LongAttnComp: Cross-Family Context Compression for Long-Context Reasoning

Mengmeng Ji [1] **Ravi Shanker Raju** [1] **Jonathan Lingjie Li** [1] **Chen Wu** [1]

Abstract

As real-world applications increasingly require processing inputs of 100k+ tokens that approach or exceed standard LLM context windows—retrieved documents, long conversations, or extended codebases—the gap between context length and inference efficiency has become a critical bottleneck. Long-context inference imposes significant memory and compute costs, motivating efficient context compression. We observe that long-context task performance decomposes into retrieval and reasoning, and existing training-free attention-based compression methods leave a substantial gap on the retrieval step in demanding long-context settings such as code understanding. We present LongAttnComp, a long-context adaptation of AttnComp (Luo et al., 2025), fine-tuning a lightweight cross-attention scoring layer and introducing token-level chunking, a token-budget top-p algorithm, and a format-agnostic query parser. Trained solely on NIAH-style data, LongAttnComp matches or exceeds full-context performance and substantially outperforms training-free baselines across multiple target models on InfiniteBench Code-Debug. Per-task analysis on RULER and LongBench v2 confirms LongAttnComp's strength on long-context code reasoning while indicating that broader long-context applicability is achievable through more diverse training data.

1. Introduction

Long-context inference with large language models (LLMs) imposes significant memory and compute costs. As real-world applications increasingly require processing tens of thousands of tokens — retrieved documents, long conversations, or extended codebases — the gap between context length and inference efficiency has become a critical bottleneck. *Context compression* addresses this concern by filtering or condensing the input context before it reaches the target model (Jiang et al., 2023; Xu et al., 2023), trading a small upfront cost for substantial savings in the target model's prefill stage.

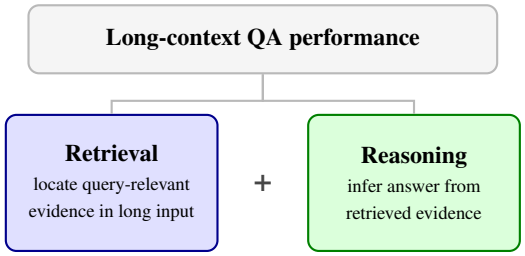


Figure 1. Long-context task performance decomposes into retrieval and reasoning (Yang et al., 2018); we focus on the retrieval bottleneck (Liu et al., 2024), framing compression as a retrieval problem.

Performance on long-context tasks depends on two factors: reliably retrieving the query-relevant content, and reasoning correctly over it. This decomposition is illustrated in Figure 1. We focus on the retrieval step as the primary bottleneck — effective context compression is fundamentally a retrieval problem, requiring the compressor to identify which tokens and segments carry query-relevant information.

Among context compression solutions, Speculative Prefill (Liu et al., 2025) establishes an attention-based, training-free, draft-model-driven compression framework with strong performance across tasks and target model families (Upasani et al., 2026), yet shows a substantial performance gap on long-context code reasoning. AttnComp (Luo et al., 2025) takes a fine-tuning approach: a frozen LLM backbone with a trainable cross-attention layer scores each document for query relevance. While AttnComp's mechanism shows promise, its evaluation and training are narrowly scoped—retrieval-augmented QA at ∼12k-token inputs, training from a single source (HotpotQA), and document-level scoring—leaving its potential as a general-purpose long-context compressor untested.

[1]SambaNova Systems, San Jose, CA, USA. Correspondence to: Mengmeng Ji <mengmeng.ji@sambanovasystems.com>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

We propose **LongAttnComp**, a robust, modular long-context compressor that adapts AttnComp's (Luo et al., 2025) fine-tuned compression mechanism for the draft-model-driven framework of Speculative Prefill (Liu et al., 2025), delivering strong cross-family performance on long-context retrieval and reasoning tasks. We retain AttnComp's core mechanism—a fine-tuned cross-attention layer and top-p selection algorithm—and extend it for inference at the 100k+ token regime. We make four adaptations for long context inference. First, we use **token-level chunking** rather than document-level scoring, enabling flexible operation on real-world long-context inputs that often lack natural document boundaries.Second, we replace AttnComp's score-threshold top-p algorithm with a **token-budget variant** for predictable compression length. Third, we apply **positional reordering** to restore selected chunks to their original order before passing them to the target model. Fourth, we introduce a **format-agnostic query parser** to handle inputs without fixed query templates. On Code-Debug task of InfiniteBench, LongAttnComp matches or exceeds full-context performance, substantially outperforms training-free baseline (speculative prefill), and transfers across three unrelated target model families without retraining.

Our contributions are:

- A **long-context adaptation of AttnComp** extended with token-level chunking, a token-budget top-p selection algorithm, a format-agnostic query parser, and a flexible target-model wrapper (Figure 2, §3).
- A **curated training corpus** combining SQuAD and HotpotQA under a NIAH-style construction with dataset-derived relevance labels, covering single- and multi-hop retrieval at 8k–48k token sequences (§4).
- **Empirical findings**: LongAttnComp (i) closes the training-free performance gap on long-context code reasoning (InfiniteBench Code-Debug), exceeding full-context inference; (ii) transfers across three unrelated target model families without retraining; and (iii) reveals a task-sensitivity pattern on RULER and LongBench v2 that we attribute to training-data composition, motivating a concrete next-step direction for broader long-context generalization (§6, §7).

