Fix the Structural Bottleneck: Context Compression via Explicit Information Transmission

Jiangnan Ye
King's College London, UK

Hanqi Yan*
King's College London, UK

Zhenyi Shen
King's College London, UK

Heng Chang
Unity China, China

Ye Mao
Imperial College London, UK

Yulan He*
King's College London, UK
The Alan Turing Institute, UK

Abstract

Long-context LLM agents often struggle with growing token, memory, and latency costs, making efficient context compression essential for practical deployment. Existing LLM-as-a-compressor methods remain noticeably inferior to using the full context. We find that this gap partly stems from their inability to preserve contextual information effectively. In this work, we revisit context compression from a structural perspective and identify two key bottlenecks in standard LLM-based compressors: limited coordination among compression tokens during information aggregation, and layerwise dilution that weakens useful signals from intermediate hidden states. To address these limitations, we propose **ComprExIT** (Context **Compr**ession via **Ex**plicit **I**nformation **T**ransmission), a new context compression framework based on explicit information transmission. ComprExIT adaptively aggregates features into anchors across frozen LLM layers, then allocates information from anchors to compression slots through a globally coordinated transport plan. Experiments on 12 datasets show that ComprExIT consistently outperforms strong soft-compression baselines, improving average F1 by up to 18.5%, while adding only $\sim$1% trainable parameters and achieving over 2$\times$ faster compression than the fastest baselines. Code is available at `https://github.com/Jiangnan0522/ComprExIT`.

1 Introduction

Large Language Models (LLMs) have become unprecedentedly powerful, and these capabilities come with ever-growing context lengths [Team et al., 2026a, Team, 2026, DeepSeek-AI, 2026]. Long contexts rapidly consume tokens, saturate the effective context window, and enlarge the KV cache, incurring substantial latency and memory overhead [Liu et al., 2025, Xiao et al., 2025]. These costs can ultimately limit the practical performance of LLMs on downstream tasks [Li et al., 2025a, Liu et al., 2024].

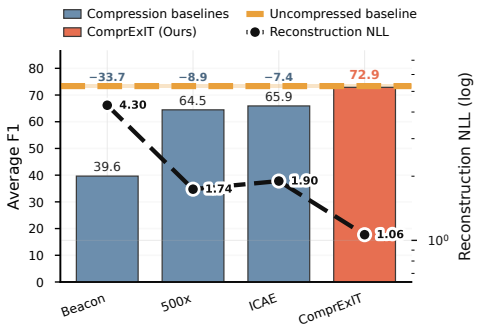


Figure 1: Performance (Average F1 over six datasets, Llama-3.2-3B) of existing context compression methods and their ability to preserve contextual information (reconstruction negative log-likelihood).

One emerging technique for addressing these problems is *Context Compression* (CC) [Li et al., 2025b], which compresses the original context into a fraction of its length. Current CC methods largely follow the LLM-as-a-compressor paradigm: they repurpose an LLM as a compressor by modifying its internal computation through continual training [Mu et al., 2023, Ge et al., 2024, Li et al., 2025c, Zhang et al., 2025,

*Corresponding authors: `hanqi.yan@kcl.ac.uk`, `yulan.he@kcl.ac.uk`.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Deng et al., 2025a]. Specifically, as shown in Figure 2-left, context tokens and a small set of gist tokens are fed into the LLM, which is trained to encode contextual information into these gist tokens. Although many design choices have been explored within this paradigm, existing methods still fall well short of the uncompressed baseline (Figure 1 bars). We hypothesize that a key reason is their limited ability to retain contextual information during compression. To investigate this, we use the gist tokens produced by different compressors as inputs and measure the decoder LLM's reconstruction negative log-likelihood (NLL) on the original context[2], which serves as a proxy for how much information the compressed representation preserves. As shown in Figure 1 (black line), methods with worse downstream performance (lower F1) also yield higher reconstruction NLL, suggesting that insufficient information retention is a critical bottleneck of existing approaches.

Since the current LLM-as-a-compressor paradigm is built on the standard LLM architecture, this gap in information retention calls for a structural re-examination. By tracing how information flows in the compressor LLM, we identify two limitations. In the *width dimension*, LLMs rely on self-attention to condense information from all context tokens into the gist tokens. Yet this process suffers from a **Lack of Allocation**: each gist token gathers information independently, without coordinating with the others about which parts of the context they represent. Consequently, multiple gist tokens may focus on the same regions while others remain under-covered, leading to information loss. In the *depth dimension*, LLMs propagate information in a layer-wise accumulative manner. As forward propagation proceeds, fine-grained evidence captured in earlier layers can become increasingly entangled with later, more abstract, generation-oriented representations [Zhang et al., 2024, Skean et al., 2025]. As a result, early-layer features become harder to recover from the final gist-token states, leading to **Information Dilution** [Team et al., 2026b].

The analysis above motivates a different formulation: instead of tuning an LLM into a compressor, we treat it as a *frozen* feature extractor and directly use its hidden states for compression. We propose ComprExIT (Context **Compr**ession via **Ex**plicit **I**nformation **T**ransmission), a new context compression framework that decomposes compression into two stages, each addressing one of the limitations above. The **Widthwise Stage** is built around an optimal-transport-based [Villani, 2008] transmission plan that globally allocates information across tokens and explicitly coordinates how contextual information is distributed to compression slots[3]. The **Depthwise Stage** establishes weighted shortcuts across layers, where fine-grained features extracted at different levels are directly aggregated, thereby mitigating information dilution across layers.

Our experiments across 12 datasets show that ComprExIT significantly outperforms state-of-the-art CC methods, with an average F1 improvement of 18.5%. It is also lightweight and efficient, adding only $\sim$1% parameters while compressing over $2\times$ faster than the fastest baselines. Further analysis shows that our key design components improve information preservation and thereby boost compression performance.

We summarize our contributions as follows:

- We analyze why existing LLM-as-a-compressor methods lose contextual information, and identify two key bottlenecks: *lack of allocation* in the width dimension and *information dilution* in the depth dimension.
- We propose ComprExIT, a lightweight context compression framework that treats a frozen LLM as a feature extractor and addresses these bottlenecks with a coordinated widthwise transmission plan and depthwise layer aggregation.
- We show that ComprExIT consistently outperforms prior context compression methods on 12 datasets, while remaining parameter-efficient and faster at compression.

2 Related Work

Existing context compression methods can be divided into two categories: token pruning and soft context compression, which operate in discrete and continuous spaces respectively.

Token pruning. Token pruning methods reduce context length by pruning input tokens according to estimated importance. Representative approaches include SelectiveContext [Li et al., 2023] and LLMLingua-style methods [Jiang et al., 2023, 2024], which score tokens/spans using a language model (e.g., via self-information or related salience measures) and retain only the most informative

[2]Experimental details are in Appendix A.1.1.

[3]Compression slots are the output representations that hold the compressed context. They play the role of gist tokens in LLM-as-a-compressor methods.

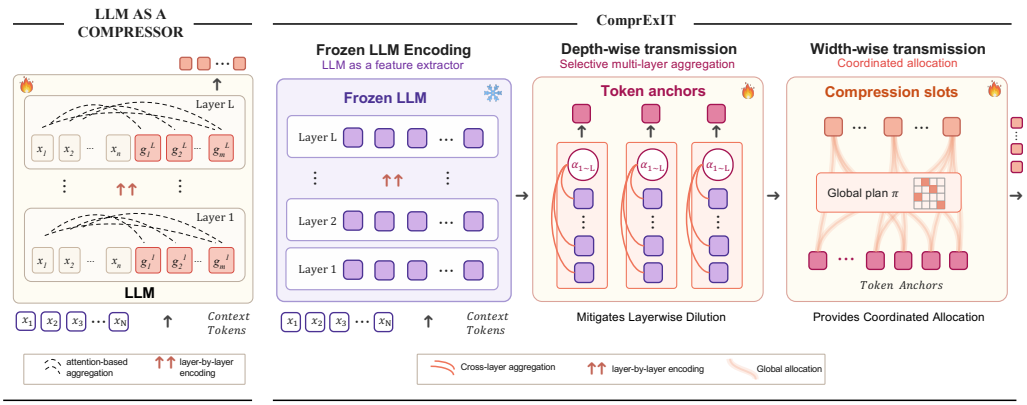


Figure 2: A comparison between existing LLM-as-a-compressor methods (left) and ComprExIT (right). **Existing methods** introduce gist tokens that are iteratively encoded across layers by the self-attention in the LLMs. **ComprExIT** instead leverages the layerwise hidden states of the context tokens encoded in a forward pass. The hidden states are selectively aggregated into token anchors, which are then transmitted to the compression tokens through a coordinated transmission plan.

parts of the context. LLMLingua-2 [Pan et al., 2024] instead distills a lightweight classifier from a stronger LLM (e.g., GPT-4) to decide which tokens to keep. EFPC [Cao et al., 2025] further unifies task-aware and task-agnostic compression within a single framework. Despite their efficiency gains, hard methods remain discrete and lossy, often facing bounded compression ratios and limited expressiveness compared with soft compression in continuous space.

