UniGist: Towards General and Hardware-aligned Sequence-level Long Context Compression

Chenlong Deng[1†], **Zhisong Zhang**[3*], **Kelong Mao**[1], **Shuaiyi Li**[2],
Tianqing Fang[2], **Hongming Zhang**[2], **Haitao Mi**[2], **Dong Yu**[2], **Zhicheng Dou**[1*]

[1]Renmin University of China [2]Tencent AI Lab [3]City University of Hong Kong
{dengchenlong, dou}@ruc.edu.cn, zhisong.zhang@cityu.edu.hk

Abstract

Large language models are increasingly capable of handling long-context inputs, but the memory overhead of key-value (KV) cache remains a major bottleneck for general-purpose deployment. While various compression strategies have been explored, sequence-level compression, which drops the full KV caches for certain tokens, is particularly challenging as it can lead to the loss of important contextual information. To address this, we introduce UniGist, a sequence-level long-context compression framework that efficiently preserves context information by replacing raw tokens with special compression tokens (gists) in a fine-grained manner. We adopt a chunk-free training strategy and design an efficient kernel with a gist shift trick, enabling optimized GPU training. Our scheme also supports flexible inference by allowing the actual removal of compressed tokens, resulting in real-time memory savings. Experiments across multiple long-context tasks demonstrate that UniGist significantly improves compression quality, with especially strong performance in detail-recalling tasks and long-range dependency modeling.

1 Introduction

As large language models (LLMs) are applied to more sophisticated and demanding applications, the ability to process long-range context has emerged as a fundamental requirement [29, 59]. A wide range of real-world applications, such as retrieval-augmented generation and long-form document understanding, require models to retain and reason over input sequences with extensive lengths [16, 60]. This has spurred growing interest in scaling up context windows from a few thousand tokens to hundreds of thousands or even millions, enabling models to incorporate more history, maintain coherence, and ground responses in broader contexts [23, 35, 46, 14].

Despite this progress, long-context modeling still remains a challenge due to the inherent compute and memory bottlenecks of the underlying Transformer architecture [37]. In particular, the cost of self-attention scales quadratically with sequence length, making both training and inference increasingly resource-intensive as the context grows [2, 50]. More critically, key-value (KV) caching, which is essential for efficient autoregressive decoding during inference, has also become a major source of memory consumption [18, 36]. In long-context settings, the memory required to store KV caches can even exceed that of the model's parameters. These limitations highlight the need for effective KV compression techniques that can reduce computational and memory overhead without sacrificing the model's ability to retain and reason over essential contextual information.

This work focuses on *sequence-level compression*, which directly reduces the number of token representations along the sequence dimension [43, 58]. A particularly promising and general sequence-

[†]This work was done during internship at Tencent AI Lab.
[*]Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

level strategy is the gist token-based approach [27, 17, 26, 30, 6, 54, 11], which segments long sequences into chunks and inserts learnable virtual tokens (i.e., gists) to represent the original tokens. In principle, this method has potential for dramatically reducing sequence length, as gist tokens can be much fewer than raw tokens.

However, we observe two major drawbacks in existing gist-based methods. First, they suffer from notable degradation when required to recall information from distant chunks. We hypothesize that this is due to the prevalent chunk-wise training scheme, which introduces a shortcut letting tokens to minimize loss by attending only to nearby uncompressed tokens within the same chunk [6, 54]. As a result, the model is less inclined to learn to integrate information across chunk boundaries, limiting its ability to reason over long-range dependencies. Second, the chunk-wise design suffers from fragmented memory usage and complex computational graphs, which hampers overall training efficiency. In this training scheme, chunks are processed in an iterative style and separate memory allocations are required for each chunk, leading to potential under-utilization of hardware resources.

In this work, we address the above challenge by introducing **UniGist**, a unified gist-based framework for context compression. Unlike prior approaches that rely on chunk-wise training, we remove this constraint and enable the model to learn compression more effectively over the entire sequence through a unified sparse gist layout. To further enhance the efficiency of this layout, we propose a gist shift trick, which is a hardware-aligned kernel design that transforms the irregular, vertically sparse attention pattern into a right-aligned block structure. This transformation aligns with the block-wise execution patterns of modern GPU architectures, significantly enhancing training throughput as well as supporting fast autoregressive inference. Experiments over a wide range of long-context tasks illustrate the effeciveness and efficiency of the proposed approach.

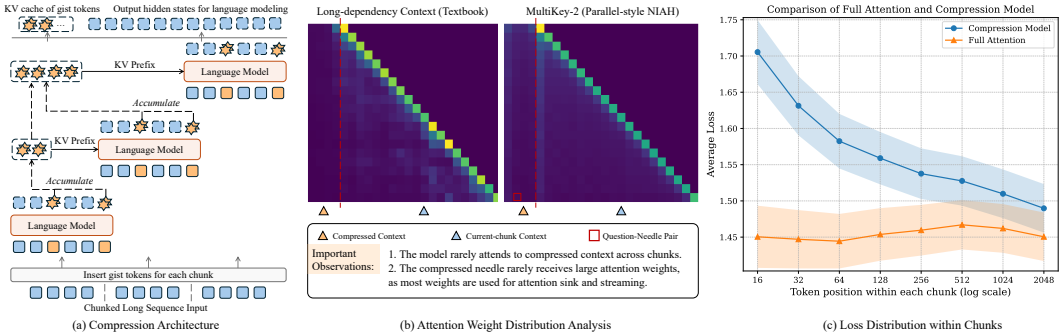
Our main contributions are as follows:

1) We propose UniGist, a sequence-level compression method with a unified sparse gist layout, enabling effective and efficient long context modeling without chunk-wise training.
2) We introduce the gist shift trick to align the proposed sparse attention layout with GPU-friendly block structures, significantly improving training throughput and supporting efficient inference.
3) With experiments on a variety of long-context tasks, we demonstrate that UniGist can provide improvements for context compression, with especially strong performance in detail-recalling tasks and long-range dependency modeling.