We plan to release our code and data to facilitate further research on long-context compression.

2. Related Work

Context compression. Context compression methods fall into two broad categories. *Abstractive* approaches train auxiliary models to produce condensed representations of the input (Xu et al., 2023; Yoon et al., 2024); *extractive* approaches retain a subset of original tokens or segments using token-level perplexity (Jiang et al., 2023; 2024) or embedding-based semantic similarity (Xu et al., 2023). A shared limitation across both categories is reliance on a predetermined compression budget that does not adapt to the variable density of relevant content. Recent adaptive methods address this via per-sentence relevance classification or threshold-based scoring (Hwang et al., 2024; Chirkova et al., 2025). Our work shares this adaptive philosophy but operates on *fixed-size token chunks*, using fine-tuned cross-attention weights as the relevance signal, enabling end-to-end training within the speculative prefill framework.

Attention-based retrieval and speculative inference. Speculative Prefill (Liu et al., 2025) demonstrated that attention weights from a lightweight draft model serve as effective token-importance signals for training-free compression, with subsequent work showing transfer across model families (Upasani et al., 2026). AttnComp (Luo et al., 2025) introduced an independent attention-based approach: a fine-tuned cross-attention scoring layer for document-level compression with adaptive top-p selection, evaluated on short-context retrieval-augmented QA. Drawing on both lines, LongAttnComp parallels the draft-model paradigm of speculative decoding (Leviathan et al., 2023), fitting within the broader theme of resource-adaptive inference.

3. Method

We adapt AttnComp (Luo et al., 2025) for long-context inference. Figure 2 shows the full pipeline. AttnComp augments the first L layers of a frozen draft LLM with a trainable cross-attention layer that produces document-level relevance scores from query–context attention; only $\sim$0.5% of parameters are updated. We retain this architecture and training objective (for full details, see Appendix A and §5.2), extending it for long-context tasks with token-level chunking and scoring; a larger, more diverse training corpus; a modified top-p algorithm suited to control context budgets; and a format-agnostic query parser enabling evaluation across diverse benchmarks.

3.1. Token-level Chunking

Whereas the original AttnComp paper uses documents as the unit for scoring and selection, we operate at token-level chunk granularity: the input is partitioned into fixed-size token chunks, and the cross-attention layer scores each chunk independently. This design is more amenable for real-world long-context inputs, where the input is often a single long document or stream that cannot be cleanly partitioned into independent documents. Fixed-size token-level chunking also turns chunk size into a tunable hyperparameter. Different tasks may benefit from different chunk sizes, and sweeping chunk sizes across tasks lets us identify the best size for

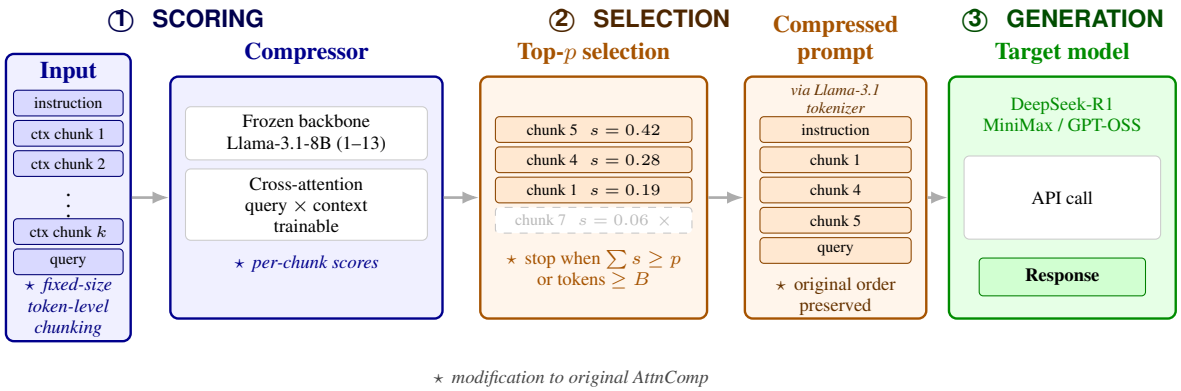


Figure 2. End-to-end workflow of the LongAttnComp compressor. **(1) Scoring**: a frozen Llama-3.1-8B backbone with a trainable cross-attention layer produces a relevance score per context chunk. **(2) Selection**: chunks are ranked by score and retained until cumulative top-p mass or token budget B is reached; selected token chunks are decoded into a natural-language compressed prompt by Llama3.1's tokenizer, with original positional order preserved. **(3) Inference**: the compressed prompt is sent to the target model via API.

each task. This, in turn, gives us a clearer understanding of how chunk granularity affects retrieval and reasoning in different long-context settings.

3.2. Modified Top-p Selection

We adapt AttnComp's top-p algorithm to operate on chunk-level scores. The original algorithm stops document selection when either the cumulative score exceeds p or a document's score falls below a minimum threshold ϵ. In practice, on long-context tasks, the minimum-score condition triggers first, causing overly aggressive compression and a significant performance drop.