Soft context compression. Soft context compression offers greater flexibility and expressiveness than discrete token removal at high compression ratios, and our method belongs to this category. Wingate et al. presented an early attempt to compress context tokens into a single vector, focusing on representation learning. AutoCompressor [Chevalier et al., 2023] builds LLMs into compressors inspired by the Recurrent Memory Transformer [Bulatov et al., 2022], accumulatively compressing contexts into summary tokens. Mu et al. introduce gist tokens as an information bottleneck by modifying attention masks, establishing the LLM-as-a-compressor paradigm. ICAE [Ge et al., 2024] further simplifies this paradigm into an encoder–decoder framework, which has since become a widely adopted paradigm. Building upon ICAE, 500× [Li et al., 2025c], Activation Beacon [Zhang et al., 2025], and UniGist [Deng et al., 2025a] pass cached key–value states to the decoder, together with carefully designed gist-token attention masks, to obtain more informative compressed representations. Deng et al. systematically study the effects of different design components within the encoder–decoder compression framework. EPL [Zhao et al., 2025] further improves compression performance by adjusting the positional encodings of gist tokens. xRAG [Cheng et al., 2024] explores converting representations from text embedding models into compression tokens. SAC [Liu et al., 2026] examines the necessity of the auto-encoding training objective, demonstrating that effective compression can be achieved using only a text completion objective. Tang et al. [2026] dynamically assign compression budget through coarse-to-fine compression. Different from the existing methods that use the internal self-attention mechanisms of LLMs to perform compression, our method adopts a fundamentally different paradigm by decoupling compression from the LLM architecture and formulating it as an explicit information transmission problem over frozen hidden states.

3 Preliminary

Context compression formulation. Context compression consists of a compressor and a decoder. Given a context sequence $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{I}^N$, the compressor produces a compact representation $Z = f_\phi(\boldsymbol{x}) \in \mathbb{R}^{K \times d}$ with $K \ll N$. We refer to the K vectors of Z as *compression slots*. The decoder then conditions on Z to generate outputs.

LLM-as-a-compressor. Most existing methods instantiate f_ϕ by turning an LLM into a compressor. Given a context sequence $\boldsymbol{x} = (x_1, \ldots, x_N)$ and K learnable compression embeddings $G = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_K) \in \mathbb{R}^{K \times d}$, often called gist tokens, the model feeds the concatenated hidden-state sequence $[\mathrm{Emb}(\boldsymbol{x}); G]$ into a Transformer:

$$H^{(0)} = [\mathrm{Emb}(\boldsymbol{x}); G] \in \mathbb{R}^{(N+K)\times d}, \qquad H^{(\ell)} = \mathrm{Block}_\ell\big(H^{(\ell-1)}\big), \quad \ell = 1, \ldots, L. \tag{1}$$

Here, $H^{(\ell)}$ contains hidden states for all positions in the concatenated sequence. The compressed representation is defined by the compression-token positions in the last-layer output:

$$H^{(L)} = \left[H^{(L)}_{\text{ctx}}; H^{(L)}_{G}\right], \qquad Z = H^{(L)}_{G}. \tag{2}$$

Thus, compression is constrained by the same mechanisms used in LLM computation: self-attention for widthwise aggregation and residual layer propagation for depthwise information transfer.

4 Limitations and Observations

Widthwise limitation: lack of allocation. At the ℓ-th layer, the attention of gist tokens over the context tokens and the gist tokens can be written as

$$A^{(\ell)} = \text{softmax}\left(\frac{Q^{\ell}_{G}\,[K^{\ell}_{X}; K^{\ell}_{G}]^{\top}}{\sqrt{d_h}} + M\right) \in \mathbb{R}^{K\times(N+K)}, \tag{3}$$

where M is the attention mask. Here, each row $A^{(\ell)}_{i,:}$ is normalized independently, meaning that each gist token decides where to attend on its own. As a result, standard self-attention provides no mechanism to allocate how gist tokens absorb information across the context: several tokens may concentrate on the same region, while other regions remain weakly covered. To test whether this happens in practice, we analyze the attention distributions of gist tokens during compression[4], which reveal where each token gathers information from. As shown in Figure 3, the attention distributions of different gist tokens are highly correlated, indicating that they often draw information from similar parts of the context. The full attention map from gist tokens to context tokens is shown in Appendix Figure 7a, where the bright and faint vertical bands mark context tokens that are attended to, or ignored, by many gist tokens together. Moreover, the aggregation matrix exhibits a steep singular-value decay, which points to low diversity in the collection behaviour. Together, these findings suggest that many gist tokens capture redundant rather than complementary content. We therefore attribute this pattern to a lack of allocation in self-attention: without explicit coordination, gist tokens cannot spread out to cover distinct context regions and instead collapse onto the same salient areas.

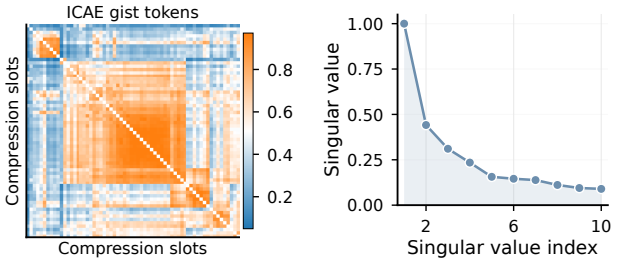


Figure 3: Two views of gist-token aggregation patterns in ICAE: Pearson correlation between gist-token attention distributions (the diagonal, which is always 1, is masked in white for readability) and the singular-value spectrum of the attention aggregation matrix.

Depthwise limitation: layerwise dilution. Let $G^{(\ell)} = H^{(\ell)}_{G}$ denote the compression-token states at layer ℓ. In most LLM-as-a-compressor methods, the decoder only receives the final representation $Z = G^{(L)}$. This design implicitly assumes that task-relevant information gathered at earlier layers is preserved through all subsequent updates. In practice, however, features that are salient at an intermediate layer can be attenuated as the network continues forwarding. To characterize how much usable information remains at depth ℓ, given the task query q and the target output y, we define

$$S_\ell = \max_{\psi\in\Psi} \text{Score}\left(p_\theta(y \mid q, \psi(G^{(\ell)}))\right), \tag{4}$$

where ψ is a lightweight readout or compression module, and $\text{Score}(\cdot)$ denotes the downstream task metric. If S_ℓ reaches its maximum at some intermediate layer and then drops as depth approaches L, this indicates a layerwise dilution bottleneck: information that is accessible earlier is not fully retained in the final state. Motivated by this hypothesis, we examine how information evolves along both the width and depth dimensions when an LLM is tuned as a compressor. We train a probing compressor that uses the hidden states from a single chosen LLM layer. The compressor consists of $4\times$ mean pooling followed by a 2-layer MLP. We train the models on SQuAD and HotpotQA and report the decoder F1. Each point in Figure 4 is thus an empirical estimate of S_ℓ in Equation (4), where we train the same probe ψ separately for each layer and use F1 as $\text{Score}(\cdot)$.

[4]Experiment specifications for examining the allocation patterns can be found in Appendix A.1.3.

