SELF-GUIDED TEST-TIME TRAINING FOR LONG-CONTEXT LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context processing has become increasingly important for large language models (LLMs), but simply extending the context window does not guarantee effective utilization of long inputs. As input length grows, accuracy often degrades, indicating that models still struggle to identify and use the evidence most relevant to a question. A promising way to improve long-context utilization is test-time training (TTT), which treats the test context as a training example for instance-specific parameter adaptation. However, applying TTT to the entire long context is prohibitively expensive, while adapting on randomly sampled spans introduces severe noise. Because most spans in a long context are irrelevant to the specific question, training on them may even degrade the base model's performance. Our preliminary study shows that TTT is highly sensitive to training-span quality: on LongBench-v2, TTT on randomly sampled spans hurts performance, whereas TTT on oracle spans substantially improves it. Motivated by this, we propose a simple method, Self-Guided TTT (S-TTT): before adaptation, the model identifies the evidence spans it should learn from, and the standard language-modeling training objective is applied only to those selected spans. On two challenging long-context reasoning benchmarks, LongBench-v2 and LongBench-Pro, S-TTT improves accuracy for both Qwen3-4B-Thinking-2507 and Llama-3.1-8B-Instruct, achieving up to a 15% relative improvement.

1 INTRODUCTION

Long-context capability has become a central requirement for modern language models. Recent models support context windows of hundreds of thousands of tokens, enabling them to process long inputs in a single prompt (Peng et al., 2024; Chen et al., 2024). Despite this progress, a larger window does not by itself ensure that the model can use long inputs effectively. As context length grows, accuracy often degrades, and models struggle to keep the most relevant evidence accessible throughout reasoning and decoding (Liu et al., 2024; Hsieh et al., 2024). This suggests that the bottleneck in long-context reasoning is not merely fitting more tokens into the prompt, but ensuring that the model can identify and use the evidence relevant to the question. Test-time training (TTT) (Sun et al., 2020; Liu et al., 2021; Hardt & Sun, 2024; Akyürek et al., 2024; Tandon et al., 2025; Zhang et al., 2025a; Feng et al., 2026) has emerged as a promising solution. Instead of answering with a fixed model, TTT treats the test input itself as a training example, adapts the model weights for that specific instance, and uses the adapted weights to generate the answer. For long-context tasks, this is especially appealing because adaptation can turn instance-specific evidence in the context into parameter updates, making it easier to use during subsequent generation (Bansal et al., 2026; Chen et al., 2026a).

However, a key challenge in applying TTT to long contexts is determining *what* data to train on—an important dimension that remains largely underexplored. Existing approaches commonly rely on either full-context adaptation (Tandon et al., 2025; Zhang et al., 2025a) or randomly sampled training spans(Bansal et al., 2026), both of which suffer from noisy signals. Not only is performing TTT on the full context computationally expensive, but it also overwhelms the adaptation process with distractors, as the vast majority of a long context is usually irrelevant to the specific query. A cheaper alternative is to train on randomly sampled spans. While this mitigates the computational cost, it may amplify the noise: random sampling frequently misses the relevant evidence, causing the model to adapt primarily on distractors.

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.

This suggests that the central bottleneck of long-context TTT is not the adaptation mechanism itself, but rather *test-time training-data quality*. We empirically demonstrate this sensitivity through a preliminary diagnostic: on LongBench-v2 (Bai et al., 2025), TTT on random spans slightly degrades performance relative to standard base model inference, whereas training on answer-aware oracle spans annotated by GPT-5.5 yields substantial improvements. This demonstrates that the effectiveness of TTT depends critically on the signal-to-noise ratio of the training tokens.

Motivated by this insight, we propose a simple solution, Self-Guided TTT (S-TTT). Rather than processing the entire context or sampling spans blindly, S-TTT leverages the LLM itself as a test-time data selector. We prompt the model to mark verbatim spans in the context that are likely to support the question. We then adapt the model on the selected spans with a next-token-prediction objective and generate the final answer from the full context. As such, S-TTT leaves the training objective, model architecture, and final decoding procedure unchanged; it optimizes only the test-time tokens used for adaptation. On two challenging long-context reasoning benchmarks, LongBench-v2 (Bai et al., 2025) and LongBench-Pro (Chen et al., 2026b), using Qwen3-4B-Thinking-2507 (Qwen Team, 2025) and Llama-3.1-8B-Instruct (Team, 2024) models, S-TTT consistently improves long-context performance and outperforms strong TTT baselines.

Our contributions are:

1. We identify *training-data quality* as a critical yet underexplored bottleneck for long-context TTT. We empirically demonstrate that adapting on noisy context can degrade performance, whereas high-quality evidence spans lead to substantial gains.
2. We propose Self-Guided TTT (S-TTT), a simple and effective framework that uses the LLM itself to select question-relevant evidence spans for test-time training, avoiding the expensive computational cost of full-context training and mitigating the severe noise of random span sampling.
3. We evaluate S-TTT on two challenging long-context reasoning benchmarks LongBench-v2 and LongBench-Pro using Qwen3-4B-Thinking and Llama-3.1-8B-Instruct models. S-TTT consistently improves long-context performance and outperforms various strong TTT baselines.