2 Related Work

KV Cache Compression. Transformer-based large language models mostly employ key-value caching to avoid repetitive attention computations during inference. However, as real-world applications demand ever-increasing context lengths, the associated memory and bandwidth overhead has quickly become a primary bottleneck for real-time systems. To tackle this, KV compression techniques are widely explored along four orthogonal dimensions (i.e, layer, head, token sequence, and numerical precision). *Layer-level* strategies typically skip certain transformer layers or share activations across layers [33, 34, 3, 32]. At the *head level*, multiple query heads can share a single key head, or the long-term cache of specific heads can be eliminated [1, 31]. Recent low-rank approaches further introduce a promising direction for this line [10, 56]. *Precision-focused* methods apply custom quantization and control the floating-point error they introduce to maintain the model's ability [24]. Moreover, some techniques within the layer, head, and precision dimensions demonstrate near-lossless performance on general long context tasks, making them broadly applicable in cutting-edge models.

On the other hand, *sequence-level* compression falls into two broad categories. **Token eviction** discards older token activations in real time once a budget is exceeded. For example, StreamingLLM [43] retains only the initial and recent tokens, which irreversibly lose the details in evicted tokens. To mitigate this, most of the following methods [58, 22, 47, 4, 12, 13] have to rely on hybrid eviction of layer or head level, or even resort to question heuristics rather than general compression. **Token merging** aggregates multiple tokens to reduce the overall cache size [57, 38, 40]. Models that introduce new "gist" tokens and adopt more training often achieve more stable compression [27, 6, 54]. Nonetheless, prior study shows that gist-based compression still struggles to retain highly complex details [11].



(a) Compression Architecture (b) Attention Weight Distribution Analysis (c) Loss Distribution within Chunks

Figure 1: Analysis of previous architectures. (a) Previous approaches adopt a chunk-wise strategy, which accumulates gist representations iteratively. (b) Our attention analysis reveals that tokens tend to ignore compressed contexts outside the current chunk. (c) Our token-position analysis indicates that tokens near the chunk ends may rely only on uncompressed tokens to minimize the target loss.

Our method addresses the key limitations of gist-based architectures, achieving high-quality and general-purpose compression using sequence-level alone.

Sparse Attention. Sparse attention has been widely adopted to mitigate the quadratic complexity of self-attention by restricting computation to a subset of token interactions. Empirical studies show that attention maps in language models are inherently sparse, enabling full attention can be approximated closely. Early methods focus on designing fixed sparsity patterns and training models to adapt to them [2, 51, 50]. With large language models scaling, researchers find that models trained with full attention could preserve performance by simply adopting dynamically searched sparsity patterns without extra tuning [20, 42, 41, 44, 53]. More recent works train dynamic attention patterns from scratch [15, 25, 49], in some cases even outperforming full attention. However, to ensure generality across diverse inputs, state-of-the-art sparse attention methods still require the cache for each token to maintain reachability. Our method leverages the intrinsic sparsity of the gist token–based architecture, eliminating the need for distant original token caches and achieving substantial acceleration during both training and inference.

3 Why Does Typical Gist-based Compression Fail in Details?

3.1 Preliminary

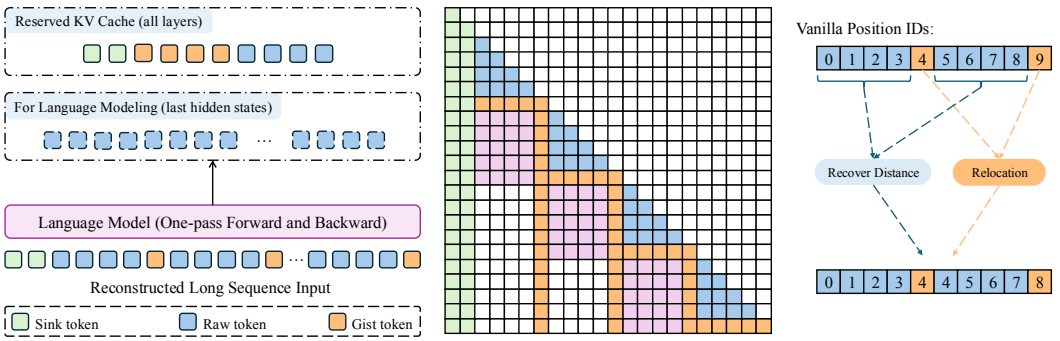
Previous empirical study [11] shows that the most effective variant of gist token-based compression inserts learnable gist tokens in an interleaved manner within each fixed-length chunk, and retains the KV cache of gist tokens across all transformer layers. This section introduces this variant as the representative architecture for our analysis.

Figure 1(a) illustrates the overall architecture. Given a long sequence $X = [x_1, x_2, \ldots, x_T]$, a chunk length L, and a compression ratio r, the sequence is first divided into $N = T/L$ fixed-length chunks. Within each chunk, one gist token is inserted after every r raw tokens to compress the sequence. As a result, each chunk contains L raw tokens and $n = L/r$ interleaved gist tokens.[1] The i-th chunk's content is given by:

$$\tilde{X}^{(i)} = [\underbrace{x_1^{(i)}, \ldots, x_r^{(i)}, g_1}_{\text{the 1-st gist unit}}, \ldots, \underbrace{x_{L-r+1}^{(i)}, \ldots, x_L^{(i)}, g_n}_{\text{the } n\text{-th gist unit}}], \quad r < L, \tag{1}$$

where g denotes the inserted gist tokens. Each group consisting of r raw tokens and the subsequent gist token is defined as a **gist unit**. For simplicity, all inserted gist tokens share the same embedding. For the i-th chunk, its input to the transformer blocks contains not only the current $\tilde{X}^{(i)}$, but also the accumulated KV cache from all preceding chunks' gist tokens $G_{<i}$ as the prefix, thereby enabling information flow across chunks. To maintain consistency with the inference stage, the same chunking

[1] Here we assumes that T can be divided by L and r for clarity.