As shown in Figure 4, hidden states from many layers provide useful compression features, but performance is not monotonic across layers, and the final layer performs the worst. This supports the view that useful features formed in earlier layers are gradually diluted rather than faithfully preserved through subsequent transformations. Moreover, contributions of layers are task-dependent: early and late layers contribute more to reasoning-intensive QA, whereas middle layers are more effective for extractive QA. These suggest that a strong compression method should both preserve and adaptively exploit features from different layers according to the context and task.

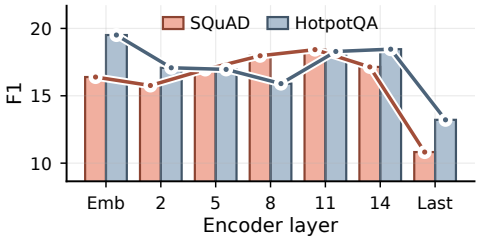


Figure 4: Performance of using each LLM layer for compression.

5 Method

We present our method in computational order: from depthwise to widthwise computation.

5.1 Depthwise Transmission to Token Anchors

We first address the depthwise limitation discussed above, namely the layerwise dilution bottleneck. In LLM-as-a-compressor methods, useful information must remain accessible after repeated layer updates before it reaches the final compression states, even though our preliminary analysis suggests that the accessible utility can peak at intermediate layers. Therefore, we directly read from hidden states across depths and construct a token anchor at each token position by adaptively selecting and aggregating multi-layer features. Given the hidden states $\{\boldsymbol{h}_t^{(\ell)}\}_{\ell=1}^{L}$ of token t, we first compute a lightweight structural context by mixing layer representations:

$$\boldsymbol{c}_t = \sum_{\ell=1}^{L} \pi_\ell \boldsymbol{h}_t^{(\ell)}, \qquad \sum_{\ell=1}^{L} \pi_\ell = 1, \tag{5}$$

where π_ℓ denotes learned layer-level structure priors, inspired by Denseformer [Pagliardini et al., 2024]. Conditioned on this context, we perform a token-wise gating attention over layers:

$$\tilde{\boldsymbol{h}}_t = \sum_{\ell=1}^{L} \mathrm{softmax}_\ell \left(\frac{\langle W_q \boldsymbol{c}_t,\ W_k \boldsymbol{h}_t^{(\ell)} + \boldsymbol{e}_\ell \rangle}{\tau} \right) W_v \boldsymbol{h}_t^{(\ell)}, \tag{6}$$

where W_q, W_k, and W_v are learnable query, key, and value projection matrices, $\boldsymbol{e}_\ell$ is a learnable layer embedding that encodes the layer index, and τ is the temperature. The resulting $\tilde{\boldsymbol{h}}_t$ serves as the token anchor. It can be viewed as a gated multi-layer readout from the frozen LLM to the compression interface, rather than an accumulated state that must survive until the last layer. In this way, features that are useful at intermediate depths remain directly accessible to the compressor, which mitigates the layerwise dilution bottleneck. Moreover, because the gating is conditioned on each token, ComprExIT can preserve different granularities of information for different tokens and tasks.

5.2 Widthwise Transmission to Compression Slots

Given the token anchors aggregated across layers, we next address the widthwise limitation discussed above, namely the lack of allocation among compression tokens. Instead of letting each slot independently retrieve information as in self-attention, we explicitly allocate information from N token anchors to K compression slots through a shared transmission plan. Formally, we seek $\Pi \in \mathbb{R}_+^{N \times K}$, where $\Pi_{t,k}$ denotes how much information token anchor t sends to compression slot k.

Utility matrix. To coordinate all possible sender-slot paths globally, we first construct a *utility matrix* $U_{t,k}$ that measures how beneficial it is to transmit information from anchor t to slot k. We treat each token anchor as a *sender*. For each compression slot, we build a corresponding *receiver* and assign it a local field of senders so that the initial slot semantics remain aligned with the original token order. Concretely, we uniformly partition the token-anchor sequence into local fields $\mathcal{F}_k$, average the anchors within each field to obtain the receiver representation, and then compute the utility matrix

using the cosine similarity between projected sender and receiver representations:

$$\boldsymbol{r}_k = \frac{1}{|\mathcal{F}_k|} \sum_{t \in \mathcal{F}_k} \tilde{\boldsymbol{h}}_t, \qquad U_{t,k} = \cos\Big(W_u \tilde{\boldsymbol{h}}_t, W_u \boldsymbol{r}_k\Big). \tag{7}$$

Here W_u is a learnable projection matrix that maps senders and receivers into a shared utility space. Higher utility indicates that sending information along this path is more likely to preserve useful content for the compressed representation.

Information capacity. Different token anchors need not contribute equally to compression. To reflect their varying contextual importance, we assign each sender a learnable information capacity so that less useful anchors can transmit less mass while salient anchors retain more influence. We predict this capacity with a linear layer followed by a softmax over token anchors:

$$\rho_t = \mathrm{softmax}_t\Big(W_\rho \tilde{\boldsymbol{h}}_t\Big), \qquad t = 1, \ldots, N. \tag{8}$$

Here W_ρ is a learnable scoring matrix that produces the sender-capacity logits.

For the receivers, we use a uniform capacity $\rho_k = \frac{1}{K}$. This gives every compression slot the same total budget, while still allowing important anchors to connect to non-local slots to better preserve long-range dependencies.

Transmission plan. Given the utility matrix and the sender/receiver capacities, we derive the transmission plan by solving the following optimization problem:

$$\min_{\Pi \geq 0} \quad \sum_{t=1}^{N} \sum_{k=1}^{K} \Pi_{t,k}\, C_{t,k} \quad \text{s.t.} \qquad \sum_{k=1}^{K} \Pi_{t,k} = \rho_t, \qquad \forall t, \; \sum_{t=1}^{N} \Pi_{t,k} = \rho_k, \qquad \forall k, \tag{9}$$

where the cost is defined as $C_{t,k} = 1 - U_{t,k}$. Unlike self-attention, whose weights are normalized independently for each slot, this objective couples all sender-slot assignments through shared marginal constraints. The resulting solution is exactly an optimal transport plan [Villani, 2008] between senders and receivers, which directly targets the lack-of-allocation issue identified in the preliminary analysis. We solve the problem with entropy regularization, yielding a strictly convex objective that can be optimized efficiently with the Sinkhorn algorithm [Sinkhorn and Knopp, 1967, Cuturi, 2013]. In practice, for stability and efficiency on long sequences, we run Sinkhorn on fixed-size segments of length T. We use a relatively large segment (e.g., $T = 128$) to preserve broad allocation flexibility while avoiding overly aggressive long-range assignments that may disrupt local semantic order. This produces a globally coordinated yet locally aware transmission plan.

Final representations. Given the transmission plan, each compression slot aggregates projected token-anchor features and is then aligned by a lightweight MLP:

$$\boldsymbol{z}_k = \mathrm{MLP}\left(\sum_{t=1}^{N} \Pi_{t,k}\, W_g \tilde{\boldsymbol{h}}_t\right), \qquad k = 1, \ldots, K. \tag{10}$$

Here W_g is a learnable projection matrix used before slot-wise aggregation. The resulting compressed sequence is denoted by $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K)$.

6 Experiments and Analysis

6.1 Experimental Setup

Implementation details. In the experiment, we use Llama-3.2-1B-Base and Llama-3.2-3B-Base [Grattafiori et al., 2024] as the base model. For the projection to the decoder's input space, we set the projection hidden size to 256. The base models are trained using BF16 precision. We employ the Sinkhorn algorithm with 30 iterations to approximately solve the entropy-regularized optimal transport problem.

Baselines. We compare with several soft state-of-the-art compression baselines: (1) **ICAE** [Ge et al., 2024]: a seminal LLM-as-a-compressor method that trains an LLM to encode compressed information into compression tokens. (2) **500×** [Li et al., 2025c]: introduces an extra design that passes the KV states of the compression tokens to the decoder. (3) **Activation Beacon** [Zhang et al., 2025]: introduces the interleaving placement of compression tokens. We also include three

Table 1: Experimental results on six QA benchmark datasets. Best among compression methods is **bold**. Our method is colored in . The prompt-tuned uncompressed baseline is colored in .

