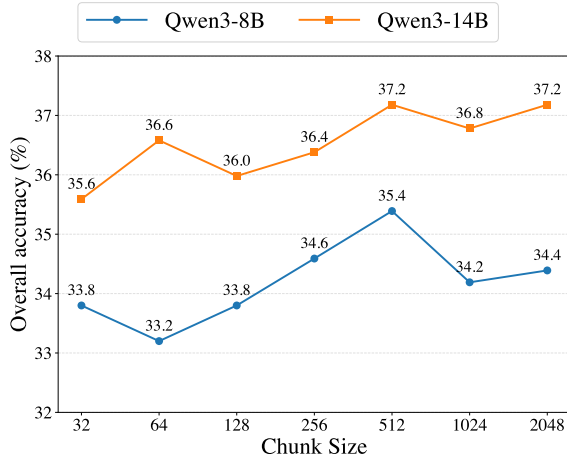


Figure 5: **Sensitivity analysis of chunk size.** Average accuracy (%) of LongBench-v2 on Qwen3 family using TTT-Sparse across various chunk sizes.

Qwen3-14B under different sparse keep ratios. As shown in Figure 6, TTT-Sparse consistently achieves lower generation latency than full attention with 80K context length. The latency remains nearly unchanged when $\rho \leq 0.1$ and increases as more chunks are retained. Even at $\rho = 0.5$, TTT-Sparse remains faster than full attention, indicating

that sparse chunk selection reduces generation-time attention cost across different keep ratios.

We further measure end-to-end (E2E) wall-clock time, where Full E2E includes prefill and full attention generation, while TTT E2E includes prefill, test-time training, and sparse generation. Tables 10, 11, 12, and 13 report end-to-end wall-clock time across different prefix lengths, generation lengths, update steps, and chunk sizes, respectively. The results show that the test-time training overhead remains small relative to the generation cost. As the prefix or generation length increases, TTT-Sparse achieves larger speedups because full attention generation cost grows more rapidly with context length, while the relative impact of the fixed test-time training overhead diminishes as more tokens are generated. In contrast, varying the number of update steps or the chunk size causes only modest changes in end-to-end wall-clock time, indicating consistent efficiency gains across the evaluated optimization budgets and chunk granularities. Overall, these results demonstrate that TTT-Sparse provides robust end-to-end efficiency gains across different hyperparameter settings.

Prefix Length	Prefill (s)	TTT (s)	Full gen. (s)	Sparse gen. (s)	Full E2E (s)	TTT E2E (s)	Speedup
40K	3.21	0.52	38.48	23.78	41.69	27.51	1.52×
60K	5.50	0.60	53.88	23.55	59.38	29.65	2.00×
80K	8.51	0.70	70.38	25.16	78.89	34.37	2.30×
100K	12.02	0.81	91.68	28.89	103.70	41.72	2.49×
120K	16.57	0.90	108.74	32.05	125.31	49.52	2.53×

Table 10: Comparison of end-to-end wall-clock time (s) using Qwen3-14B across different prefix lengths. TTT-Sparse achieves increasing speedups as the prefix length grows.

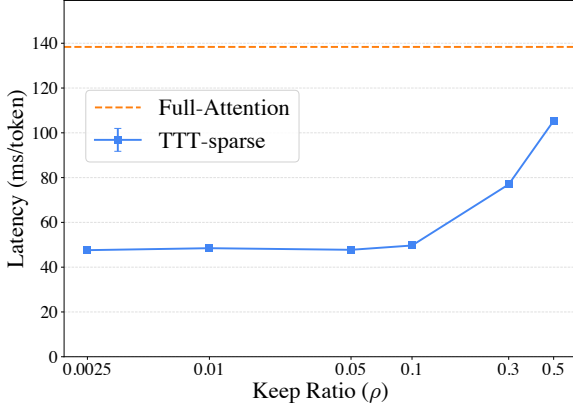


Figure 6: Comparison of generation latency (ms/token) of Qwen3-14B across different keep ratios with 80K context length.

Generated Tokens	Full E2E (s)	TTT E2E (s)	Speedup
128	26.31	15.76	1.67×
256	44.02	22.10	1.99×
512	78.89	34.37	2.30×
1024	150.74	60.44	2.49×
2048	294.90	112.93	2.61×

Table 11: Comparison of end-to-end wall-clock time (s) using Qwen3-14B across different generation lengths. TTT-Sparse achieves larger speedups for longer generations.

Detailed results on short-context QA tasks. Table 14 provides the detailed results on short-context QA tasks. As can be seen, TTT-Sparse outperforms full attention on both the SQuAD-based and HotpotQA-based QA tasks across the evaluated model sizes and context lengths. These results indicate that the improvements are consistent across reading comprehension and multi-hop question answering, rather than being driven by a single task.