(a) Model Architecture (b) Attention Pattern (c) Revised Position Encodings

Figure 2: Overall architecture of UniGist. (a) We adopt a chunk-free training strategy that allows one-pass sequence processing. (b) We design a gist-enhanced attention pattern to effectively aggregate information. (c) We revise positional encoding to maintain the relevant distances between raw tokens.

and iterative chunk-size forward strategy are used during training. The final training objective follows the standard cross-entropy language modeling loss over raw tokens.

3.2 Analysis

We analyze why previous gist-based models struggle to fully utilize compressed context from two perspectives by experimenting on a well-trained gist-based model:

Attention Patterns. We examine the attention patterns on two datasets with long-range dependencies (Figure 1(b)). In natural data such as textbooks, most tokens exhibit little attention across chunk boundaries. For structured tasks like RULER MultiKey-2 [19], which follow a parallel needle-in-a-haystack pattern, attention also remains mostly localized within the current chunk. In addition, many attention weights concentrate at the beginning of each chunk, even though this region may contain irrelevant key-value pairs. In contrast, the actual related piece of context receives little attention.

Token Positions. We perform continue-training with a well-trained compressed model on 32K-length sequences and measure the average loss at each relative position within chunks, excluding the first chunk that has no previous contexts. As shown in Figure 1(c), tokens at the beginning of each chunk consistently exhibit higher loss, while those closer to the end show loss patterns similar to full attention models. This suggests that tokens near the end can rely on nearby uncompressed tokens to minimize the language modeling loss. As a result, previous contexts may not be effectively used for compression learning, which limits the overall efficiency of training.

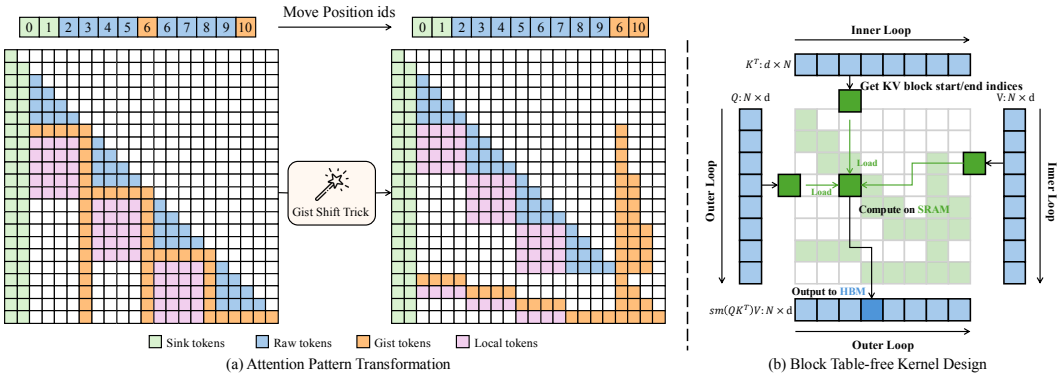
These observations suggest that chunk-wise training is a primary bottleneck limiting compression quality. Previous approaches rely on chunk-wise training to ensure consistency between training and inference, but this strategy prevents models from learning cross-chunk dependencies and limits compression performance, calling for better designs for training strategies.

4 Method

4.1 Unified Gist Context-based Language Modeling

As illustrated in Section 3.2, the chunk-wise training leads to an under-attention problem, that is, tokens in each block tend to ignore earlier compressed contexts. To address this, we eliminate the coarse-grained chunking strategy and introduce a unified attention pattern to enable the language model to learn more effective context compression. Figure 2 provides an overview of our strategy. Given a token sequence $[x_1, x_2, \ldots, x_T]$, we construct the augmented input with the following steps:

1) **Attention Sink Prefix.** We prepend a fixed number of s sink tokens $[s_1, s_2, \ldots, s_s]$ at the beginning of the entire sequence. They serve as fixed "attention sinks" [43] to prevent mode collapse that can occur when initial raw tokens are compressed and then removed.



(a) Attention Pattern Transformation (b) Block Table-free Kernel Design

Figure 3: Kernel design of our efficient sparse gist attention. (a) We adopt the gist shift trick by moving all gist tokens to the rightmost position of the sequence to facilitate GPU block processing. (b) With fixed compression rate, non-visible blocks can be skipped by pre-calculating block indexes.

2) **Gist Token Insertion.** We interleave learnable gist tokens into the token sequence with a compression ratio r (i.e., one gist token for every r raw tokens). We uniformly apply this insertion operation over all raw tokens. After this operation, the new sequence will be:

$$\begin{aligned} Z &= [s_1, \ldots, s_s, x_1, \ldots, x_r, g_1, x_{r+1}, \ldots, x_{2r}, g_2, \ldots, x_T, g_{T/r}] \\ &= [z_1, \ldots, z_{T'}], \quad T' = s + T + T/r \end{aligned} \tag{2}$$

3) **Position Encoding Alignment.** Inserting gist tokens shifts the position indices of later tokens, which changes the original distances between raw tokens and may harm language coherence. To mitigate this, we assign each gist token the same position id as the raw token right after it, so that all raw-token distances remain unchanged.