2 METHOD

2.1 PRELIMINARY ANALYSIS

Test-time training has been used to improve LLMs long-context performance by adapting the model to the specific context observed at test time. For long inputs, however, directly training on the full context is expensive, and a naive alternative is to train on short spans sampled uniformly from the context. This reduces the compute but has a cost: in a long document, most uniformly sampled spans are irrelevant to the question. As a result, TTT on randomly sampled spans may adapt the model to distractors rather than evidence.

Table 1: Test-time training is sensitive to training-token quality. Training Qwen3-4B-Thinking-2507 on random span tokens does not lead to improvement; instead, it hurts performance.

Method	**LongBench-v2**
Base Model	40.4
Random Span TTT	38.9
Oracle Span TTT	**45.9**

Table 1 shows a diagnostic experiment on LongBench-v2. The base model Qwen3-4B-Thinking-2507 reaches 40.4% accuracy without fine-tuning. After adapting the model via TTT on uniformly sampled spans, accuracy drops to 38.9%, indicating that TTT does not guarantee an improvement when the training tokens are noisy. In contrast, when the training spans are oracle spans annotated by GPT-5.5 with access to the ground-truth answer, the same TTT procedure achieves 45.9% accuracy. Notably, we explicitly control the length of oracle spans to be comparable to that of the random spans, therefore, the number of training tokens is not the factor affecting the performance. This gap isolates

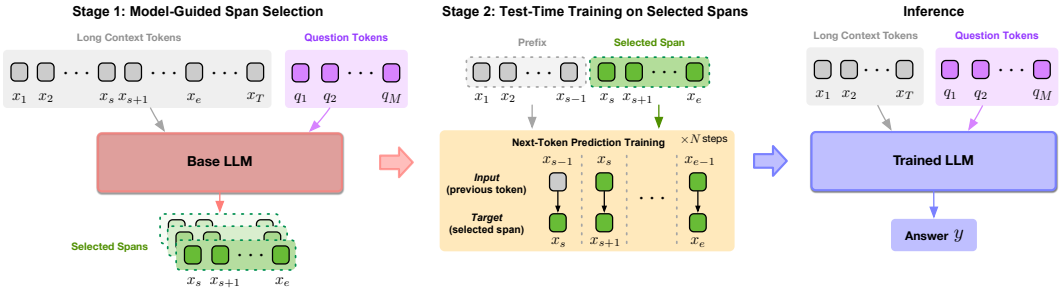

Figure 1: Overview of Self-Guided TTT. In Stage 1, the base LLM reads the long context and question, and identifies question-relevant spans from the context. In Stage 2, these selected spans are used for TTT. At inference time, the adapted model generates the answer conditioning on the original full context and the question.

the role of the training data quality: TTT can help, but only when the tokens used contain useful evidence.

This motivates our core view: the central bottleneck in long-context TTT is not only how to adapt the model, but also what to adapt on. High-quality spans provide a much stronger training signal, however, relying on an external oracle is not a practical solution. We therefore ask whether the model can identify the effective test-time training tokens by itself.

2.2 Self-Guided TTT

The preliminary analysis suggests that test-time training is effective only when the training tokens provide useful information for the current test instance. Based on this observation, we introduce Self-Guided TTT (S-TTT), in which the model first identifies question-relevant evidence from the context and then adapts itself on the selected evidence. Specifically, given a context $x = (x_1, \ldots, x_T)$, a question q, and a base model with parameters θ, S-TTT consists of two stages:

Stage 1: Model-guided span selection. We first ask the model to identify the parts of the context that are most relevant to answering q. Concretely, the model reads the full context and question and returns a set of verbatim supporting spans,

$$\mathcal{S}(x, q) = \{x_{s_j:e_j}\}_{j=1}^{M},$$

where each interval $[s_j, e_j]$ corresponds to a contiguous span copied from the original context. This selection step relies on the model's own relevance judgment to construct instance-specific training data. The purpose of this stage is not to replace the original context at generation time, but to identify the subset of tokens that provides the most useful adaptation signal, which may otherwise be buried among a large amount of irrelevant information in the full context.

Stage 2: Test-time training on selected spans. Starting from a fresh copy of the base model $\theta' \leftarrow \theta$, we perform next-token prediction on the selected spans. For a selected span $x_{s_j:e_j}$, the training objective is

$$\mathcal{L}_{\mathrm{TTT}}(\theta') = -\sum_{i=s_j}^{e_j} \log p_{\theta'}(x_i \mid x_{<i}).$$

Across adaptation steps, we cycle through the valid spans in $\mathcal{S}(x, q)$ and update θ' using the training objective above. The model is encouraged to internalize information that is likely to be useful for answering the current question identified by itself, rather than arbitrary content from the long context.

After adaptation, the updated model generates the answer conditioned on the original full context and question:

$$y \sim p_{\theta'}(\cdot \mid x_{1:T}, q).$$

The full context remains available during generation, so span selection determines only the test-time training data and does not remove potentially useful information from the final input. Once the instance is completed, θ' is discarded and the next instance begins from the original parameters θ. A per-instance loop is described in Algorithm 1.