Methods	SQuAD		NewsQA		TriviaQA		SearchQA		HotpotQA		NQ		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Llama-3.2-1B														
Prompt tuning [w/ context]	71.89	81.09	30.72	45.76	61.04	66.75	64.03	70.64	53.60	69.49	53.03	66.59	55.72	66.72
Zero-shot [w/ context]	16.59	38.76	9.97	26.99	35.50	49.31	5.83	12.39	22.51	39.24	18.14	38.90	18.09	34.27
Zero-shot [w/o context]	2.80	8.83	0.97	3.11	15.63	21.60	4.04	6.64	2.76	7.17	3.37	8.43	4.93	9.29
ICAE [Ge et al., 2024]	36.84	50.21	20.20	33.73	57.59	63.54	67.40	74.43	40.67	57.04	42.26	58.00	44.16	56.16
500x [Li et al., 2025c]	4.65	14.02	1.85	5.27	26.09	32.66	17.44	25.59	5.56	13.44	5.58	13.89	10.20	17.48
Beacon [Zhang et al., 2025]	24.88	39.50	9.12	15.74	0.51	2.15	1.90	9.86	31.23	46.35	21.04	37.05	14.78	25.11
ComprExIT	**51.38**	**68.08**	**30.39**	**49.12**	**64.62**	**70.93**	**71.91**	**78.93**	**49.84**	**68.40**	**45.93**	**63.88**	**52.34**	**66.55**
Llama-3.2-3B														
Prompt tuning [w/ context]	80.39	88.41	35.23	50.98	71.23	76.30	72.04	78.42	55.31	72.65	59.46	73.32	62.28	73.35
Zero-shot [w/ context]	35.85	55.05	16.33	34.89	57.52	67.83	29.40	37.26	38.94	57.24	32.74	52.09	35.13	50.73
Zero-shot [w/o context]	9.14	17.21	2.40	5.78	46.20	52.06	25.43	30.35	8.63	16.03	10.31	18.47	17.02	23.32
ICAE [Ge et al., 2024]	48.51	62.53	27.02	43.04	68.62	74.13	76.43	82.89	47.92	66.16	50.87	66.75	53.23	65.92
500x [Li et al., 2025c]	52.08	65.56	25.45	40.99	65.77	71.59	69.26	76.28	47.28	65.71	51.32	66.60	51.86	64.46
Beacon [Zhang et al., 2025]	**59.81**	72.65	15.46	24.39	1.36	4.72	5.24	9.13	51.55	67.41	45.05	59.56	29.75	39.64
ComprExIT	59.26	**75.68**	**35.73**	**55.23**	**72.97**	**78.86**	**78.00**	**84.40**	**55.92**	**74.15**	**52.23**	**68.95**	**59.02**	**72.88**

non-compression baselines: (4) **Zero-shot [w/ context]** prompts the LLM with context, serving as an untrained baseline with zero information loss; (5) **Zero-shot [w/o context]** prompts the LLM without the context, serving as an untrained baseline with complete information loss; and (6) **Prompt tuning** [Lester et al., 2021] also prepends learnable tokens while retaining full access to the context, and therefore serves as a strong uncompressed baseline in our experiments.

Training and evaluation. Except for the base models evaluated under the inference-only setup, all baselines are trained under the same two-phase procedure: next-token prediction (NTP) on 1B tokens sampled from SlimPajama [Shen et al., 2023], followed by supervised fine-tuning (SFT) on MRQA [Fisch et al., 2019] with six in-domain and six out-of-domain question-answering datasets. Only the context is compressed and each context can be used for more than one question. We set the compression ratio to $\times 4$. All models use a context length of 512 tokens in both phases. We report EM and F1, and study additional settings in the ablations.

6.2 Main Results

Overall performance. Under the question-unaware training setup described above, Table 1 shows that ComprExIT consistently outperforms all compression baselines on the six in-domain QA benchmarks for both Llama-3.2-1B and Llama-3.2-3B. In average F1, it improves over the strongest compression baseline by 18.50% and 10.56%, respectively, while remaining close to the uncompressed prompt-tuning baseline (within 0.25% on 1B and 0.64% on 3B). Overall, these results show that ComprExIT performs better under the same compression budget.

Out-of-domain performance. Using the same training recipe and experimental setup, we further evaluate on six out-of-domain MRQA datasets in Table 10. ComprExIT again performs best overall, improving average F1 over the strongest compression baseline by 30.97% on Llama-3.2-1B and 16.00% on Llama-3.2-3B, with only a small gap to ICAE on RelationExtraction. Since RelationExtraction has very short contexts (30 tokens on average) and ICAE and 500x employ a fixed budget of 128 compression tokens, they are constrained by a much lower compression ratio on this dataset, thereby yielding better performance. We also evaluate on summarization, fact-checking, and in-context learning in Appendix A.4. ComprExIT outperforms the uncompressed baseline on XSum summarization and FEVER fact-checking while remaining competitive on SST-2 in-context learning. Together, these results suggest that ComprExIT generalizes well to various domains.

6.3 Ablation Studies

We examine our design choices under the same training and evaluation setup as before with Llama-3.2-1B. To isolate the effect of coordinated allocation, we replace the OT-based allocation module with a window attention mechanism that preserves locality but removes global coordination. As shown in Table 2, performance drops consistently across all datasets, indicating that global allocation is critical for effective compression. Next, we remove layer-wise aggregation and use features from only a single LLM layer for compression. Performance degrades under all single-layer settings, with

Table 2: Key design ablations for ComprExIT on Llama-3.2-1B (16 layers), using the main experiment's setup.

Ablation	Avg. EM	Avg. F1
ComprExIT	52.34	66.55
w/o alloc.	47.69 ↓4.65	61.94 ↓4.61
only emb. layer	28.65 ↓23.69	44.49 ↓22.06
only layer 5	45.13 ↓7.21	61.39 ↓5.16
only layer 9	49.34 ↓3.00	63.89 ↓2.66
only layer 14	44.18 ↓8.16	60.74 ↓5.81
only last layer	36.12 ↓16.22	49.35 ↓17.20

Table 3: Latency analysis results.

Method	Compress (s)	Decode (s)	E2E (s)
Context Length = 512			
Full Context	–	25.04	25.04
ICAE	0.43	10.16	10.59
ComprExIT	**0.18**	**9.83**	**10.01**
Context Length = 2048			
Full Context	–	101.09	101.09
ICAE	1.93	27.37	29.30
ComprExIT	**0.84**	**27.18**	**28.02**

the middle layer (Layer 9) incurring the smallest drop. This result further validates that the design in ComprExIT that selectively aggregates features from all layers benefits compression.

We further vary three hyperparameters of the allocation module: the number of Sinkhorn iterations, the segment length T, and the projection dimension of the utility space. Performance saturates at around our default of 30 iterations and stays stable across projection dimensions and a larger segment, with clear drops only under very few iterations or a short segment. This shows that ComprExIT is robust to moderate changes of the OT hyperparameters (see Appendix A.6 for details).

6.4 Latency Analysis

Table 3 reports the latency breakdown under two context lengths[5]. Compared with ICAE, ComprExIT is more than $2\times$ faster overall in the compression stage across settings. In both settings, the compressed methods are also substantially faster end-to-end than decoding the full context, showing the efficiency gain brought by context compression. These results suggest that ComprExIT forms compressed representations more efficiently, yielding a stronger efficiency-quality trade-off. This compression time already includes the Sinkhorn step, which alone takes only about 5 ms per sample under our default setting and thus adds little overhead (Appendix A.1.2).

6.5 Memory Analysis

Since ComprExIT aggregates features across layers, it keeps the hidden states of multiple layers during compression, which leads to a higher peak GPU memory than ICAE at long contexts (32.57 GB vs. 23.44 GB at 32k tokens). This overhead can be reduced by windowed compression, a common strategy for long contexts that compresses the context window by window and keeps only the hidden states of the current window. It lowers the peak memory of ComprExIT to 16.83 GB, below that of ICAE, with the same compression time. Since we only measure the efficiency of this windowed variant, we leave training it to future work (Appendix A.7).

6.6 Scalability Analysis

We examine the scalability of our method from three dimensions: higher compression ratios, larger models and longer context.