With this gist-augmented input, we define a unified sparse autoregressive attention pattern that is applied to *all tokens*. For each $z_t \in Z$, the set of its visible tokens is defined as:

$$\mathcal{A}(z_t) = (\mathcal{S} \cup G \cup \mathcal{W}_t) \cap \{z_j | j \leq t\}, \tag{3}$$

where the intersection with $\{z_j | j \leq t\}$ enforces the autoregressive constraint, restricting each token to attend only to itself or prior tokens. S and G are sink and gist tokens, respectively, which can provide conpressed contexts. The local window W_t consists of the current gist unit that contains z_t, as well as the previous k gist units (each containing r raw tokens followed by one gist token). This helps the model retain its basic language modeling ability with local contexts. The final training objective remains the standard autoregressive cross-entropy loss over raw tokens $x_t \in X \subset Z$.

4.2 Hardware-aligned Kernel Design

Sparse attention patterns in gist-token-based models are inherently efficient. Nevertheless, to fully realize such efficiency on GPU devices, we need suitable hardware-aligned kernels. Because of the sparsely inserted gist tokens, our attention patterns are incompatibility with FLASH_ATTN kernels [8]. Therefore, we design a custom kernel that supports efficient processing.

Since our attention pattern evenly distributes the gist tokens in a fine-grained way, conventional GPU kernels cannot fully exploit the advantages of sparse attention. The main reason is that GPU kernels are optimized for processing data in blocks (typically of size 64 or 128). In our setting (with a compression ratio of 4 or 8), gist tokens are always attended to and are present in every block. As a result, no blocks can be skipped during attention computation in training, even though the raw tokens in previous blocks outside the local window are not visible to later tokens and can be skipped.

We address this issue with the **gist shift trick**, which moves all gist tokens to the rightmost position of the sequence. This transformation converts the attention pattern into a dense, right-aligned scheme that aligns well with standard GPU block processing, as shown in Figure 3. In this way, all the gist tokens are gathered together, and each token will only need to attend to the compacted gist blocks in addition to the sink and local window blocks. This leads to a scheme that can fully exploit the sparsity benefits by skipping non-attended blocks. Note that with a fixed compression ratio, our attention

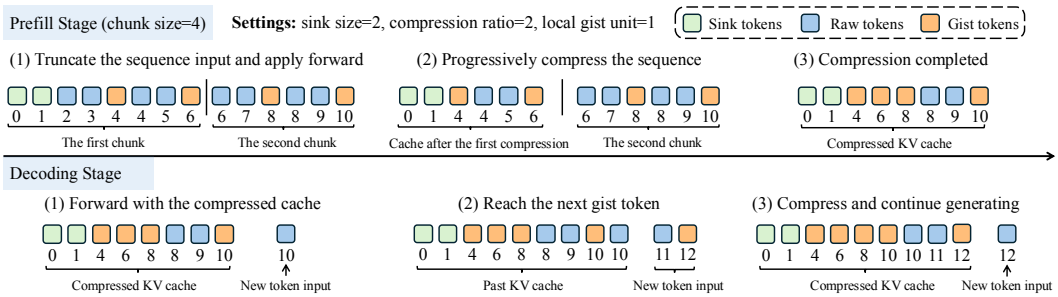


Figure 4: Illustration of flexible inference. Since the raw tokens outside the local window will never be attended to by future tokens, these invisible raw tokens can be safely discarded at each compression point. For prefilling, we can re-adopt the chunk-wise scheme and encode each chunk of the input with our efficent kernel, with nonvisible tokens dropped after each chunk. For decoding, nonvisible raw tokens can be discarded after each new gist is inserted.

layout is deterministic, and there is no need to build or store any block index tables. Instead, the indexes of the visible blocks for each token can be directly pre-calculated.

This trick brings a side effect that the original token indexes are distorted, which leads to the difficulty of calculation **in-block attention masks**. To mitigate this issue, we pass an auxiliary input of the original token indexes to the kernel. With this information, we can easily calculate the attention masks according to Equation 3.

4.3 One-pass Training and Flexible Inference

Training: One-pass with Full Sequence. With our unified attention pattern and custom kernel, we can process long sequences in a single pass without chunk-wise iterative processing during training. This eliminates memory fragmentation and reduces the complexity associated with the chunk-based scheme, leading to faster and more memory-efficient training.

Inference: Flexible Decoding with Low Memory Usage. For prefilling, which naturally follows the iterative manner, we can re-adopt the chunk-wise scheme to encode each chunk with our kernel to reduce peak memory usage.[2] The overall procedure is illustrated in Figure 4. During inference, each token only attends to tokens defined by the autoregressive pattern. Raw tokens beyond the local window are no longer required and can be dropped on the fly during generation. This also allows compatibility with efficient decoding techniques like FlashDecoding [9].

5 Experiments

5.1 Experimental Setup

Baselines. We compare UniGist primarily with methods that perform sequence-level context compression, as compression techniques along other dimensions are often orthogonal and not directly comparable. Our baselines include three categories of methods: (1) Vanilla Full Attention Models, including the original model and its supervised fine-tuned variant using the same data as ours to control for variables. (2) Training-free Context Compression, where we include StreamingLLM [43], SnapKV [22], PyramidKV [4], and AdaKV [12] (based on SnapKV) as popular eviction methods. (3) Gist Token-based Compression, where we evaluate AutoCompressors [6] and Activation Beacon [54] as state-of-the-art methods. Both use chunk-wise compression to enable general-purpose compression.