We replace the minimum-score threshold with a **content token budget** B: selection now stops when either the cumulative score exceeds p or the retained tokens reach B (see Algorithm 1 in Appendix B). This gives direct, predictable control over compressed context length: with $B = 16$k, compressed prompts consistently approach the budget (15k+ tokens in practice), avoiding the degenerate under-retention caused by the score threshold.

Positional Reordering. After selection, retained token chunks are restored to their original positional order. The original AttnComp paper returns the selected documents as an unordered set; we preserve positional order to maintain discourse coherence.

3.3. Query Parsing

AttnComp requires identifying the query span within the concatenated input to form X_q, but the original paper does not discuss query parsing explicitly. It is understandable given that its standard QA evaluation uses fixed templates with well-defined query boundaries.

For long-context benchmarks such as InfiniteBench Code-Debug, the prompt structure is more complex. We evaluate two parsing strategies. **Accurate parsing** precisely extracts the query and instruction spans specific to the Code-Debug format, providing the compressor with clean, task-specific query representations. **Arbitrary parsing** allocates the last 128 tokens of the input as the query, with the instruction set identically. Notably, arbitrary parsing incurs only minor performance degradation relative to accurate parsing, suggesting LongAttnComp is robust to approximate query identification; we discuss implications in §7.

4. Data

We build our training corpus using a modified RULER data generation pipeline (Hsieh et al., 2024): each sample contains 100 documents, a query, and binary relevance labels, with 25% of samples constructed as all-negative (i.e., none of the 100 documents are relevant to the query). Our training data differs from AttnComp (Luo et al., 2025) in two ways. First, we curate from both SQuAD (0 or 1 relevant document per sample, single-fact retrieval) and HotpotQA (0 or 2, multi-hop retrieval) instead of HotpotQA alone, broadening coverage across single-needle and multi-hop retrieval. Second, we derive relevance labels directly from dataset-provided structural annotations—HotpotQA's `supporting_facts` field and SQuAD's single-passage construction—rather than AttnComp's Llama-3.1-based annotation pipeline, avoiding dependency on a labeler that shares the compressor's backbone.

We create two datasets: **16k dataset** (8k SQuAD + 8k HotpotQA samples, sequences 8k–48k tokens) and **32k dataset** (16k SQuAD + 16k HotpotQA samples, sequences 8k–48k tokens). Needle positions are uniformly distributed across front, middle, and end of the context (≈33% each) to avoid position bias. Full token-length and needle-position

distributions for all subsets are provided in Appendix C.

5. Experimental Setup

5.1. Models

Compressor. The compressor uses the first L=13 layers of `Llama-3.1-8B-Instruct` (Dubey et al., 2024) as a frozen backbone, with a trainable query-relevant cross-attention layer appended on top. Given a query and a long context, the compressor selects query-relevant token chunks via our modified top-p algorithm; selected chunks are concatenated and decoded using Llama-3.1's tokenizer to produce a natural-language compressed prompt, which is then passed to the target model (e.g., DeepSeek-R1-0528 (DeepSeek-AI, 2025)).

Target Models. We evaluate compressed prompts on three unrelated target models accessed via the SambaNova Cloud API: DeepSeek-R1-0528 (DeepSeek-AI, 2025), MiniMax-M2.5 (MiniMax, 2026), and GPT-OSS-120B (OpenAI, 2025). This selection tests both whether compression retains sufficient information for downstream answering and whether the compressor generalizes beyond the `Llama-3.1-8B-Instruct` family it was trained on.

5.2. Compressor Training Details

The compressor trained on the 32k dataset achieves the best performance across benchmarks and is used for all reported results. We hold out 5% of the training data as a validation split and select the best checkpoint by validation loss. The cross-attention layer is trained for 15 epochs using the AdamW optimizer (weight decay 0.01), with a learning rate of 2×10^{-4} following a cosine decay schedule with linear warmup, batch size 8, gradient accumulation steps 1, and a dropout rate of 0.1. Gradient clipping with maximum norm 1.0 is applied. All experiments are run on 8×H200 GPUs.

5.3. Evaluation Benchmarks

We center our evaluation of LongAttnComp on **Code Debug** from InfiniteBench (Zhang et al., 2024), a multiple-choice bug-identification benchmark over long code inputs averaging ∼115k tokens with some samples exceeding 200k. We focus on this task for two reasons. First, the original AttnComp work was evaluated exclusively on Wikipedia-based QA (Luo et al., 2025), leaving open whether attention-guided compression remains effective in substantially longer contexts and in domains beyond natural language; Code Debug provides a direct stress test along both axes. Second, the task couples *retrieval*—locating the relevant buggy region within a long codebase—with *reasoning*—interpreting code semantics to identify the correct option—probing what information a compressor must preserve to support downstream long-context inference.

Table 1. Accuracy (%) on InfiniteBench Code-Debug with DeepSeek-R1-0528. All LongAttnComp configurations use top-$p = 0.95$ and a 16k token budget (compression rate ≈83%). **Accurate query** uses task-specific query extraction; **Arbitrary query** takes the last 128 tokens as the query. Input budget: 120k tokens (8k reserved for output); over-budget inputs are middle-truncated. `</think>` tags are stripped before answer extraction.