Algorithm 1 Self-Guided TTT

Require: Model θ, context $x_{1:T}$, question q, steps N, span length k, learning rate η

Initialize a fresh model θ' from θ
$\mathcal{S} \leftarrow$ spans in $x_{1:T}$ annotated by θ relevant to q
if $\mathcal{S} = \emptyset$ **then**
    $\mathcal{S} \leftarrow$ random spans sampled from $x_{1:T}$
end if
for $n = 1, \ldots, N$ **do**
    Choose span $x_{s_j:e_j}$ from $\mathcal{S}$
    $\mathcal{L}_{\text{TTT}} \leftarrow -\sum_{i=s_j}^{e_j} \log p_{\theta'}(x_i \mid x_{<i})$
    Update $\theta' \leftarrow \theta' - \eta \nabla \mathcal{L}_{\text{TTT}}$
end for
return answer $y \sim p_{\theta'}(\cdot \mid x_{1:T}, q)$

3 Experimental Results

3.1 Setup

Models and benchmarks. We evaluate two base models: **Qwen3-4B-Thinking-2507** (Qwen Team, 2025) and **Llama-3.1-8B-Instruct** (Team, 2024). We conduct experiments on two challenging long-context benchmarks. **LongBench-v2** (Bai et al., 2025) is a four-way multiple-choice benchmark covering diverse long-context reasoning tasks and is evaluated using answer accuracy. **LongBench-Pro** (Chen et al., 2026b) evaluates a broader set of long-context capabilities and has English and Chinese subsets, we use its English subset as our evaluation set and apply its official evaluation pipeline for scoring. We use the Qwen3 tokenizer to measure context length and keep examples whose contexts contain at most 128k tokens.

Compared methods. We compare the following settings:

- **Base Model.** The base model directly generates an answer conditioned on the full context and question, without any parameter updates.
- **LongLLMLingua.** LongLLMLingua (Jiang et al., 2024b) is a prompt compression method. The base model first compresses the full context into a shorter one conditioned on the question, and then answers the question using the compressed context. We set the compressed-context budget to be 4,096 tokens.
- **qTTT.** qTTT (Bansal et al., 2026) is an efficiency-oriented TTT method that adapts on uniformly sampled random spans. It first runs a single forward pass over the full context to build the KV cache, then keeps the cache frozen and updates only the query-projection parameters, avoiding recomputation of the full-context KV at every adaptation step.
- **QRHead Span TTT.** Following QRHead (Zhang et al., 2025b), we identify query-relevant attention heads on a retrieval set BEIR (Thakur et al., 2021) by scoring each head's query-to-context attention as a retriever and the 16 highest-scoring heads are kept as QRHeads. At test time, we run a single forward pass over the full context to obtain QRHead attention scores, aggregate them over each 512-token candidate span to obtain span-level scores, and select the 8 highest-scoring spans for TTT.
- **Random Span TTT.** We randomly sample 8 spans from the context, each containing 512 tokens, and use them for TTT. Random Span TTT differs from qTTT in updating with a non-frozen KV cache.
- **Full Context TTT.** We partition the full context into N contiguous chunks, where N is the number of adaptation steps. At each step, we perform one step TTT update on one chunk.
- **Self-Guided TTT (ours).** We ask the model to identify at most 8 spans in the context that are relevant to the question and then perform TTT on the selected spans. If the model fails to output valid spans, it falls back to using uniformly sampled spans. We report fallback rates in Appendix B.

Table 2: Results on LongBench-v2 and LongBench-Pro across different context-length buckets. All numbers are the mean over three independent runs, with standard deviations as gray subscripts. The best result for each model and evaluation setting is shown in **bold**. †QRHead Span TTT is not directly comparable with the other TTT methods as it requires additional information to identify retrieval heads.