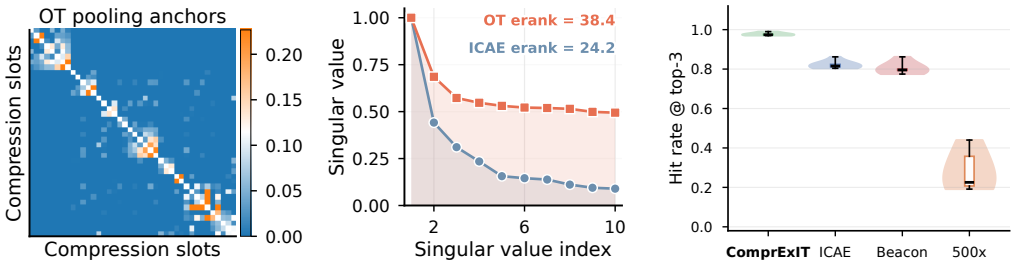
Higher compression ratios. Under the same setup, we train the Llama-3.2-1B with compression ratios of $\times 8$ and $\times 16$ and report the results in Table 4. ComprExIT outperforms the baseline at both $8\times$ (+17.2% Avg. F1) and $16\times$ (+7.6%). Notably, ComprExIT at $8\times$ compression (58.66) surpasses ICAE at $4\times$ compression (56.16; Table 1). These results show that ComprExIT exhibits consistently strong performance under higher compression ratios.

Larger model sizes. To examine the scalability of ComprExIT on larger models, we train the methods with Llama-3.1-8B under the same setup and report the performance in Table 4. ComprExIT continues to outperform ICAE by a clear margin and even slightly

Table 4: Scalability analysis experiments for ComprExIT on higher compression ratios, larger models, and longer context.

Analysis	Setting	Avg. EM	Avg. F1
Compression ratio	$\times 8$ - ICAE	38.72	50.07
	$\times 8$ - ComprExIT	**44.83**	**58.66**
	$\times 16$ - ICAE	37.23	48.82
	$\times 16$ - ComprExIT	**40.01**	**52.53**
Model size	8B - Prompt tuning	**65.03**	75.92
	8B - ICAE	59.24	70.41
	8B - ComprExIT	63.25	**76.48**
Context length	8k - Prompt tuning	27.43	37.27
	8k - ICAE	21.99	30.00
	8k - ComprExIT	**32.46**	**43.97**

[5]Check Appendix A.1.2 for details of the latency analysis experiment.



(a) Pearson correlation. (b) Singular-value spectrum. (c) Entity coverage.

Figure 5: Information allocation analysis. **(a)** Pearson correlation (diagonal masked) and **(b)** singular-value spectrum of allocation patterns. **(c)** Question-relevant entity recall on HotpotQA by TF-IDF rank. Top-10 entities are selected per context, and violin plot shows retrieval consistency across examples.

surpasses the uncompressed prompt-tuning baseline in F1 (76.48 vs. 75.92), showing that its advantage persists at larger model scale.

Longer context. We train the Llama-3.2-1B model on three long-context QA datasets, TriviaQA [Joshi et al., 2017], QuALITY [Pang et al., 2022], and NaturalQuestions [Kwiatkowski et al., 2019], with contexts truncated to a maximum length of 8k. The results are shown in Table 4. On this setting, ComprExIT improves over the uncompressed prompt-tuning baseline by 18.0% in F1 and outperforms ICAE by a large margin. This suggests that ComprExIT can preserve task-relevant content under more challenging long-context compression settings.

6.7 Further Analysis

6.7.1 Information Allocation

Allocation patterns. To further understand the impact cast by coordinated allocation, we analyze the aggregation behaviors of compression tokens[6]. In Figure 5a, compared with the aggregation pattern of ICAE in Figure 3, the consistently lower inter-slot correlations of ComprExIT indicate that different compression tokens absorb complementary information rather than collapsing into redundancy. In Figure 5b, this same behavior is reflected in the singular-value spectrum: compared with ICAE, ComprExIT exhibits a slower decay and a higher effective rank, showing that its allocation patterns span a richer and less redundant subspace. Taken together, the two views suggest that without global allocation, compression tokens tend to become correlated and low-rank, whereas ComprExIT encourages a more diverse and information-efficient distribution of content across slots.

Coverage of entities. We further examine whether coordinated allocation helps preserve important entities during compression. We extract named entities from each context and rank them by TF-IDF score. For each gist token, we identify the top-3 entities it attends to most and take their union within the context. We then compute the coverage of the ten most question-relevant entities by the gist tokens during compression, as shown in Figure 5c. Note that TF-IDF is used here only as a selection heuristic, and its notion of relevance is limited, since entities that are less relevant to one question may still be important for others. ComprExIT achieves the highest average coverage rate, nearly 1.0. More importantly, its thin violin shape indicates that it consistently covers relevant entities across examples, whereas the larger variance of the baselines suggests that they allocate attention more unevenly and sometimes miss important entities in certain contexts. These results indicate that a well-coordinated allocation is better able to preserve critical contextual information during compression.

6.7.2 Layer Selection

General layer preference. Figure 6-left shows that the gating mass concentrates on early and middle layers, with a clear emphasis before layer 10 and consistently low weights on later layers. This suggests that effective compression mainly relies on representations that still preserve lexical and contextual information, while deeper layers are more specialized for generation and therefore less suitable for forming compact, decoder-friendly representations.

Token-wise layer preference. The right panel shows a clear dependence on token type. Entity tokens, common nouns, adjectives, and numbers tend to receive higher weights from middle layers, where contextual, relational, and attribute-level information is more salient. In contrast, other tokens are

[6]Experiment specifications for examining the allocation patterns can be found in Appendix A.1.3.

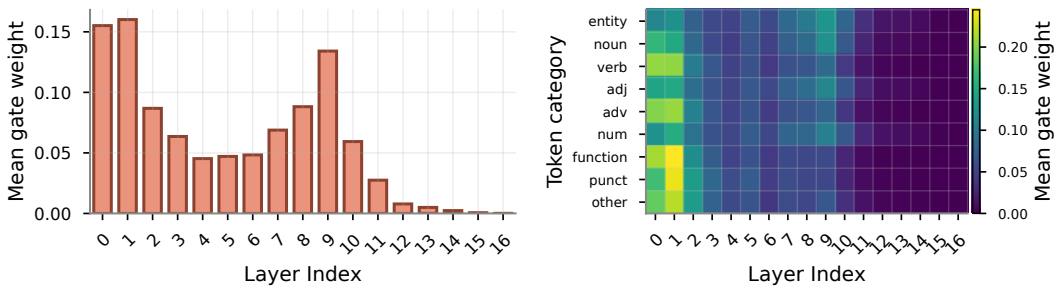


Figure 6: Layer-selection analysis in ComprExIT. *Left*: depth-wise gating weights across layers. *Right*: position–layer heatmap showing token-wise layer preferences.

more often routed to the initial layers, indicating that shallow lexical features are usually sufficient for them. Overall, these patterns suggest that ComprExIT adapts layer selection to token type, reserving richer mid-layer features for informative content words and using earlier-layer features for less informative tokens.

6.7.3 Dependence on the Decoder

Since the compressors are trained to produce inputs for the decoder LLM, one may question whether a higher F1 comes from compressed representations that are easier for this particular decoder to read, rather than from representations that retain more information about the context. To answer this question, we follow Badger [2026] and probe the compressed representations with a decoder that is *independent of the decoder LLM*. Specifically, we keep each trained compressor frozen and train a randomly initialized 8-layer decoder (158M parameters) from scratch to reconstruct the context from its compressed representations[7]. We report the reconstruction NLL and the entropy ratio H_r [Badger, 2026] on held-out contexts, where $H_r = 1$ means the context is fully recovered and $H_r = 0$ means the prediction is no better than a uniform guess over the vocabulary. As shown in Table 5, the probing decoder recovers much more of the context from the representations of ComprExIT than from those of ICAE and 500x. We take this as evidence that the compressed representations of ComprExIT preserve more information about the context, and that its advantage does not come only from representations that better suit the decoder LLM.

Table 5: Context reconstruction by a decoder trained from scratch on the frozen compressed representations (Llama-3.2-3B, $4\times$ compression).