Implementation Details. We use Llama3.1-8B-Instruct and Llama-3.2-3B-Instruct as the base models to evaluate performance across different model sizes [35]. Continued pretraining is conducted on 16B tokens of 32K-length samples drawn from Prolong's [14] mixed dataset, followed by supervised tuning on 1B tokens of mixed instruction data. Cross-document masking is applied during training to block attention across document boundaries. All custom attention kernels are implemented in Triton. For all methods, we adopt a question-agnostic compression setup to ensure generality and

[2]The chunk size can be configured as any integer multiple of the compression ratio during prefilling.

Method	RAG	Rerank	LongQA	ICL	Synthetic	Summ.	Average
Llama-3.1-8B-Instruct							
Full Attention	74.5	55.1	43.5	81.8	99.3	28.9	63.9
Full Attention (FT)	72.7	48.4	43.8	79.5	97.4	26.8	61.4
StreamingLLM	58.5	28.4	33.1	36.0	18.6	12.2	31.1
SnapKV	60.3	11.8	40.4	29.6	19.8	12.6	29.1
PyramidKV	60.9	9.3	41.3	22.4	21.7	12.7	28.0
AdaKV	61.1	14.0	40.5	37.6	31.6	12.5	32.9
AutoCompressors	64.7	20.9	36.4	40.1	23.2	16.5	33.6
Activation Beacon	67.9	34.4	40.6	79.2	61.8	21.0	51.8
UniGist (Ours)	**71.3**	**45.5**	**45.5**	**85.8**	**91.3**	**22.9**	**60.4**
Llama-3.2-3B-Instruct							
Full Attention	68.6	17.5	38.6	79.6	78.9	28.2	51.9
Full Attention (FT)	66.3	16.9	39.1	77.1	85.6	27.7	52.1
StreamingLLM	52.3	10.9	13.2	5.8	15.1	11.5	18.1
SnapKV	52.8	3.0	29.4	8.6	16.2	12.3	20.4
PyramidKV	52.5	3.2	31.5	7.8	12.6	11.6	20.0
AdaKV	53.8	3.1	30.5	6.6	28.6	12.4	22.5
AutoCompressors	55.7	7.2	26.3	31.2	7.9	15.2	23.9
Activation Beacon	58.4	11.4	36.5	57.4	47.1	16.8	37.9
UniGist (Ours)	**63.6**	**15.6**	**41.7**	**73.8**	**82.3**	**21.6**	**47.8**

Table 1: Evaluation results of various context compression methods under two base models across long-context tasks. "FT" denotes to the fine-tuned full attention model. Bolded numbers indicate the best results among compression methods.

difficulty. Greedy decoding is used for all tasks. For UniGist, the sink size is set to 128, and the local window corresponds to 128 raw tokens. The main results reported use a compression ratio of 4. All training and inference experiments are conducted using the Huggingface framework. More details about data composition and training setup are introduced in Appendix A.

5.2 Main Results

Method	MMLU-Pro	GSM8K	HellaSwag
Full	47.7	83.6	60.2
Full-FT	47.1	84.3	59.8
Beacon	47.3	84.2	60.1
UniGist	47.6	83.9	60.1

Table 2: Performance on three widely-used short-context datasets. The "Full" and "FT" denote to full attention and fine-tuning, respectively.

Long Context Evaluation. We evaluate models' long-context understanding using the HELMET benchmark [48], which includes a broad range of long-context tasks, including retrieval-augmented generation (RAG), summarization, and synthetic recall. HELMET incorporates several representative datasets such as InfBench [55] and RULER [19], offering a comprehensive assessment of model performance across diverse long-context scenarios. For each task, we report the average score over its datasets. Detailed task configurations are provided in Appendix B.

Table 1 presents the overall evaluation results. We highlight three key observations: (1) **UniGist achieves the best performance among all compression methods across both model sizes.** On the synthetic tasks such as RULER MK-3, it is the only method that performs close to full attention. (2) **UniGist shows clear improvements over prior methods with similar architectures** (e.g., Beacon). This suggests that our unified attention pattern design plays a key role in helping the model read compressed context effectively. (3) **The effect of compression varies by task.** In tasks like RAG, many methods still perform reasonably well. In contrast, many-shot ICL is much more sensitive, especially for training-free approaches. This may be because compression disrupts the parallel structure required for in-context learning. Gist token-based methods show better robustness under these conditions.

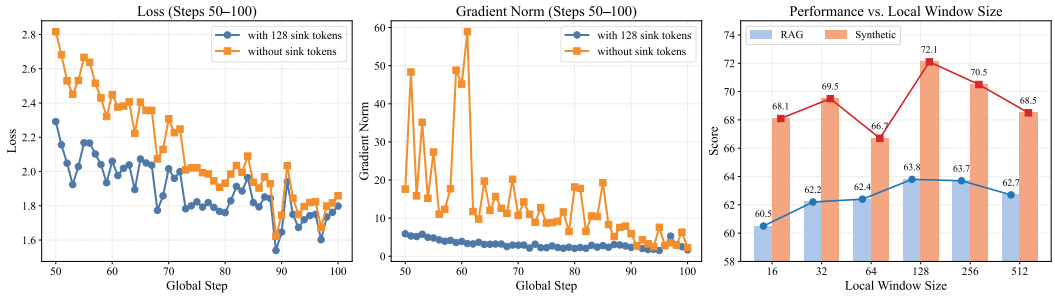


Figure 6: **Left & Middle:** Loss and gradient norm curves of Llama-3.1-8B-Instruct during training steps 50–100. Using 128 sink tokens leads to more stable training, with lower loss and smaller gradient fluctuations. **Right:** Performance of two long-context tasks of Llama-3.2-3B-Instruct after 4B-token continual pretraining and then supervised finetuning under different local window sizes.