Model	Method	LongBench-v2		LongBench-Pro	
		$<$ 64k	64k–128k	$<$ 64k	64k–128k
Qwen3-4B-Thinking-2507	Base Model	46.7 $_{0.5}$	30.7 $_{0.7}$	55.1 $_{0.4}$	41.6 $_{0.9}$
	LongLLMLingua	41.8 $_{0.7}$	31.7 $_{1.1}$	35.0 $_{1.3}$	30.3 $_{1.8}$
	qTTT	44.7 $_{1.9}$	34.0 $_{2.6}$	56.6 $_{1.2}$	41.5 $_{1.5}$
	QRHead Span TTT†	47.2 $_{2.1}$	32.1 $_{0.2}$	**56.7** $_{1.1}$	40.8 $_{1.7}$
	Random Span TTT	43.6 $_{0.6}$	34.2 $_{0.4}$	55.0 $_{1.9}$	41.0 $_{0.1}$
	Full Context TTT	45.1 $_{0.9}$	32.6 $_{1.9}$	55.8 $_{1.0}$	40.4 $_{1.8}$
	Self-Guided TTT	**47.7** $_{1.4}$	**35.3** $_{1.2}$	56.2 $_{0.2}$	**42.0** $_{1.5}$
Llama-3.1-8B-Instruct	Base Model	36.9 $_{0.2}$	26.3 $_{0.6}$	28.2 $_{0.9}$	19.4 $_{1.1}$
	LongLLMLingua	34.1 $_{0.5}$	26.9 $_{0.4}$	25.2 $_{1.6}$	21.0 $_{1.2}$
	qTTT	35.7 $_{0.5}$	27.5 $_{0.2}$	29.7 $_{1.1}$	19.3 $_{0.3}$
	QRHead Span TTT†	35.7 $_{1.0}$	27.5 $_{0.4}$	29.4 $_{1.4}$	20.4 $_{0.5}$
	Random Span TTT	36.0 $_{2.1}$	26.7 $_{0.8}$	28.7 $_{1.0}$	20.4 $_{1.1}$
	Full Context TTT	35.2 $_{1.4}$	27.7 $_{1.7}$	29.4 $_{0.5}$	19.8 $_{0.3}$
	Self-Guided TTT	**38.4** $_{1.0}$	**28.2** $_{0.8}$	**29.9** $_{1.4}$	**21.7** $_{1.2}$

For all TTT methods, the final answer is generated conditioned on the full context rather than the selected spans. We use LoRA for parameter-efficient test-time adaptation and perform 16 gradient-update steps for each test instance. More details can be found in Appendix A.

Evaluation. For each test instance, we sample 4 responses and evaluate each response using the corresponding benchmark evaluator. We report the mean scores for the four samples. We sample responses with a temperature of 0.6 and a top-p of 0.95. The maximum generation length is set to 32,768 tokens for Qwen3-4B-Thinking-2507 and 10,240 tokens for Llama-3.1-8B-Instruct. Prompt templates can be found in Appendix C.

3.2 Results

Table 2 summarizes the main results. We highlight three observations. First, S-TTT consistently improves over the base model across models, benchmarks, and length buckets. Second, S-TTT consistently outperforms or is comparable to all the other TTT methods, showing that model-annotated training spans provide a more reliable adaptation signal than uniformly sampled spans or full context. Third, the gains are especially pronounced in the longer-context buckets, where irrelevant context is more abundant and training-token selection becomes more important.

LongBench-v2. Using Qwen3-4B-Thinking-2507 as the base model, Random Span TTT degrades the $<$ 64k bucket, reducing accuracy from 46.7 to 43.6, whereas S-TTT improves it to 47.7. In the 64k–128k bucket, all TTT baselines help, but S-TTT leads to the strongest score, reaching 35.3. QRHead Span TTT is competitive in the shorter bucket, but drops behind in the longer bucket, suggesting that attention-based span scores are less stable as the context grows. Other baselines are less consistent: LongLLMLingua often underperforms the base model, and Full Context TTT remains below S-TTT in every setting.

The trend also transfers to Llama-3.1-8B-Instruct. S-TTT gives the best LongBench-v2 scores in both buckets, improving the base model from 36.9 to 38.4 and from 26.3 to 28.2, respectively. This indicates that the gain from S-TTT is model-agnostic.

LongBench-Pro. With Qwen3-4B-Thinking-2507, S-TTT improves over the base model in both length buckets. In the shorter bucket, qTTT and QRHead Span TTT are slightly higher than S-TTT.

In the longer bucket, however, S-TTT is the strongest method, reaching 42.0 and outperforming all TTT baselines. This mirrors the LongBench-v2 trend: model-annotated spans become more valuable when the context is longer and noisier.

With Llama-3.1-8B-Instruct, S-TTT again gives the best LongBench-Pro scores in both buckets, improving the base model from 28.2 to 29.9 and from 19.4 to 21.7, respectively. Overall, these results support our main hypothesis that training-token quality is a central bottleneck for long-context TTT.

4 Analysis

We further analyze why and when S-TTT works. We ask three questions: (1) whether question-conditioned model annotation is more effective than annotation-free intrinsic span scores, (2) how adaptation on selected spans changes the model's attention to the relevant evidence, and (3) how the end-to-end overhead of S-TTT scales with context length.

4.1 Span selection strategies

The main results show that span selection matters. We next ask whether explicit model annotation is necessary, or whether simpler annotation-free signals can select useful training spans. A natural alternative is to use intrinsic model statistics, selecting spans that are difficult to predict or induce high uncertainty in the next-token distribution.

We compare against two intrinsic selectors. The perplexity selector ranks each 512-token window by mean negative log-likelihood, while the entropy selector ranks each window by mean predictive entropy. Each method then performs TTT on the top 8 highest-scoring spans. We keep the training configuration fixed across all methods, changing only how the training spans are selected.

Table 3: Model-annotated spans outperform intrinsic metric-selected spans on LongBench-v2 with Qwen3-4B-Thinking-2507.