Method	Chunk	Acc. query	Arb. query
Full context	–	74.37	–
Speculative Prefill	128	62.44	–
16k, const LR, 15ep	128	61.42	–
	256	68.02	–
	512	74.37	–
	1024	73.86	–
16k, cos LR, 30ep	512	74.37	–
	1024	75.13	75.13
32k, cos LR, 15ep	128	56.85	–
	256	72.08	–
	512	72.08	–
	1024	**76.40**	**75.38**

To assess generalization beyond code, we additionally evaluate on **RULER** (Hsieh et al., 2024), a synthetic suite measuring long-context utilization across single- and multi-needle retrieval and question answering, and **LongBench v2** (Bai et al., 2024), a suite of long-document tasks requiring multi-hop inference and fine-grained comprehension across diverse domains.

6. Results

We organize our evaluation along two axes. Holding the task fixed at long-context code reasoning, we ask: (i) does LongAttnComp match or exceed full-context performance on InfiniteBench Code-Debug (§6.1); and (ii) does the same compressor, trained with a `Llama-3.1-8B-Instruct` draft model, generalize to unrelated target model families (§6.2). Holding the compressor fixed and varying tasks, we further ask: (iii) how does LongAttnComp's effectiveness change across the retrieval and reasoning demands of RULER and LongBench v2 beyond code understanding, and what does this reveal about the compressor's task-type sensitivity (§6.3).

6.1. Long-Context Code Reasoning

We evaluate LongAttnComp on InfiniteBench Code-Debug—the setting where speculative prefill has shown the largest gap to full-context inference. We use DeepSeek-R1-0528 as the target model and reporting accuracy on the full test set. Table 1 compares the trained compressor against

Table 2. Task-sensitivity analysis on RULER. LongAttnComp uses the 32k cosine-LR 15-epoch checkpoint with chunk size 256, top-p tuned, and a 16k input budget.

Target	Method	niah_s_1	niah_s_2	niah_s_3	niah_multik_1	niah_multik_2	niah_multik_3	niah_multiv	niah_multiq	qa_1	qa_2
Llama-3.1	Full context	100.0	99.4	99.8	97.2	74.6	55.2	93.7	97.7	71.8	43.8
	+ LongAttnComp	100.0	99.4	98.2	92.8	73.6	79.0	64.0	81.5	79.6	62.0
DeepSeek-R1	Full context	94.9	91.8	57.4	88.4	89.4	44.4	88.9	91.7	71.4	72.0
	+ LongAttnComp	100.0	99.4	98.2	92.8	73.6	80.0	64.0	81.5	89.0	83.4

two baselines: **Full context**, the uncompressed prompt sent directly to the target, and **Speculative Prefill** (Liu et al., 2025), a training-free attention-based compressor that uses the same Llama-3.1-8B-Instruct draft model.

Our best configuration—32k training corpus, cosine-decay learning rate, 15 epochs, chunk size 1024, token budget B=16k—reaches **76.40%**, exceeding full-context by 2.0 points and Speculative Prefill by 13.9 points while compressing the prompt by 83%. Two patterns emerge. First, *chunk size has a large impact on accuracy*: accuracy increases from 56.85% at chunk 128 to 76.40% at chunk 1024 on the 32k checkpoint. Second, *training schedule and corpus size also contribute*: cosine decay and the 32k corpus each improve over their constant-LR and 16k counterparts, but the 16k cosine checkpoint at 30 epochs already reaches 75.13%, indicating that longer training partially compensates for smaller data. Together, these results indicate that LongAttnComp paired with larger chunk sizes is particularly effective at retaining task-relevant information for long-context multiple-choice reasoning.

Robustness to query parsing. Replacing task-specific query extraction with the last 128 tokens of the prompt costs only 1.0 point on the 32k checkpoint (76.40% $\rightarrow$ 75.38%) and is neutral on the 16k checkpoint (75.13% in both). Accurate query parsing remains a practical challenge for deployment, but this narrow gap suggests LongAttnComp is largely robust to query specification.

6.2. Cross-Family Generalization

Table 3 reports cross-family results. LongAttnComp matches or closely tracks each target's full-context accuracy—exceeding DeepSeek-R1-0528 by 1.0 points and trailing MiniMax-M2.5 and GPT-OSS-120B by under 3 points each—while consistently outperforming Speculative Prefill by 12.9, 24.1, and 30.7 points on the same three targets. Because the same Llama-3.1-trained compressor produces all three sets of numbers without any target-specific fine-tuning or hyperparameter adjustment, LongAttnComp behaves as a modular, target-agnostic preprocessing step, delivering robust long-context code reasoning across reasoning, coding, and general-purpose model families.

Table 3. Cross-family generalization on Code-Debug. The same LongAttnComp compressor (32k corpus, cosine LR, 15 ep., chunk 1024) and the same Speculative Prefill configuration (chunk 128) are deployed across three target models; all runs use top-p=0.95 and a 16k compression budget. Full-context budget equals context window − 8k (reserved for response via middle-token truncation). `</think>` tags are stripped before answer extraction.All accuracies are reported under arbitrary query parsing (last 128 tokens).