Method	**NLL** $\downarrow$	$\boldsymbol{H_r}$ $\uparrow$
ICAE	4.294	0.635
500x	4.176	0.645
ComprExIT	**0.639**	**0.946**

7 Conclusion

This work presents ComprExIT, a new paradigm for soft context compression that formulates compression as explicit information transmission over frozen LLM hidden states, mitigating the limitations of layerwise dilution and lack of allocation. Experiments show that it consistently outperforms prior compression methods. More broadly, the proposed paradigm provides a more flexible design space beyond standard attention-based formulations. Therefore, we hope that this work can inspire future research to explore more design choices in this new paradigm and further advance context compression research.

Limitations. First, due to limited computational resources, we do not test ComprExIT on larger backbones (e.g., 32B or 70B models), and we evaluate task performance only up to 8k-token contexts, measuring efficiency alone at 16k and 32k. Second, ComprExIT keeps the hidden states of multiple layers during compression, so its peak memory is higher than that of ICAE unless the context is compressed window by window. Third, our experiments focus on text-only language models. Extending the framework to vision-language models, which also accept tokenized inputs, is left to future work.

[7] Experimental details are in Appendix A.1.4.

Acknowledgments and Disclosure of Funding

This work was supported in part by the UK Engineering and Physical Sciences Research Council through a Turing AI Fellowship (grant no. EP/V020579/1, EP/V020579/2) and the Prosperity Partnership scheme (grant no. UKRI566).

The authors acknowledge use of King's Computational Research, Engineering and Technology Environment (CREATE) at King's College London [King's College London, 2026].

The authors acknowledge the use of resources provided by the Dawn National AI Research Resource (AIRR). Dawn is operated by the University of Cambridge and is funded by the UK Government's Department for Science, Innovation and Technology (DSIT) via UK Research and Innovation the Science and Technology Facilities Council [ST/Z000890/1], Dell Technologies and Intel.

The authors declare no competing interests.

References

Benjamin L. Badger. Language model memory and memory models for language. *arXiv preprint arXiv:2602.13466*, 2026.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Yun-Hao Cao, Yangsong Wang, Shuzheng Hao, Zhenxing Li, Chengjun Zhan, Sichao Liu, and Yi-Qi Hu. EFPC: Towards efficient and flexible prompt compression. *arXiv preprint arXiv:2503.07956*, 2025.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6pTlXqrO0p.

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*, pages 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232/.

Philippe Clement and Wolfgang Desch. An elementary proof of the triangle inequality for the wasserstein metric. *Proceedings of the American Mathematical Society*, 136(1):333–339, 2008.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Chenlong Deng, Zhisong Zhang, Kelong Mao, Shuaiyi Li, Tianqing Fang, Hongming Zhang, Haitao Mi, Dong Yu, and Zhicheng Dou. Unigist: Towards general and hardware-aligned sequence-level long context compression. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=1C4mXyh31p.

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. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4861–4879, Vienna, Austria, July 2025b. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.241. URL https://aclanthology.org/2025.acl-long.241/.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen, editors, *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5801. URL https://aclanthology.org/D19-5801/.

Tao Ge, Hu Jing, 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*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

King's College London. King's Computational Research, Engineering and Technology Environment (CREATE). https://doi.org/10.18742/rnvf-m076, 2026. Retrieved June 18, 2026.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context LLMs struggle with long in-context learning. *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL https://openreview.net/forum?id=Cw2xlg0e46.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.391. URL https://aclanthology.org/2023.emnlp-main.391/.

Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. Prompt compression for large language models: A survey. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7182–7195, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. URL https://aclanthology.org/2025.naacl-long.368/.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xCompressor: Generalized prompt compression for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25081–25091, Vienna, Austria, July 2025c. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1219. URL https://aclanthology.org/2025.acl-long.1219/.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*, 2025.

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

Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and JingBo Zhu. Autoencoding-free context compression for LLMs via contextual semantic anchors. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8Pi6Du0n7F.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=2DtxPCL3T5.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206/.

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=kMnoh7CXrq.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. QuALITY: Question answering with long input texts, yes! In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.391. URL https://aclanthology.org/2022.naacl-main.391/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, et al. SlimPajama-DC: Understanding data combinations for LLM training. *arXiv preprint arXiv:2309.10818*, 2023.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=WGXb7UdvTX.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Jiwei Tang, Shilei Liu, Zhicheng Zhang, Yujin Yuan, Libin Zheng, Wenbo Su, and Bo Zheng. COMI: Coarse-to-fine context compression via marginal information gain. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OGDIXDfaN4.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026a.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026b.

Qwen Team. Qwen3.5-Omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2008.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5621–5634, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.412. URL https://aclanthology.org/2022.findings-emnlp.412/.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, junxian guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context LLM inference with retrieval and streaming heads. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cFu7ze7xUm.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1eQT9OzfNQ.

Runsong Zhao, Xin Liu, Xinyu Liu, Pengcheng Huang, Chunyang Xiao, Tong Xiao, and JingBo Zhu. Position IDs matter: An enhanced position layout for efficient context compression in large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 17715–17734, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.962. URL https://aclanthology.org/2025.findings-emnlp.962/.

A Appendix

A.1 Experiment Specification

We present more details of the experiments conducted in our paper.

A.1.1 Measuring Information Loss during Compression

For each method and each compressor, we extract the gist hidden states at layer ℓ, apply the method's own final projection where one exists, and feed them to the frozen decoder as the prefix of the sequence $[\text{BOS}] \,\|\, \mathbf{g}^{(\ell)} \,\|\, \text{Embed}(\mathbf{p}) \,\|\, \text{Embed}(\mathbf{x})$, where $\mathbf{p}$ is the fixed prompt `"Please reproduce the previous passage exactly."` and $\mathbf{x}$ is the original context. We calculate the standard causal-LM cross-entropy loss on the context tokens only. We then aggregate token-weighted across the dataset, $\widehat{\mathcal{L}}_\ell = \sum_i \text{loss}_i \,/\, \sum_i n_i$, where n_i is the number of non-padded target tokens in example i. We compute this loss on each of the six in-domain datasets (Appendix A.2) and report its average over these datasets in Figure 1, together with the average F1 on the same datasets. We use the compression models trained with Llama-3.2-3B in this experiment.

A.1.2 Latency Analysis

We benchmark each method on synthetic batches with fixed context lengths of 512 and 2048, a batch size of 8, a generation length of 128 tokens, and a compression ratio of 4. We use NVIDIA L40S ADA GPU, with 48GB memory. For each sequence, we run 50 untimed warm-up iterations followed by 200 timed iterations, and report the mean. We separately time the *compression* stage (the encoder forward pass that produces the gist) and the *decoding* stage (autoregressive generation of 128 tokens), and report both along with their sum (E2E).

Latency of the Sinkhorn step. The compression time above already includes the Sinkhorn step of the allocation module. To further measure its cost, we time the Sinkhorn step alone under different segment lengths T and numbers of iterations, using Llama-3.2-1B with a context length of 512 and a batch size of 64 on a single NVIDIA A100 80GB PCIe GPU. As shown in Table 6, it takes about 5 ms per sample under our default setting ($T = 128$ and 30 iterations). The time grows only mildly with more iterations or a larger segment, and stays below 14 ms even with $T = 512$ and 100 iterations, since the Sinkhorn step can be computed in parallel.

Table 6: Per-sample time (ms) of the Sinkhorn step with a batch size of 64. Our default setting is underlined.

	Sinkhorn iterations			
Segment T	10	30	50	100
64	4.83	5.18	5.52	6.37
128	4.81	<u>5.45</u>	6.10	7.69
512	5.46	7.28	9.12	13.65

A.1.3 Gist Token Allocation Patterns

We probe whether the K compressed slots of each method actually carry distinct content, or whether they collapse onto similar input regions. We run our experiments on the SQuAD [Rajpurkar et al., 2016] dataset with 1000 random samples. For ICAE we take the compressor's last layer attention map of the gist tokens and average over the attention heads. For ComprExIT, we use the captured transport plan as the allocation scores. In both cases this yields a $[K, N]$ matrix A whose rows are the per-slot distributions over input tokens. The *similarity heatmap* is the $K \times K$ Pearson-correlation matrix between these rows, $\rho(a_i, a_j) = (a_i - \bar{a}_i)^\top (a_j - \bar{a}_j) / (\|a_i - \bar{a}_i\|_2 \|a_j - \bar{a}_j\|_2)$, reordered hierarchically for readability. Its diagonal entries are always 1 and carry no information, so we mask them (shown in white) to keep them from setting the top of the color scale, which would make the lower off-diagonal correlations hard to tell apart. The *attention spectrum* is the sequence of singular values of A, plus the effective rank $\text{erank}(A) = \exp\big(-\sum_i \tilde{\sigma}_i \log \tilde{\sigma}_i\big)$ with $\tilde{\sigma}_i = \sigma_i / \sum_j \sigma_j$. A high effective rank and a near-diagonal similarity heatmap indicate that the K compressed slots cover the input non-redundantly.