Span selector	**<64k**	**64–128k**
Model annotation	**47.7**	**35.3**
Perplexity score	46.7	31.9
Entropy score	45.1	33.0

Table 3 shows that model-annotated spans perform best in both length buckets, indicating that model intrinsic metrics are not the best choice for selecting useful TTT data. The gap is small below 64k for perplexity-selected spans but becomes much larger in the 64k–128k bucket. This is the regime where the context contains more distractors and where selecting question-relevant evidence becomes most important.

These results suggest that useful TTT spans are not simply the spans that are surprising or uncertain under the language model. High-perplexity or high-entropy text may be difficult to predict for many reasons unrelated to the question, such as formatting, rare entities, or local distribution shift. In contrast, model annotation conditions span selection on the question, allowing it to better target evidence that can improve the final answer.

4.2 Case study

We next visualize how S-TTT changes the model's use of the selected span. We compare question-and-answer-to-context attention before and after S-TTT, averaging over all heads and plotting the attention by layer. Figure 2 shows one such example. Before adaptation, the model already assigns some attention to the annotated evidence span, but the mass is sparse and uneven across layers. After training on that span, attention becomes stronger and more continuous around the selected tokens, especially in the middle layers. The difference panel shows that this change is localized: the warm region aligns with the training span, while most neighboring positions remain close to zero. This qualitative example suggests one mechanism behind S-TTT: adaptation on selected evidence induces

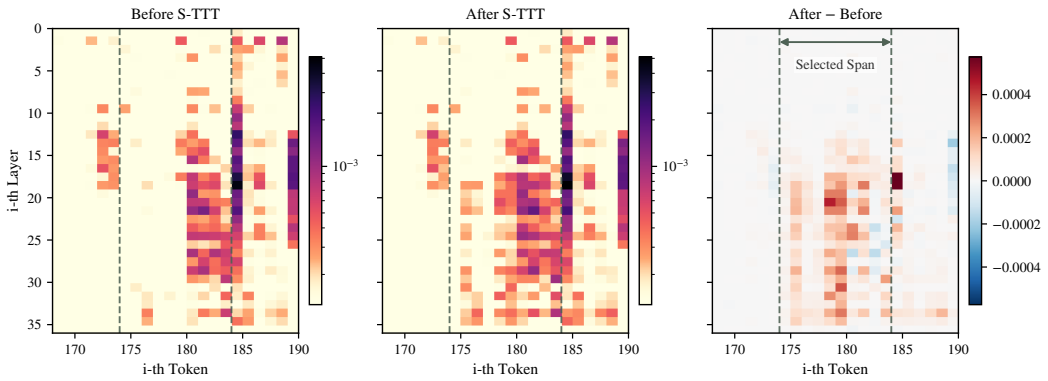

Figure 2: A concrete example of question-and-answer-to-context attention before and after S-TTT. Rows are layers and columns are context positions around the model-annotated span. The dashed vertical lines mark the selected training tokens. After S-TTT, attention to the selected span increases, while the change outside the span remains small.

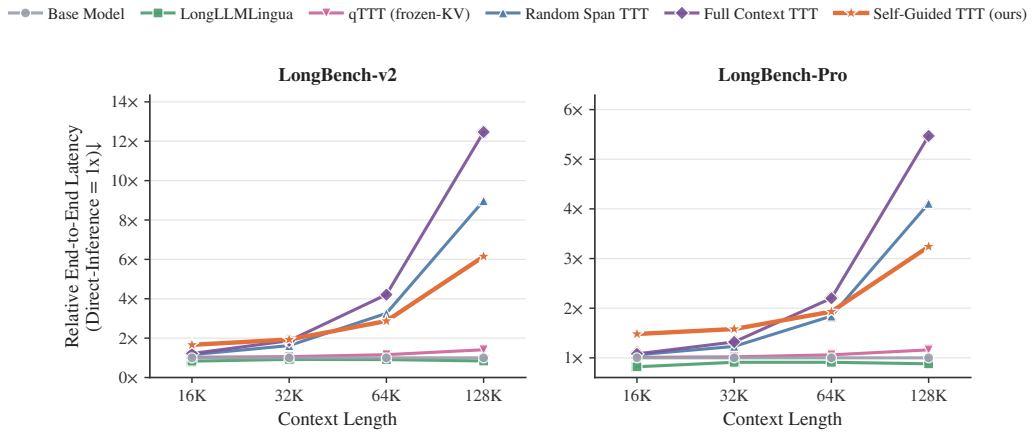

Figure 3: End-to-end latency normalized by full-context inference as context length increases using Qwen3-4B-Thinking-2507. S-TTT incurs a higher latency in the beginning, but it becomes cheaper than other non-frozen KV cache TTT methods at longer context.

a localized shift in attention toward tokens that are relevant to the current question. More visualized examples can be found in Appendix D.