Basic Short-context Capability. To assess whether context compression compromises the model's core capabilities, we further evaluate all methods on a set of standard short-context benchmarks, including MMLU-Pro [39] (knowledge and reasoning), GSM8K [7] (math), and HellaSwag [52] (commonsense inference) with 3-shot demonstration, as shown in Table 2. These tasks do not involve long contexts and thus serve as a proxy to measure whether compression-specific training introduces undesirable side effects on general performance. We find that all gist token-based methods maintain performance comparable to the full attention baseline across all tasks. Differences between methods are within the range of natural variance, and no consistent degradation is observed. This suggests that continued pretraining with the compression target does not impair the model's ability to perform basic tasks. The gains in long-context understanding come without sacrificing short-context capabilities.

Boundary Effect Test. As shown in Section 3.2, chunk-based training tends to introduce boundary effects, where a token's performance depends on its position within the chunk. We evaluate UniGist under the same setting and compare it with full attention. Figure 5 shows that UniGist maintains nearly uniform perplexity across positions, closely matching full attention. This confirms that UniGist avoids position-related bias caused by the chunk-wise training scheme.

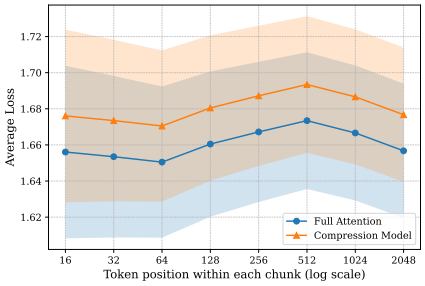


Figure 5: Boundary Effect Test. The compression model exhibits a similar trend to full attention, showing no degradation near chunk boundaries.

5.3 Ablation Study

The core mechanism of UniGist is the unified attention layout, which is designed for learning better compression. Its effectiveness is already evident in the main results through the performance comparison with Beacon's chunk-wise training. In this section, we focus on two additional components that contribute to UniGist's performance: the use of sink tokens and the choice of local window size.

Sink tokens for Stable Learning. To understand their impact, we compare training behaviors between models with and without sink tokens during the first 50 to 100 training steps. As shown in the left of Figure 6, the model with sink tokens achieves consistently lower loss and smoother gradient norms. In contrast, the model without sink tokens shows frequent gradient spikes, which slow down convergence. One possible reason is that, without sink tokens, the model fails to consistently attend to the initial tokens in the sequence because of progressive compression. This may break the autoregressive pattern and weaken early-stage learning signals.

Effect of Local Window Size. We further examine how the size of the local attention window affects the trade-off between language modeling and compression learning. A small window limits the model's access to raw tokens, forcing it to rely more heavily on gist tokens. This encourages the model to learn to extract compressed representations, but also increases the difficulty of language modeling due to reduced contextual continuity. In contrast, a large window gives the model easy

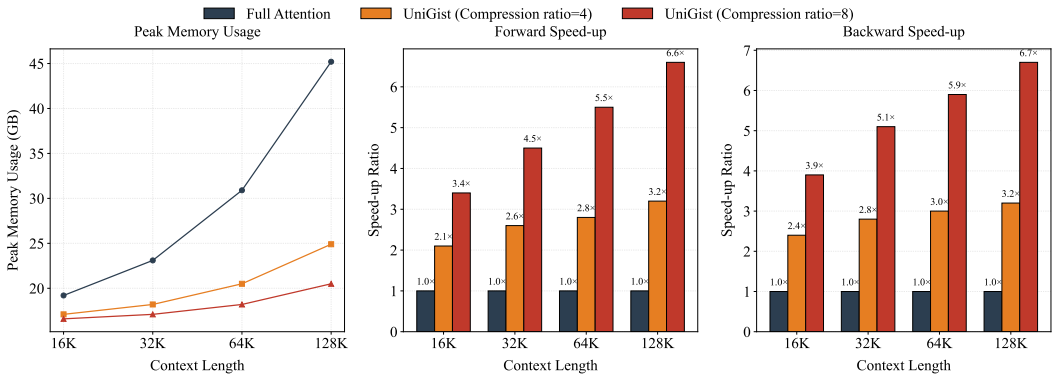


Figure 7: Comparison of efficiency between UniGist and standard full attention. **Left:** Peak memory usage increases steeply with context length for full attention, while UniGist maintains significantly lower memory consumption. **Middle & Right:** UniGist delivers substantial forward and backward attention speed-up, with higher gains under longer sequences and larger compression ratios.

access to raw tokens, which weakens the pressure to use gist tokens and may cause the model to bypass the compression path altogether. To quantify this trade-off, we evaluate UniGist with different window sizes on two long context tasks. As shown on the right of Figure 6, performance on both the RAG and synthetic recall tasks exhibits a non-monotonic pattern, initially improving and then declining. Notably, both tasks achieve peak performance when the window size is set to 128. This suggests that a moderate window provides sufficient local context for stable language modeling, while still encouraging the model to rely on gist tokens for global understanding.

5.4 Efficiency Comparison

UniGist is primarily designed to reduce the memory occupation of the KV cache, while also providing substantial speed-ups during both training and inference. We evaluate the quantitative benefits from two perspectives: memory usage and computation speed.

Memory Usage. Long-context modeling places heavy demands on memory resources, often becoming a bottleneck for deployment. We evaluate peak GPU memory usage under compression ratios of 4 and 8, comparing UniGist with full attention on the Llama-3.1-8B-Instruct model. As shown in the left of Figure 7, UniGist achieves a substantial reduction in memory usage in both settings, with more pronounced savings under ratio=8. These results demonstrate that UniGist can effectively alleviate memory bottlenecks while preserving modeling performance. The benefits grow with longer contexts, making UniGist a strong choice for scaling to ultra-long sequences.