Target model	Method	Ctx. budget	Accuracy
DeepSeek-R1-0528	Full context	120k	74.37
	Spec Prefill	16k	62.44
	LongAttnComp	16k	**75.38**
MiniMax-M2.5	Full context	152k	83.76
	Spec Prefill	16k	57.10
	LongAttnComp	16k	**81.22**
GPT-OSS-120B	Full context	120k	86.00
	Spec Prefill	16k	52.28
	LongAttnComp	16k	**82.99**

6.3. Task Sensitivity Analysis

Beyond code reasoning, we evaluate LongAttnComp on RULER to characterize per-task behavior across diverse long-context skills. We compare two settings: the **Llama-3.1 in-family** setting isolates compressor effects from target-model behavior, and the **DeepSeek-R1-0528** setting tests whether the same pattern persists with an unrelated downstream reasoner. Table 2 shows that LongAttnComp preserves single-needle retrieval, multi-key retrival and improves QA, while degrading on multi-value and multi-query retrieval. [1] A complementary evaluation on LongBench v2 with DeepSeek-R1-0528 (Appendix F.1)—which emphasizes multi-document retrieval and complex multi-hop reasoning—shows similar underperformance. We attribute

[1]For RULER, we use the test set generated with the Llama-3.1 tokenizer at 128k length. For DeepSeek-R1, the baseline RULER input is middle-truncated to fit within DeepSeek's effective input budget (after reserving capacity for `<think>` reasoning), since DeepSeek's tokenizer differs from the Llama-3.1 tokenizer. For LongAttnComp, the input fits within the compressor's Llama-3.1-8B-Instruct context window; the compressed output (with a 16k token budget) is then sent to DeepSeek-R1. The compressor recovers single-needle accuracy that truncation would otherwise lose (e.g., `niah_s_3`: 57.4 $\rightarrow$ 98.2). On multi-value and multi-query subtasks, LongAttnComp underperforms the truncated baseline, consistent with the spread-evidence weakness noted in the body.

this task-type variation to training-data composition: the current corpus consists only of synthetic NIAH-style data from SQuAD and HotpotQA, and predictably underperforms on tasks involving naturalistic text and nuanced reasoning patterns.

7. Discussion

When does LongAttnComp work? Per-task analysis reveals a consistent pattern. LongAttnComp performs well on tasks where (i) evidence is clearly query-aligned and (ii) the amount of evidence to be retrieved is bounded—which captures Code-Debug (query-driven retrieval, limited buggy region) and RULER's single-needle, multi-key, QA. It underperforms when evidence is indirectly query-aligned (requiring inference to identify) or spread across many locations (requiring aggregation), which captures RULER's multi-value and multi-query NIAH and LongBench v2's naturalistic multi-document reasoning. On one hand, this pattern aligns with the compressor's relatively narrow training corpus—synthetic NIAH-style data drawn solely from SQuAD and HotpotQA. On the other hand, LongAttnComp achieves strong performance on multiple long-context tasks despite this narrow training scope, suggesting that the method itself is robust. We therefore hypothesize that the per-task variation reflects training-data composition rather than a fundamental limitation of the method, and have already curated a second-stage training corpus targeting more diverse retrieval and reasoning patterns; training and evaluation form the immediate next step.

Chunk size as a task-dependent hyperparameter. Optimal chunk size varies by task—1024 tokens for Code-Debug, where evidence spans full functions, and 256 for RULER—suggesting chunk size should be set per task rather than as a universal constant.

Robustness to query parsing. LongAttnComp is largely robust to query parsing: replacing task-specific extraction with the last 128 tokens of the prompt costs at most 1.0 point on Code-Debug. This robustness supports deployment where reliable query identification is unavailable, a common case for general-purpose long-context inputs. The small remaining gap nevertheless suggests that accurate parsing helps at the margins; developing a robust task-agnostic query parser is a natural direction for further improvement.

Efficiency. Once properly trained, LongAttnComp has a smaller compute and memory footprint than previous draft-model-based methods such as Speculative Prefill, which uses the full draft model rather than the first $L{=}13$ layers. For reference, Speculative Prefill reports a TTFT reduction from 46s to 2.5s when compressing 128k tokens to 16k with a Llama-3.1-8B draft model (Upasani et al., 2026); since LongAttnComp's compressor uses only the first 13 of 32 layers of the same backbone, compression overhead should be roughly one-third of Speculative Prefill's, at comparable or better accuracy (Table 1).

Adaptive compression rate. The token-budget top-p algorithm adapts compression rate to content density: when a small subset of chunks captures enough query relevance to reach the top-p threshold, selection terminates well below the budget B. On RULER's `niah_s_1` (single-needle retrieval), LongAttnComp reaches 100% accuracy with compressed prompts averaging $\sim$2k tokens despite the 16k budget—a $\sim 64\times$ reduction from the 128k input that yields additional inference speedup beyond the budget-based compression rate.

Future work. Curating a more diverse training corpus—particularly natural-language documents and reasoning-heavy tasks beyond SQuAD/HotpotQA—is the most direct path to extending LongAttnComp's effectiveness to LongBench-v2-style benchmarks. We have curated such a second-stage corpus targeting these gaps; training and evaluation are immediate next steps. Meanwhile, from a deployment perspective, two future directions arise: First, since optimal chunk size varies by task (larger for code reasoning, smaller for sentence-level retrieval), an adaptive chunk-size selection mechanism would simplify deployment to inputs whose task type is unknown in advance. Second, the robust task-agnostic query parser noted above is a complementary direction for improving deployment performance. As a longer-term direction, fine-tuning the draft model itself to better align with target model behavior is a natural avenue we considered but did not pursue in this work: high-quality long-context training data—particularly for code-reasoning tasks—remains scarce, and generation pipelines for such data are not openly available. We leave this as a meaningful direction for future exploration.