4.3 Efficiency Analysis

TTT methods introduce extra overhead over direct inference because they add an adaptation stage before generation. We use pytorch FSDP (Paszke et al., 2019) for training and vLLM (Kwon et al., 2023) for inference. All measurements are conducted on a single NVIDIA H200 GPU. Figure 3 reports measured end-to-end latency normalized by full-context inference using Qwen3-4B-Thinking-2507. S-TTT has a higher latency in the beginning when the context length is relatively short. The crossover happens at longer context: S-TTT becomes cheaper than Full Context TTT from 64k onward on both benchmarks, and cheaper than Random Span TTT at 64k on LongBench-v2 and comparable on LongBench-Pro. Notably, at 128k context length, S-TTT has the lowest latency among the non-frozen-KV TTT methods. This is expected because Full Context TTT will incur significant overhead when the context length scales up as it trains on the entire input. Random Span TTT samples spans uniformly across the full context, which leads to a larger average effective training window of $0.50C$, where C is the context length. In contrast, model-annotated spans are more localized, resulting in shorter effective training windows on average ($0.39C$ on LongBench-v2 and $0.37C$ on

LongBench-Pro). The annotation cost dominates at shorter lengths, but quickly becomes smaller than the saved adaptation cost at long context.

5 Related Work

Test-Time Training. TTT adapts model parameters to a single test input before prediction, using supervision derived from the input itself rather than from new labels (Sun et al., 2020). Earlier work studies when self-supervised TTT helps or fails under distribution shift (Liu et al., 2021), and nearest-neighbor TTT adapts LLMs using retrieved examples at inference time (Hardt & Sun, 2024). More recent LLM work shows that per-instance adaptation can improve reasoning when the test input contains useful self-supervision (Akyürek et al., 2024). For long-context tasks, Bansal et al. (2026) show that TTT can be a more effective use of inference-time compute than simply generating more reasoning tokens. Related long-context TTT work also explores parameter-efficient adaptation for reasoning over long inputs (Chen et al., 2026a). These works primarily study how to perform adaptation efficiently. In this work, we study what tokens the model should be trained on at test time. S-TTT shows that selecting the right spans is a key component of effective long-context TTT.

Long-Context LLMs. Modern LLMs increasingly support very long context windows, but a longer window does not guarantee reliable use of the information inside it. Models remain sensitive to evidence position, often degrading when relevant content appears in the middle of a long input (Liu et al., 2024), and long-context benchmarks such as LongBench, LongBench-v2, LongBench-Pro, ZeroSCROLLS, RULER, and HELMET make these failures visible across multi-document QA, code, dialogue, structured reasoning, recall, and long in-context learning tasks (Bai et al., 2024; 2025; Chen et al., 2026b; Shaham et al., 2023; Hsieh et al., 2024; Yen et al., 2025). A broad line of work addresses long-context limitations by extending usable context windows (Peng et al., 2024; Chen et al., 2024), improving prefill or attention efficiency (Jiang et al., 2024a), compressing prompts (Jiang et al., 2024b), retrieving external evidence (Lewis et al., 2020; Zhang et al., 2025b), or analyzing and steering attention behavior at inference time (Wu et al., 2025; Zhang et al., 2025b; Ye et al., 2026). These approaches largely aim to help the model condition on the right evidence or process long inputs more efficiently. We instead approach long-context reasoning from the perspective of test-time training: rather than compressing the long context or intervening in the decoding procedure, S-TTT only requires the model to first select relevant spans from the context before TTT. This keeps the model architecture and decoding algorithm unchanged, avoiding complex interventions that are often infeasible in modern inference engines, while yielding consistent gains.

6 Conclusion

We propose Self-Guided TTT (S-TTT), a simple test-time adaptation framework for long-context LLMs that uses the model itself to select question-relevant evidence spans for training. Instead of adapting on the full context or on randomly sampled spans, S-TTT first identifies supporting spans from the input context, adapts the model only on those selected spans, and then generates the final answer using the original full context. Our results on LongBench-v2 and LongBench-Pro show that S-TTT consistently improves over TTT on random span across Qwen3 and Llama-3.1 models, while remaining cheaper than other TTT variants at long context. Empirical results demonstrate that the effectiveness of long-context TTT depends critically on the quality of the test-time training tokens. Overall, S-TTT provides a simple yet effective framework for long-context test-time training, highlighting training-token selection as a promising direction for solving long-context tasks with TTT. We discuss future directions in Appendix E.

AI use statement

We used generative AI tools to assist with language editing of this manuscript, including polishing wording, improving clarity and readability, and identifying grammatical, spelling, and typographical errors. All AI-suggested edits were reviewed by the authors before being incorporated. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

Ethics statement

This work trains and evaluates publicly available LLMs on publicly released long-context benchmarks and does not involve human subjects, newly collected data, or personally identifiable information.

Reproducibility statement

We describe S-TTT in Section 2.2 and give the full procedure in Algorithm 1. Section 3.1 specifies the base models, the benchmarks and data preprocessing, the configuration of every baseline, and the evaluation protocol. Appendix A gives the LoRA configuration, the optimizer settings, and the learning rate. Appendix B reports the rate at which S-TTT falls back to random spans, and Appendix C lists the prompt templates for each model and benchmark. All results in Table 2 are averaged over three independent runs, with standard deviations reported. The software stack and the hardware are described in Section 4.3.

References

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2308.14508.