Speed-up Evaluation. To isolate the speed-up benefit of the attention mechanism itself, we measure the forward and backward pass latency of a single attention layer. This avoids confounding factors from other components of the model. As shown in the right of Figure 7, the speed-up achieved by UniGist grows with context length and gradually approaches the theoretical upper bound, which equals the compression ratio (e.g., $4\times$ speed-up under compression ratio=4)[3]. This confirms that the sparse attention path introduced by UniGist leads to a real acceleration in real scenarios, especially in long sequences where the quadratic cost of full attention becomes prohibitive.

6 Conclusion

This work presents UniGist, a sequence-level compression framework for long-context language modeling without relying on hybrid strategies. By removing chunk-wise training and introducing a unified gist attention layout, UniGist enables effective learning over compressed sequences. We design a gist shift trick and a block-table-free sparse attention kernel that improve training efficiency and are fully reusable during inference. UniGist also supports flexible chunking at inference time for adaptive context processing. Experiments across a broad range of tasks demonstrate that UniGist

[3]We provide the original data and experimental details in Appendix C.

achieves solid performance across a range of tasks, and proves particularly effective at retaining details in long context scenarios. It also improves efficiency over full attention, with faster runtime and lower memory usage during inference.

7 Acknowledgments

This work was supported by Beijing Municipal Science and Technology Project No. Z231100010323009, National Natural Science Foundation of China No. 62272467, Beijing Natural Science Foundation No. L233008. The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics, 2023.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *CoRR*, abs/2004.05150, 2020.

[3] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[4] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069, 2024.

[5] 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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[6] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics, 2023.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

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

[9] Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flashdecoding. `https://crfm.stanford.edu/2023/10/12/flashdecoding.html`.

[10] Deepseek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434, 2024.

[11] Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Xinting Huang, Dong Yu, and Zhicheng Dou. A silver bullet or a compromise for full attention? A comprehensive study of gist token-based context compression. *CoRR*, abs/2412.17483, 2024.

[12] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. *CoRR*, abs/2407.11550, 2024.

[13] Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level KV cache compression method with integrated retrieval and reasoning. *CoRR*, abs/2410.19258, 2024.

[14] Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively), 2025.

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

[16] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997, 2023.

[17] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[18] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023.

[19] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[20] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir 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 Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[21] Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. BOOKSUM: A collection of datasets for long-form narrative summarization. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6536–6558. Association for Computational Linguistics, 2022.

[22] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: LLM knows what you are looking for before generation. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[23] Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, Zhejian Zhou, Ruijie Zhu, Junlan Feng, Yang Gao, Shizhu He, Zhoujun Li, Tianyu Liu, Fanyu Meng, Wenbo Su, Yingshui Tan, Zili Wang, Jian Yang, Wei Ye, Bo Zheng, Wangchunshu Zhou, Wenhao Huang, Sujian Li, and Zhaoxiang Zhang. A comprehensive survey on long context language modeling, 2025.

[24] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[25] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *CoRR*, abs/2502.13189, 2025.

[26] Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *CoRR*, abs/2305.16300, 2023.

[27] Jesse Mu, Xiang Li, and Noah D. Goodman. Learning to compress prompts with gist tokens. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[28] NVIDIA. Kvpress. `https://github.com/NVIDIA/kvpress`.

[29] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[30] Guanghui Qin, Corby Rosset, Ethan C. Chau, Nikhil Rao, and Benjamin Van Durme. Nugget 2d: Dynamic contextual compression for scaling decoder-only language models. *CoRR*, abs/2310.02409, 2023.

[31] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019.

[32] Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[33] Gemma 2 Team. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024.

[34] Gemma 3 Team. Gemma 3 technical report. *CoRR*, abs/2503.19786, 2025.

[35] Llama 3 Team. The llama 3 herd of models, 2024.

[36] MiniMax Team. Minimax-01: Scaling foundation models with lightning attention. *CoRR*, abs/2501.08313, 2025.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[38] Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, and Mi Zhang. D2O: dynamic discriminative operations for efficient generative inference of large language models. *CoRR*, abs/2406.13035, 2024.

[39] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[40] Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive KV cache merging for llms on long-context tasks. *CoRR*, abs/2407.08454, 2024.

[41] Wei Wu, Zhuoshi Pan, Chao Wang, Liyi Chen, Yunchu Bai, Kun Fu, Zheng Wang, and Hui Xiong. Tokenselect: Efficient long-context inference and length extrapolation for llms via dynamic token-level KV cache selection. *CoRR*, abs/2411.02886, 2024.

[42] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[43] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[44] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *CoRR*, abs/2503.16428, 2025.

[45] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

[46] 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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.

[47] Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramidinfer: Pyramid KV cache compression for high-throughput LLM inference. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3258–3270. Association for Computational Linguistics, 2024.

[48] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly, 2025.

[49] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *CoRR*, abs/2502.11089, 2025.

[50] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[51] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[52] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics, 2019.

[53] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattn: Accurate sparse attention accelerating any model inference. *CoRR*, abs/2502.18137, 2025.

[54] Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. *arXiv preprint arXiv:2401.03462*, 2024.

[55] Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. ∞bench: Extending long context evaluation beyond 100k tokens, 2024.

[56] Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Zhen Qin, Yang Yuan, Quanquan Gu, and Andrew Chi-Chih Yao. Tensor product attention is all you need. *CoRR*, abs/2501.06425, 2025.

[57] Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. Cam: Cache merging for memory-efficient llms inference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[58] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H2O: heavy-hitter oracle for efficient generative inference of large language models. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[59] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

[60] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. Large language models for information retrieval: A survey. *CoRR*, abs/2308.07107, 2023.