A.1.4 Probing with a Decoder Trained from Scratch

We follow the probing setup of Badger [2026]. For each method, we take the compressor trained with Llama-3.2-3B at a compression ratio of $4\times$ in the main experiment, keep it frozen, and use the last-layer hidden states of its compression tokens as the compressed representations. For 500x, we also use these hidden states rather than the KV states that it passes to the decoder LLM, so that all

methods are probed in the same way. The compressed representations are mapped to a hidden size of 768 by a single linear projection, and are then fed as a prefix to a randomly initialized 8-layer Transformer decoder with 158M parameters. The decoder is trained to reconstruct the context with the next-token prediction loss on the context tokens. We use contexts of 512 tokens from the training splits of the six in-domain datasets (Appendix A.2), train the decoder for 10k steps, and evaluate it on the validation splits of the same datasets. In addition to the reconstruction NLL, we report the entropy ratio

$$H_r = 1 - \frac{\text{NLL}}{\log |\mathcal{V}|},$$

where $\log |\mathcal{V}|$ is the cross-entropy of a uniform prediction over the vocabulary $\mathcal{V}$ of the Llama-3 tokenizer ($|\mathcal{V}| = 128{,}256$). In other words, H_r measures how much the NLL is reduced relative to this uniform baseline, so a higher H_r indicates that more information about the context can be recovered from the compressed representations.

A.2 Datasets

Table 7 shows the statistics of the datasets we use in the experiments.

Table 7: Statistics of the training and evaluation datasets used, including in-domain and out-of-domain datasets. #Train represents the number of training samples and #Validation represents the number of validation samples.

Dataset	Average Question Length	Average Context Length	#Train	#Validation
SQuAD	11	137	86,588	10,507
NewsQA	8	599	74,160	4,212
TriviaQA	16	784	61,688	7,785
SearchQA	17	749	117,384	16,980
HotpotQA	22	232	72,928	5,904
Natural Questions	9	153	104,071	12,836
BioASQ	11	248	–	1,518
DROP	11	243	–	1,501
DuoRC	9	681	–	1,503
RACE	12	349	–	1,502
RelationExtraction	9	30	–	1,500
TextbookQA	11	657	–	1,508

A.3 Model Inference Implementation

We carefully handle the positions of the padding tokens, which we find brings improvements to all methods. We keep the no-padding setup during NTP to make full use of the GPUs. In SFT, as we stick to the question-unaware setup, we have to encode the contexts first and feed the context compression tokens along with the question tokens to the decoder. For efficient batch compression, we first apply left-padding to the context tokens for the compressors. We then concatenate the compression tokens with padded question and answer tokens, and perform an operation to move all the padding tokens to the leftmost side according to the attention mask. We also create the shifted new attention mask and labels accordingly. This ensures that the generation starts from a non-padding token and there is no position gap between the compression tokens and the questions.

A.4 Evaluation on More Tasks

We further evaluate ComprExIT on summarization, fact-checking, and in-context learning. For summarization, we fine-tune all methods on XSum [Narayan et al., 2018] with Llama-3.2-1B and report the results in Table 8. ComprExIT achieves the best ROUGE-1/ROUGE-L scores, 0.333/0.262, outperforming both ICAE and the uncompressed prompt-tuning baseline. This suggests that ComprExIT preserves not only answer-relevant facts but also the broader discourse information needed for abstractive generation. For fact-checking and in-context learning, we directly evaluate the NTP-trained Llama-3.2-3B models on FEVER [Thorne et al., 2018] and SST-2 [Socher et al., 2013], respectively, as shown in Table 9. On FEVER, ComprExIT achieves the best score, 64.39, surpassing the prompt-tuning baseline by 3.91% and ICAE by 8.69%. On SST-2 ICL, ComprExIT obtains 69.17,

Table 8: ROUGE-1 and ROUGE-L scores on the summarization (XSum) dataset.

Model	ROUGE-1	ROUGE-L
ICAE	0.324	0.257
Prompt Tuning	0.320	0.253
ComprExIT	**0.333**	**0.262**

Table 9: F1 scores on fact checking (FEVER) and in-context learning (SST-2) datasets.

Model	FEVER	SST-2 ICL
ICAE	59.24	66.01
Prompt Tuning	61.97	**72.34**
ComprExIT	**64.39**	69.17

Table 10: Experimental results on six **out-of-domain** MRQA benchmark datasets. Out-of-domain: these datasets are not included in the training data. Best among compression methods is **bold**. Our method is colored in . The prompt-tuned uncompressed baseline is colored in . Note that ICAE and 500x have a smaller compression ratio on samples with contexts shorter than 512 tokens because they are trained with a fixed set of 128 compression tokens. In particular, on the RelationExtraction dataset, whose average context length is only 30, they use more compression tokens than context tokens.

Methods	RelationExtraction		BioASQ		TextbookQA		DuoRC		DROP		RACE		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Llama-3.2-1B-Base														
Prompt tuning [w/ context]	69.91	81.15	65.96	78.80	52.10	59.61	38.51	47.40	32.54	43.14	33.68	46.30	48.78	59.40
Zero-shot [w/ context]	27.48	47.87	6.98	28.84	18.16	29.97	17.46	30.54	18.43	32.30	10.53	22.71	16.51	32.04
Zero-shot [w/o context]	5.09	10.33	14.76	20.56	11.11	16.03	0.60	2.55	14.84	19.73	0.45	2.88	7.81	12.01
ICAE [Ge et al., 2024]	**62.28**	**75.97**	42.62	54.09	28.21	34.60	12.59	18.80	24.15	33.23	9.50	16.95	29.89	38.94
500x [Li et al., 2025c]	6.31	16.30	22.54	35.52	16.77	24.40	1.07	3.91	8.18	18.32	1.93	5.70	9.47	17.36
Beacon [Zhang et al., 2025]	33.18	48.82	32.25	47.84	5.92	10.34	2.40	4.90	21.82	29.24	15.88	25.35	18.57	27.75
ComprExIT	59.46	74.96	**48.40**	**63.74**	**41.65**	**50.73**	**28.78**	**38.77**	**29.01**	**39.86**	**23.00**	**37.95**	**38.38**	**51.00**
Llama-3.2-3B-Base														
Prompt tuning [w/ context]	77.37	86.70	67.62	81.50	60.48	68.75	45.50	55.63	48.84	57.18	43.62	56.35	57.24	67.68
Zero-shot [w/ context]	54.24	68.46	43.22	64.31	36.46	46.66	28.31	42.36	23.69	37.45	22.70	38.16	34.77	49.57
Zero-shot [w/o context]	14.28	21.03	20.94	25.76	23.29	28.89	2.07	4.44	16.17	21.26	1.78	5.65	13.09	17.84
ICAE [Ge et al., 2024]	**70.66**	**82.10**	55.85	67.21	42.25	50.70	23.65	32.81	35.33	44.89	21.51	33.85	41.54	51.93
500x [Li et al., 2025c]	65.13	79.67	47.67	60.82	29.41	38.10	22.25	30.91	29.74	39.12	19.14	30.58	35.56	46.53
Beacon [Zhang et al., 2025]	58.92	71.62	54.52	66.86	10.98	15.68	4.73	7.95	39.39	50.55	30.71	43.95	33.21	42.77
ComprExIT	65.31	77.25	**59.64**	**73.87**	**56.09**	**64.50**	**34.31**	**44.98**	**43.58**	**53.16**	**31.60**	**47.69**	**48.42**	**60.24**

improving over ICAE by 4.79% and remaining competitive with the prompt-tuning baseline (72.34). Overall, these results show that ComprExIT generalizes consistently well beyond QA.