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. 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 (ACL)*, 2025. URL https://aclanthology.org/2025.acl-long.183/.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Fnu Devvrit, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Kale, and Samy Jelassi. Let's (not) just put things in context: Test-time training for long-context LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H0bcEdPCoc.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongLoRA: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=qxDTe8fIyA.

Ziyang Chen, Xing Wu, Junlong Jia, Chaochen Gao, Qi Fu, Debing Zhang, and Songlin Hu. Longbench pro: A more realistic and comprehensive bilingual long-context evaluation benchmark. *CoRR*, abs/2601.02872, 2026b.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Wenhao Huang, Di He, and Tianle Cai. In-place test-time training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dTWfCLSoyl.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=CNL2bku4ra.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language

models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1658–1677, 2024b. URL https://aclanthology.org/2024.acl-long.91.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 9459–9474, 2020. URL https://arxiv.org/abs/2005.11401.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 21808–21820, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b618c3210e934362ac261db280128c22-Abstract.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.

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

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. ZeroSCROLLS: A zero-shot benchmark for long text understanding. In *Findings of the Association for Computational Linguistics: EMNLP*, pp. 7977–7989, 2023. doi: 10.18653/v1/2023.findings-emnlp.536. URL https://aclanthology.org/2023.findings-emnlp.536.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pp. 9229–9248, 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. Retrieval head mechanistically explains long-context factuality. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EytBpUGB1Z.

Xi Ye, Wuwei Zhang, Fangcong Yin, Howard Yen, and Danqi Chen. DySCO: Dynamic Attention-Scaling Decoding for Long-Context Language Models, April 2026. URL http://arxiv.org/abs/2602.22175. arXiv:2602.22175 [cs].

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=293V3bJbmE.

Tianyuan Zhang, Sai Bi, Yicong Hong, Kai Zhang, Fujun Luan, Songlin Yang, Kalyan Sunkavalli, William T Freeman, and Hao Tan. Test-time training done right. *arXiv preprint arXiv:2505.23884*, 2025a.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. Query-focused retrieval heads improve long-context reasoning and re-ranking. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23791–23805, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1214. URL https://aclanthology.org/2025.emnlp-main.1214/.

A Implementation Details

We use LoRA (Hu et al., 2022) for parameter-efficient test-time training. Following qTTT (Bansal et al., 2026), we apply LoRA only to the query projection layers, with rank $r = 16$ and scaling parameter $\alpha = 32$. We optimize the LoRA parameters using AdamW with a 0.01 weight decay. For each method, we sweep the learning rate over $\{3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$ on a small validation set and select the best one for testing. For span annotation, LongBench-v2, which consists of multiple-choice questions, we append the answer choices to the question when prompting the model to annotate relevant spans. For LongBench-Pro, which contains open-ended questions, span annotation is performed using only the context and the question.

B Annotation Coverage

Table 4 reports how often the model produces valid verbatim spans. Fallback instances use random spans, so they are equivalent to Random Span TTT for those cases. The fallback rate is low on LongBench-v2 for both base models, indicating that most instances receive genuine model-selected training spans. However, the fallback rate becomes higher on LongBench-Pro, especially for Llama-3.1-8B-Instruct, suggesting that self-annotation is more difficult on the open-ended benchmark.

C Prompts

Table 4: Model annotation coverage on LongBench-v2 and LongBench-Pro. "Fallback" means the model fails to produce valid verbatim spans.

Model	Benchmark	Fallback
Qwen3-4B-Thinking-2507	LongBench-v2	8.2%
Qwen3-4B-Thinking-2507	LongBench-Pro	21.5%
Llama-3.1-8B-Instruct	LongBench-v2	6.9%
Llama-3.1-8B-Instruct	LongBench-Pro	39.9%

Table 5: Qwen3-4B-Thinking-2507 prompt template for LongBench-v2.

```
<|im_start|>system
You are a helpful assistant.  Read the context and answer the
question.<|im_end|>
<|im_start|>user
{CONTEXT}

{QUESTION}
Pick from the following options:
{CHOICES}
Please show your choice in the answer field with only the choice
letter,
e.g., "answer":  "C".<|im_end|>
<|im_start|>assistant
<think>
```

Table 6: Qwen3-4B-Thinking-2507 prompt template for LongBench-Pro.

```
<|im_start|>system
You are a helpful assistant.  Read the context and answer the
question.<|im_end|>
<|im_start|>user
{CONTEXT}

{QUESTION}<|im_end|>
<|im_start|>assistant
<think>
```

Table 7: Llama-3.1-8B-Instruct prompt template for LongBench-v2.

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date:  December 2023
Today Date:  26 Jul 2024

<|eot_id|><|start_header_id|>user<|end_header_id|>

Please read the following text and answer the question below.

<text>
{CONTEXT}
</text>

What is the correct answer to this question:  {QUESTION}
Choices:
{CHOICES}

Let's think step by step.  After thinking, choose a single, most
likely
answer.  Output your final answer follows:  "The correct answer is
(insert choice here)".<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Table 8: Llama-3.1-8B-Instruct prompt template for LongBench-Pro.