8. Conclusion

We presented LongAttnComp, an effective fine-tuning-based long-context compression method. The trained compressor acts as a modular, target-agnostic preprocessing step that transfers across unrelated target model families without retraining. Per-task effectiveness varies across broader benchmarks, but this variation reflects the compressor's narrow training distribution rather than a fundamental limitation of the method—suggesting that with more diverse training data, the same architecture can extend to more complex long-context reasoning tasks.

Impact Statement

This paper presents work whose goal is to advance efficient inference for large language models. More efficient LLM inference reduces energy consumption and broadens access

to capable models. There are no specific negative societal consequences we feel must be highlighted.

Acknowledgments

We thank Bo Li for his guidance and support throughout this project, and Taylor Lee for help setting up API endpoints used in our experiments.

References

Bai, Y., Tu, S., Zhang, J., Peng, H., Wang, X., Lv, X., Cao, S., Xu, J., Hou, L., Dong, Y., Tang, J., and Li, J. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Chirkova, N., Rücklé, A., Gurevych, I., and Nikoulina, V. Provence: Efficient and robust context pruning for retrieval-augmented generation. *arXiv preprint arXiv:2501.16214*, 2025.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Dubey, A., Jauhri, A., Pandey, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hsieh, C.-P., Sun, S., Kriman, S., Agrawal, S., Rekesh, D., Fu, J., Zhang, Y., and Ginsburg, B. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Hwang, T., Jeong, S., Lim, J., Song, S., and Park, J. EXIT: Context-aware extractive compression for enhancing retrieval-augmented generation. *arXiv preprint arXiv:2412.12559*, 2024.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. 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*, 2024.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 19274–19286, 2023.

Liu, J., Chen, B., and Zhang, C. Speculative prefill: Turbocharging TTFT with lightweight and training-free token importance estimation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Luo, L., Cao, Y., and Luo, P. AttnComp: Attention-guided adaptive context compression for retrieval-augmented generation. *arXiv preprint arXiv:2509.17486*, 2025.

MiniMax. MiniMax M2.5: Built for real-world productivity. https://www.minimax.io/news/minimax-m25, 2026.

OpenAI. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Upasani, S., Raju, R. S., Li, B., Ji, M., Long, J., Wu, C., Thakker, U., and Wang, G. Cross-family speculative prefill: Training-free long-context compression with small draft models. In *International Conference on Learning Representations*, 2026. arXiv:2603.02631.

Xu, F., Shi, W., and Choi, E. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*, 2023.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Yoon, C., Kim, T., Hwang, H., Jeong, M., and Kang, J. CompAct: Compressing retrieved documents actively for question answering. *arXiv preprint arXiv:2407.09014*, 2024.

Zhang, X., Chen, Y., Hu, S., Xu, Z., Chen, J., Hao, M. K., Han, X., Thai, Z. L., Wang, S., Liu, Z., and Sun, M. ∞Bench: Extending long context evaluation beyond 100k tokens. *arXiv preprint arXiv:2402.13718*, 2024.

A. Background: AttnComp Review

We review the AttnComp framework (Luo et al., 2025) that our method builds upon, covering its architecture, training procedure, and compression algorithm.

Architecture. Given an instruction I, k retrieved documents $\mathcal{D} = \{d_1, \ldots, d_k\}$, and a query q, the concatenated input $[I; d_1; \ldots; d_k; q]$ is passed through the first L frozen layers of a draft LLM, yielding hidden states $X_c \in \mathbb{R}^{n \times d_{\text{model}}}$ for the context (instruction and documents) and $X_q \in \mathbb{R}^{m \times d_{\text{model}}}$ for the query. An additional cross-attention layer—initialized from layer $L{+}1$ of the LLM and the only trainable component—computes query-context attention weights $A \in \mathbb{R}^{m \times n}$:

$$
\begin{aligned}
Q_i = X_q W_i^Q, \quad K_i = X_c W_i^K, \\
A = \frac{1}{H} \sum_{i=1}^{H} \operatorname{softmax}\left(\frac{Q_i K_i^\top}{\sqrt{d_a}} \right),
\end{aligned}
\tag{1}
$$

where H is the number of attention heads and $W_i^Q, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_a}$ are per-head projection matrices. Freezing the first L layers and fine-tuning only the cross-attention layer updates $\approx 0.5\%$ of total parameters.

Training. Each training sample contains a query, 100 retrieved documents, and binary relevance labels $r_i \in \{0, 1\}$. AttnComp trains with two complementary losses. *Document-level supervision* discriminates relevant from irrelevant documents via binary cross-entropy:

$$
\mathcal{L}_{\text{doc}} = -\sum_{i=1}^{k} \big[r_i \log s_{d_i} + (1 - r_i) \log(1 - s_{d_i}) \big]. \tag{2}
$$

Instruction-level supervision handles the all-irrelevant case by directing attention to the instruction when no document is relevant:

$$
\mathcal{L}_{\text{ins}} = -\big[r_{\text{ins}} \log s_{\text{ins}} + (1 - r_{\text{ins}}) \log(1 - s_{\text{ins}}) \big], \tag{3}
$$

where $r_{\text{ins}} \triangleq \mathbb{I}\Big(\sum_{i=1}^{k} r_i = 0\Big)$. The combined loss is $\mathcal{L} = \mathcal{L}_{\text{doc}} + \lambda \mathcal{L}_{\text{ins}}$. AttnComp trains on 8k HotpotQA samples (25% all-negative) using the Adam optimizer with lr $= 2{\times}10^{-4}$, batch size 8, for 8 epochs with $\lambda = 0.8$. Relevance labels are obtained via an automated annotation pipeline.