Impact of query adapter scale α . We evaluate the robustness of the proposed method to α through a sensitivity analysis on LongBench-v2 of Qwen3-8B with fixed hyperparameters. As shown in Table 15, TTT-Sparse outperforms the full attention baseline across various α , which demonstrates the robustness of the proposed method to α . However,

Update Steps	TTT (s)	TTT E2E (s)	Speedup
1	0.39	33.98	2.32×
2	0.44	34.23	2.30×
4	0.53	34.15	2.31×
8	0.70	34.37	2.30×
16	1.04	34.88	2.26×
32	1.80	35.40	2.22×

Table 12: Comparison of end-to-end wall-clock time (s) using Qwen3-14B across different numbers of update steps. TTT-Sparse maintains consistent speedups across various update steps.

Chunk Size	TTT E2E (s)	Speedup
32	33.21	2.38×
64	33.48	2.36×
128	32.84	2.40×
256	33.31	2.37×
512	33.95	2.32×
1024	33.34	2.37×

Table 13: Comparison of end-to-end wall-clock time (s) using Qwen3-14B across different chunk sizes with $\rho = 0.01$.

both excessively large and excessively small values underperform $\alpha = 0.1$, suggesting that sample-level adaptive α may offer further improvements. We leave this extension for future work.

Analysis of selected chunks. To evaluate whether each method can accurately locate task-relevant evidence, we measure Hit@K on RULER NIAH prompts with 32K context length. For each prompt, selection scores are normalized within each layer and KV head and then averaged to obtain a unified chunk ranking. Hit@K is defined as the fraction of prompts for which at least one gold-evidence chunk is contained within the top-K selected chunks. As shown in Table 16, the selected chunks of TTT-Sparse contain the gold evidence for every prompt, whereas the baselines only reach comparable Hit@K at larger K. This shows that

Model	Length	SQuAD		HotpotQA		Mean	
		Full Attention	TTT-Sparse	Full Attention	TTT-Sparse	Full Attention	TTT-Sparse
Qwen3-8B	4K	86.0	88.0	58.0	61.0	72.0	74.5
	8K	80.0	86.0	56.0	58.0	68.0	72.0
Qwen3-14B	4K	88.0	89.0	64.0	66.0	76.0	77.5
	8K	85.0	86.0	59.0	60.0	72.0	73.0

Table 14: Detailed accuracy (%) on the SQuAD-based and HotpotQA-based RULER QA tasks at 4K and 8K context lengths. The better result between full attention and TTT-Sparse is shown in **bold**.

α	0.01	0.05	0.10	0.20	0.50
Avg.	34.39	34.99	35.39	34.79	34.19

Table 15: Sensitivity analysis of the query adapter scale α on LongBench-v2 using Qwen3-8B. The best result is shown in **bold**.

Selector	Hit@1	Hit@2	Hit@4	Hit@8	Hit@16
Frozen attention	0.017	0.100	0.500	0.883	1.000
Random	0.000	0.000	0.000	0.000	0.017
Quest	0.667	0.800	0.933	0.983	1.000
SnapKV	0.333	0.800	1.000	1.000	1.000
TTT-Sparse	1.000	1.000	1.000	1.000	1.000

Table 16: Comparison of Hit@K for locating gold-evidence chunks across different chunk selection methods on RULER NIAH prompts with 32K context length. The best results are shown in **bold**.

TTT-Sparse locates the gold evidence substantially more accurately than existing baselines, supporting the effectiveness of the proposed method.

Furthermore, we measure the miss rate of the original attention pattern and TTT-Sparse at the actual keep ratio budget. We find that the original attention pattern misses the gold-evidence chunk in only 1.8%, while TTT-Sparse recovers 51.4% of the gold chunks the original attention pattern misses, i.e., the proposed method corrects the frozen model’s relevance estimation rather than inheriting it. This gain comes from the contextual chunk key states: replacing uniform averaging of token key states with an aggregation weighted by intra-document and observation-conditioned importance scores helps recover gold-evidence chunks missed by the original attention pattern.

Analysis of using pre-RoPE states. Since RoPE may incorporate relative positional information into standard attention, post-RoPE chunk scores may entangle semantic relevance with positional preference. To quantify this effect, we evaluate on the RULER NIAH prompts with 32K context length in which the evidence is randomly distributed across the document and measure the

Method	Mean selected position
Post-RoPE	0.642
Pre-RoPE	0.608

Table 17: Mean positions of selected chunks using pre-RoPE and post-RoPE selection on RULER NIAH prompts with 32K context length.

Method	Avg.
RocketKV	33.80
UNIQUE	33.40
TTT-Sparse	35.39

Table 18: Accuracy (%) on LongBench-v2 using Qwen3-8B. The best result is shown in **bold**.

normalized mean position of the selected chunks, where 0 and 1 denote the beginning and end of the document, respectively, and 0.5 indicates no net early/late preference. As shown in Table 17, using pre-RoPE states reduces the mean selected position from 0.642 to 0.608, moving it closer to the position-neutral value of 0.5 and indicating a weaker late-position preference. This result demonstrates the effectiveness of pre-RoPE states in mitigating positional preference during chunk selection while improving downstream performance.

Comparison with additional sparse baselines. We compare with two additional sparse baselines (i.e., RocketKV (Behnam et al., 2025) and UNIQUE (Deng et al., 2026)) on LongBench-v2 of Qwen3-8B. As shown in Table 18, TTT-Sparse achieves better performance than these baselines.