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date:  December 2023
Today Date:  26 Jul 2024

You are a helpful assistant.  Read the context and answer the
question.<|eot_id|><|start_header_id|>user<|end_header_id|>

{CONTEXT}

{QUESTION}<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

D QUALITATIVE EXAMPLES

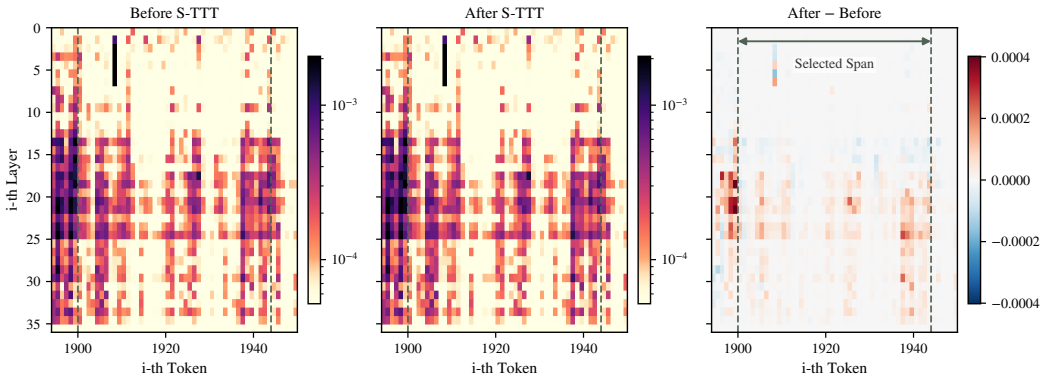

Figure 4: Example of span attention before and after S-TTT.

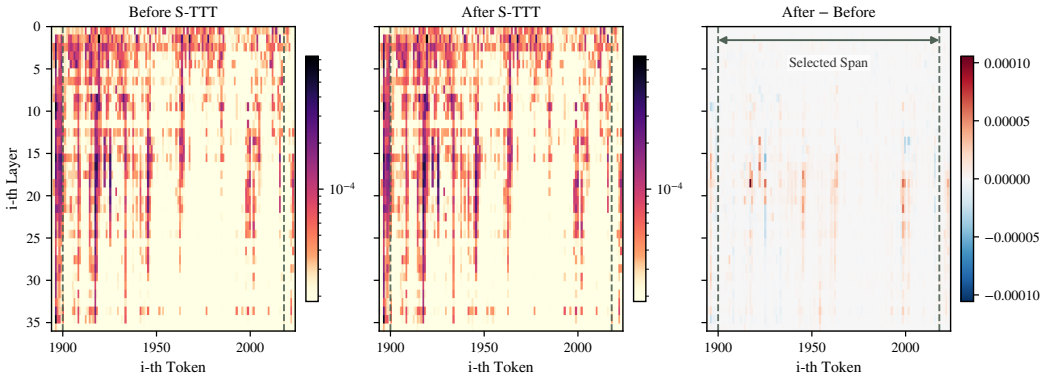

Figure 5: Example of span attention before and after S-TTT.

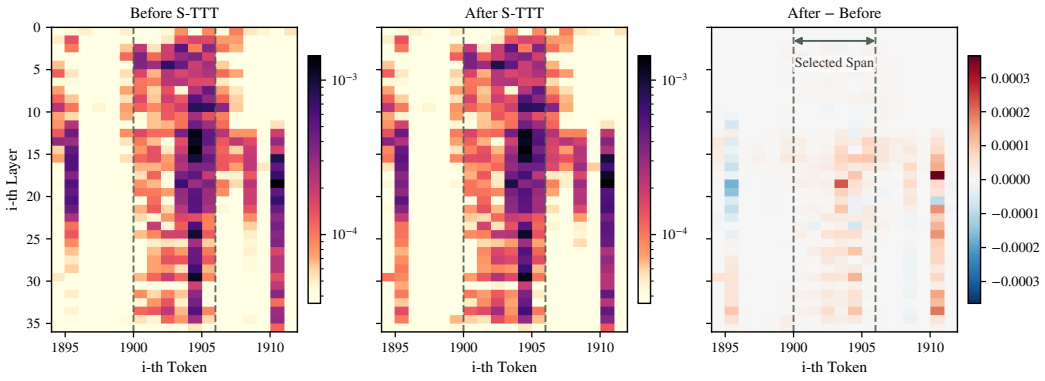

Figure 6: Example of span attention before and after S-TTT.

E FUTURE DIRECTIONS

TTT opens up opportunities for adapting LLMs in realistic production settings. In many applications, users may upload a long document, such as a financial report, legal contract, or book, and then ask multiple questions about it. Unlike methods that require architectural changes or specialized attention

mechanisms, TTT relies on standard gradient-based adaptation and on-the-fly weight updates. This makes it compatible with modern training and serving infrastructure, where each conversation session could maintain its own lightweight adapted weights for multi-turn use, personalization, or document-specific specialization. However, the largest bottleneck is latency: even parameter-efficient TTT adds adaptation overhead before generation, which needs to be reduced. Addressing these challenges is an important direction for making TTT a practical framework for solving long-context tasks in real-world systems.