Top-p compression. Each document d_j receives a scalar score s_{d_j} by aggregating rows of A over its token span; the instruction receives score s_{ins} similarly. Documents are sorted descending and a cumulative sum is accumulated starting from s_{ins}, retaining documents until the sum exceeds threshold p or a score falls below minimum ϵ, yielding $\mathcal{D}^* \subseteq \mathcal{D}$. AttnComp uses $p = 0.95$ and $\epsilon = 10^{-2}$.

B. Our Top-p Algorithm

Algorithm 1 presents our token-budget top-p selection procedure, replacing AttnComp's minimum-score threshold with a content token budget B that halts selection once the retained tokens reach B or the cumulative score exceeds p.

Algorithm 1 Token-Budget Top-p Compression

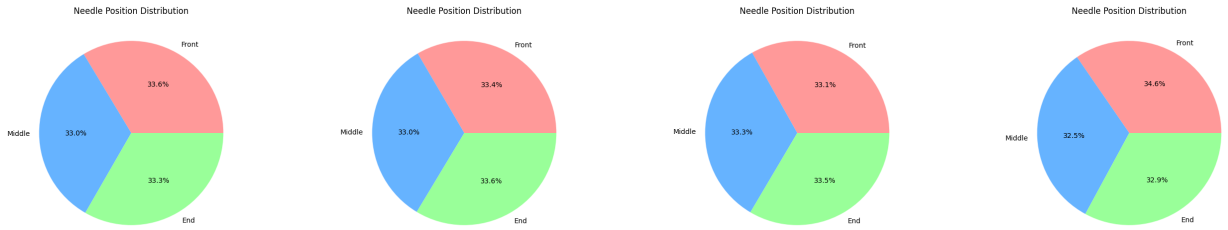
Input: Instruction score s_{ins}, document scores $\{s_{d_1}, \ldots, s_{d_k}\}$, top-$p$ threshold p, token budget B
Output: Compressed document set $\mathcal{D}'$
$\{d_{(1)}, \ldots, d_{(k)}\} \leftarrow \text{argsort}(\{s_{d_i}\}_{i=1}^{k}, \text{ desc.})$
Initialize $sum \leftarrow s_{\text{ins}}$, $\mathcal{D}' \leftarrow \emptyset$, $tokens \leftarrow 0$
for $i = 1$ **to** k **do**
  if $sum \geq p$ **or** $tokens + |d_{(i)}| > B$ **then**
    break
  end if
  $sum \leftarrow sum + s_{d_{(i)}}$
  $tokens \leftarrow tokens + |d_{(i)}|$
  $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{d_{(i)}\}$
end for
return $\mathcal{D}'$

C. Dataset Construction Details

Table 4 reports per-subset statistics of our training corpus. Figures 3 and 4 show needle position and token length distributions for all four training subsets. All subsets exhibit uniform needle position coverage (≈33% each across front, middle, and end), confirming position-agnostic training.

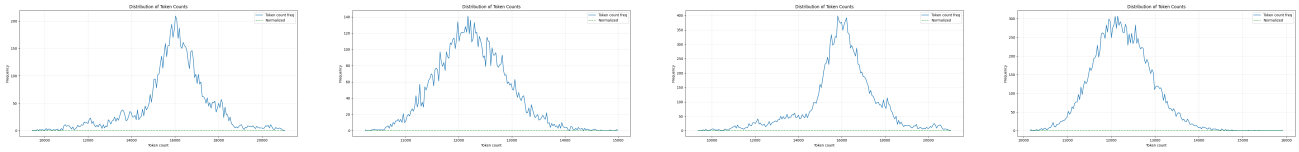
Table 4. Summary of training dataset statistics.

Dataset	Subsets	Samples	Needles	Ctx. len
16k	SQuAD	8,000	0 or 1	8k–48k
	HotpotQA	8,000	0 or 2	8k–16k
32k	SQuAD	16,000	0 or 1	8k–48k
	HotpotQA	16,000	0 or 2	8k–48k



(a) 8k-SQuAD *(b)* 8k-HotpotQA *(c)* 16k-SQuAD *(d)* 16k-HotpotQA

Figure 3. Needle position distributions across all training subsets.



(a) 8k-SQuAD *(b)* 8k-HotpotQA *(c)* 16k-SQuAD *(d)* 16k-HotpotQA

Figure 4. Token length distributions across all training subsets.

D. Evaluation Protocols

We follow a three-stage evaluation procedure. First, we train the compressor on different data subsets and evaluate compressor+generator performance on the QA1 and QA2 tasks of RULER to determine the best training configuration; qualitative analysis of these results motivated combining SQuAD and HotpotQA training data. Second, using DeepSeek-R1 as the target model on the Code-Debug task, we sweep context budget B and top-p threshold p on a held-out subset to determine the best inference hyperparameters. Finally, we fix these parameters and evaluate on the full Code-Debug benchmark with three target model families to assess cross-family generalization, and on RULER and LongBench v2 to characterize task-sensitivity beyond code reasoning.