Category	Tasks	Metrics
RAG	NQ	SubEM
	TriviaQA	SubEM
	PopQA	SubEM
	HotpotQA	SumEM
Rerank	MS Marco	NDCG@5
Long-doc QA	∞Bench QA	ROUGE Recall
	∞Bench MC	Accuracy
Many-shot ICL	TREC Coarse	Accuracy
	TREC Fine	Accuracy
	NLU	Accuracy
	BANKING77	Accuracy
	CLINIC150	Accuracy
Synthetic recall	JSON KV	SubEM
	RULER MK Needle	SubEM
	RULER MK UUID	SubEM
	RULER MV	SubEM
Summ.	∞Bench Sum	ROUGE-Sum F1
	Multi-LexSum	ROUGE-Sum F1

Table 3: Data Composition and metrics of the used HELMET benchmark.

A Model and Training Details

A.1 Baseline Configuration

For training-free compression methods, we adopt the implementation based on the KVPress framework [28] to ensure consistency. For chunk-wise training methods, we divide sequences into 2K-length chunks and continue pretraining and fine-tuning on 32K-length samples. To ensure fairness, all trainable methods are fine-tuned with full-parameter updates. As for vanilla LLaMA models, given their extensive pretraining on approximately 15T tokens, we skip additional pretraining and apply supervised fine-tuning directly.

A.2 Hyper-parameters

For continued pretraining, we use a batch size of 2M tokens and set the learning rate to 1e-5. The learning rate is warmed up linearly from 0 over 256 steps and then decayed to 50% of its peak using cosine scheduling. The AdamW optimizer is used for our experiments. The compression ratio and local window size remain fixed throughout training. During fine-tuning, we retain most hyperparameters but reduce the warm-up steps to 128.

A.3 Training Data Processing

For continued pretraining, we use the 64K-length dataset from Prolong [14] and restructure it into 32K-length samples. Specifically, we split all 64K-length samples into two 32K segments. The original data includes two types of samples: those consisting of a single long document, and those formed by concatenating multiple shorter documents. For single-document samples, we apply no additional processing beyond the split. For multi-document samples, we pad each document individually so that its length is divisible by the compression ratio, ensuring compatibility with our attention pattern.

For supervised fine-tuning, we use the Magpie-Llama-3.1-Pro-MT-300K-Filtered dataset [45] as the main source. Then, we refer to the Beacon's [54] setup to further augment it with samples from LongAlpaca [5], BookSum [21], and 500 RULER-like synthetic data designed for long-context tasks. In total, the fine-tuning data comprises approximately 1B tokens.

Direction	Method	Ratio	16K	32K	64K	128K
Forward	Causal	-	13.8	52.7	206.9	863.9
	UniGist	4	6.45	20.5	72.9	271.8
	UniGist	8	4.10	11.6	37.3	130.9
Backward	Causal	-	57.4	225.6	871.5	3586.8
	UniGist	4	24.3	80.9	294.7	1116.8
	UniGist	8	14.9	44.5	147.9	538.9

Table 4: Forward and backward time (ms) under different context lengths. UniGist is evaluated at compression ratios (CR) of 4 and 8.

B Evaluation Details

Long Context Evaluation. We evaluate on six tasks from the HELMET benchmark. The datasets and metrics are listed in Table 3. We use a chat template for all prompts. For the LongQA task, we use a 2-shot demonstration to ensure an exact match in multi-choice questions. Since most models are trained with a 32K context length, we use the corresponding 32K configuration files. For tasks like Rerank, we append a brief reminder after the question to reinforce the expected output format and task objective.

C More Experiments

Speed-up Details We report the raw measurements of speed-up under a controlled setting with batch size 1, 32 attention heads, and a head dimension of 128. All methods are implemented with Triton to ensure a fair comparison. Each result is averaged over five runs, as shown in Table 4.

D Limitations

Model Scale and Context Length Our experiments are limited to models with at most 8B parameters and a context length of 32K due to computational constraints. Although this setup enables controlled and efficient experiments, it does not fully capture the potential of larger models to learn and benefit from compression. Understanding how model scale and context length influence compression quality remains an open question for future research.

Training Protocol In this work, we adapt existing language models to our attention pattern through continued pretraining. While we do not train from scratch, the UniGist architecture is fully compatible with end-to-end training. Training from scratch may allow the model to better internalize the structure and inductive biases of our compression pattern, without being influenced by pre-existing attention mechanisms. However, such training typically requires hundreds of billions of tokens to yield meaningful insights, which is beyond our current computational budget. We leave a comprehensive investigation of scratch training for future work, especially to understand how compression-aware architectures behave during early-stage pertaining.

E Ethical Discussion

UniGist aims to reduce the computational and memory overhead of large language models by introducing efficient context compression. This brings tangible benefits: it lowers the cost of inference, reduces carbon footprint, and facilitates the deployment of long-context models in resource-constrained environments. However, compression may inevitably alters model behavior. Compared to full attention baselines, compressed models may yield less accurate or misleading outputs in certain edge cases, especially when critical information is omitted or distorted during compression. This poses a risk in high-stakes applications where incorrect responses can have real-world consequences. We urge users to carefully evaluate reliability when applying UniGist in downstream tasks and to consider human oversight when appropriate.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim our contributions and scope in these two sections.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a limitation statement in Appendix D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details in Section 5.1, Appendix A and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: we plan to release our code in the future.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide details in Section 5.1, Appendix A and Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report the average scores for most of our experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We include this details in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We don't volate the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss these potential impacts in Appendix E.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We discuss these in Appendix E.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The assets are explicitly mentioned and respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: We don't publish new assets until submission, but we plan to release in accordance with regulations.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.