E. Additional Ablations

E.1. Compressor Training Ablations

We conducted ablations throughout development to investigate the impact of training-data composition and hyperparameters on LongAttnComp. To select the final training configuration, we evaluated each candidate checkpoint on the `qa_1` and `qa_2` subtasks of RULER. These subtasks probe complementary skills: `qa_1` (SQuAD-derived) tests single-fact retrieval, while `qa_2` (HotpotQA-derived) requires multi-hop retrieval and reasoning, together covering the retrieval and reasoning axes central to this work. This choice also matches the QA-centric evaluation methodology of the original AttnComp paper.

Table 5. Training-data composition and schedule ablation on RULER QA subtasks (`qa_1`: SQuAD; `qa_2`: HotpotQA; Llama-3.1-8B-Instruct as target model, no top-p tuning).

Training configuration	qa_1	qa_2
Llama-3.1, no compression (baseline)	71.8	43.8
SQuAD only, const LR, 15 ep.	61.4	43.0
HotpotQA only, const LR, 15 ep.	26.6	51.8
Combined 16k, const LR, 15 ep.	53.8	53.4
Combined 16k, const LR, extended	59.4	51.8
Combined 32k, cos LR + dropout, 15 ep.	**68.2**	**58.2**
Combined 32k, cos LR + dropout, 18 ep.	65.8	57.1

As shown in Table 5, single-source training produces strong specialization: SQuAD-only and HotpotQA-only checkpoints each preserve performance on their source task but suffer significant drops on the other, suggesting that retrieval-biased training data trades off against reasoning capability and vice versa. Combining the two sources balances this: even at 16k samples, performance is balanced across both subtasks, and `qa_2` (the more reasoning-heavy subtask) exceeds the no-compression baseline. Doubling the corpus to 32k further improves both subtasks. While this is not a stress test of the cross-attention layer's training capacity, the fact that the layer absorbs four times the data used by the original AttnComp (Luo et al., 2025) without saturating suggests that the lightweight scoring layer can support substantially larger training corpora—a useful design observation for further scaling. On the hyperparameter axis, we obtain the best results with cosine LR decay, dropout, and 15 training epochs.; extending to 18 epochs produces a small drop, suggesting mild overfitting at fixed corpus size. These ablations both selected the configuration used throughout the main text and surfaced general training insights for single-cross-attention-layer compressors.

E.2. Top-p Ablation

Although the original AttnComp paper recommends $p = 0.95$ (Luo et al., 2025), our modified top-p algorithm and long-context inference setting differ enough that we re-verified this choice. We sweep p on a small Code-Debug validation subset, using DeepSeek-R1-0528 as the target model and the 16k-trained compressor. As shown in Table 6, the sweep is lightweight but suggestive: $p = 0.95$ remains the best choice in our setting, and we adopt it for all subsequent experiments.

Table 6. Top-p threshold ablation. Accuracy (%) on a 10-sample Code-Debug validation subset under our modified token-budget top-p algorithm (16k input budget, chunk size 256, 16k training corpus). $p = 0.95$ matches the original AttnComp default (Luo et al., 2025).

Top-p	Accuracy
0.70	50
0.80	60
0.90	50
0.95	**90**

F. Additional Experimental Results

F.1. LongBench v2 Results

We evaluate LongAttnComp on LongBench v2 (Bai et al., 2024) with DeepSeek-R1-0528 as the target model. Table 7 reports accuracy across the benchmark's standard difficulty (easy/hard) and length (short/medium/long) breakdowns. LongAttnComp underperforms both full-context baselines and Speculative Prefill, consistent with the training-data-composition gap discussed in §6.3 and §7: the synthetic NIAH-style needles in our current training corpus (SQuAD and HotpotQA) do not cover LongBench v2's mix of naturalistic multi-document retrieval and multi-hop reasoning over diverse domains. The gap is not narrowed by chunk size — two extreme settings (32 and 1024) yield nearly identical underperformance, in contrast to Code-Debug where chunk size has significant impact. This benchmark establishes a current limitation of LongAttnComp and motivates our planned second-stage training corpus targeting naturalistic, reasoning-heavy long-context tasks.

Table 7. LongBench v2 accuracy (%) with DeepSeek-R1-0528 as the target model, broken down by difficulty (easy/hard) and input length (short/medium/long). The 100k-truncated baseline reflects our cloud serving constraint (input truncated to 100k tokens to reserve response budget); the untruncated baseline is reported for reference. Speculative Prefill numbers are reproduced from a prior internal evaluation under the same deployment setting.

Method	Overall	Easy	Hard	Short	Medium	Long
Full context (untruncated)	56.7	59.4	55.0	66.7	50.9	51.4
Full context (100k truncation)	51.1	58.9	46.3	54.4	44.2	59.3
Speculative Prefill	46.3	49.0	44.7	56.7	40.5	40.7
LongAttnComp (chunk 32)	41.7	47.9	37.9	46.7	37.7	41.7
LongAttnComp (chunk 1024)	41.0	44.8	38.6	43.3	37.2	44.4