A.5 Out-of-Domain Results

Table 10 reports the full results on the six out-of-domain MRQA benchmarks. Overall, ComprExIT remains the strongest compression method across both Llama-3.2-1B-Base and Llama-3.2-3B-Base, showing that its learned compressed representations transfer well beyond the training domains. We note that ICAE is competitive on RelationExtraction, whose contexts are very short on average. Because prior methods use a fixed budget of 128 compression tokens, this dataset effectively gives them a much smaller compression ratio, which partially explains the smaller gap on that benchmark.

A.6 Sensitivity to OT Hyperparameters

We study how sensitive ComprExIT is to three hyperparameters of the allocation module: the number of Sinkhorn iterations, the segment length T on which Sinkhorn is run, and the projection dimension of the utility space. For each setting, we train Llama-3.2-1B with the full NTP and SFT procedure under the same setup as the main experiment. As shown in Table 11, performance saturates at around 30 iterations, our default setting, and more iterations bring only a negligible gain (66.58 at both 50 and 100 iterations). Even with 5 iterations, ComprExIT still reaches 61.32 and outperforms the strongest compression baseline in Table 1 (56.16), since a few iterations are already enough to produce a rough allocation plan. For the segment length, a small segment ($T = 64$) degrades performance because it limits long-range allocation from token anchors to compression

Table 11: Sensitivity of ComprExIT to OT hyperparameters on Llama-3.2-1B (Avg. F1 on the six in-domain datasets). Our default settings are underlined.

Sinkhorn iter.	5	10	<u>30</u>	50	100
Avg. F1	61.32	65.49	66.55	66.58	66.58
Segment T	64	<u>128</u>	512		
Avg. F1	64.07	66.55	66.00		
Proj. dim.	64	<u>256</u>	512	1024	
Avg. F1	65.64	66.55	65.89	65.62	

slots, while a larger segment ($T = 512$) brings no further gain, which supports our choice of $T = 128$ in Section 5. The projection dimension has little effect, and all settings stay within one F1 point of each other. Overall, these results show that our default settings lie in a stable range, where moderate changes of the hyperparameters have little effect on performance. Note that we do not tune the number of compression slots separately, since it is determined by the compression ratio, which we study in Table 4.

A.7 Memory Cost under Long Contexts

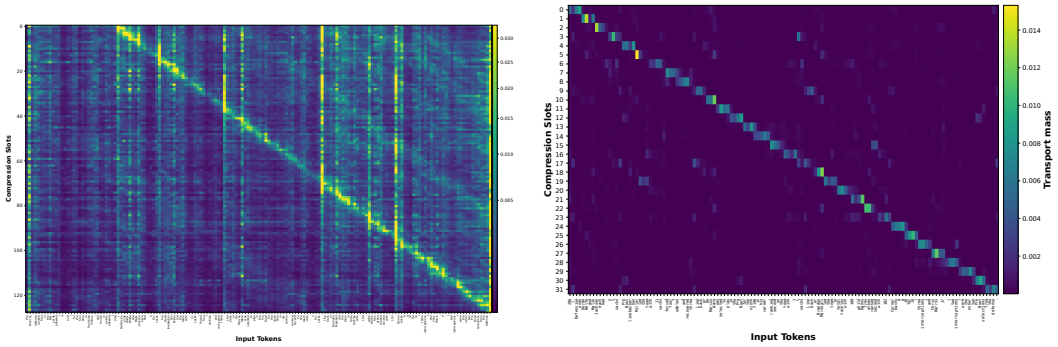
To study the memory cost of ComprExIT at long contexts, we extend the latency analysis (Appendix A.1.2) to context lengths of 16k and 32k. We use the models trained with Llama-3.2-3B in the main experiment, and benchmark each method in BF16 precision with FlashAttention-2, a batch size of 1, and a generation length of 4096 tokens on a single NVIDIA A100 80GB PCIe GPU. We report the end-to-end peak GPU memory together with the compression time. We do not include 128k contexts, since they do not fit into the memory of a single 80GB GPU. For windowed compression, we split the context into windows of 2048 tokens and compress them sequentially, so that only the hidden states of the current window need to be kept in memory. Note that these windows are different from the segments of length T on which Sinkhorn is run.

As shown in Table 12, ComprExIT has a higher peak memory than ICAE at both lengths, because it needs to keep the hidden states of multiple layers for the layer-wise aggregation. At the same time, its compression stage remains about $2\times$ faster than that of ICAE. Windowed compression effectively reduces this memory overhead: it lowers the peak memory of ComprExIT from 19.33 GB to 11.76 GB at 16k and from 32.57 GB to 16.83 GB at 32k, both below that of ICAE, while keeping the same compression time. Since we only measure the efficiency of this windowed variant, we leave training ComprExIT with windowed compression to future work.

Table 12: Peak GPU memory (Mem., GB) and compression time (Time, s) at long contexts. The windowed variant is only used to measure efficiency and is not trained.

Method	16k		32k	
	Mem.	Time	Mem.	Time
Full Context	8.91	–	11.79	–
ICAE	15.47	1.12	23.44	2.75
ComprExIT	19.33	**0.52**	32.57	**1.36**
ComprExIT (windowed)	**11.76**	**0.52**	**16.83**	**1.36**

A.8 The Allocation Matrix



(a) ICAE (b) ComprExIT

Figure 7: Last-layer attention heatmap of gist tokens produced by ICAE (left) and width-wise information transmission plan in ComprExIT (right).

Figure 7b visualizes the learned width-wise transmission plan in ComprExIT. Compared with existing methods (e.g., ICAE in Figure 7a), ComprExIT exhibits a more structured allocation pattern: each compression slot is softly anchored to a contiguous local region of input tokens, while still retaining non-negligible connections to distant tokens. This enables ComprExIT to preserve the semantic order of the context without sacrificing the ability to capture long-range dependencies. In contrast, although ICAE tries to maintain an order bias (there is a diagonal pattern of high attention values in Figure 7a), its compression-token attention patterns are weakly localized. Moreover, it can be observed that some important tokens (e.g., entity "Marcus", number "4.") are ignored collectively by

most of the compression tokens, but some entities (e.g., Broncos) are overly focused. We attribute these observations to the lack of coordinated allocation.

A.9 Distribution Mismatch Induced by Layerwise Dilution

In this subsection, we formalize the potential mismatch introduced by layerwise dilution. The key idea is that if the compression-token representations drift slightly at each layer, then these small discrepancies can accumulate with depth and enlarge the mismatch between the final compressor output and the representation space expected by the decoder.

Let $p(Z^{(\ell)})$ denote the empirical distribution of compression-token representations $\{\boldsymbol{z_k^{(\ell)}}\}_{k=1}^{K}$ at layer ℓ, and let p_{dec} denote the representation distribution expected by the decoder. We measure this mismatch using the Wasserstein distance $\mathcal{W}$.

Proposition A.1 (Accumulation of layerwise drift)**.** *Assume $p(Z^{(0)}), \ldots, p(Z^{(L)})$ and p_{dec} all have finite first moments. Define the final compressor–decoder mismatch as*

$$\mathcal{M} := \mathcal{W}\Big(p(Z^{(L)}),\, p_{dec}\Big)\,. \tag{11}$$

Then

$$\mathcal{M} \le \sum_{\ell=0}^{L-1} \mathcal{W}\Big(p(Z^{(\ell+1)}),\, p(Z^{(\ell)})\Big) + \mathcal{W}\Big(p(Z^{(0)}),\, p_{dec}\Big)\,. \tag{12}$$

Proof. By the triangle inequality for the Wasserstein distance [Clement and Desch, 2008],

$$\mathcal{W}\Big(p(Z^{(L)}),\, p_{\text{dec}}\Big) \le \mathcal{W}\Big(p(Z^{(L)}),\, p(Z^{(L-1)})\Big) + \mathcal{W}\Big(p(Z^{(L-1)}),\, p_{\text{dec}}\Big)\,. \tag{13}$$

Applying the same inequality recursively to the second term yields

$$\mathcal{W}\Big(p(Z^{(L-1)}),\, p_{\text{dec}}\Big) \le \mathcal{W}\Big(p(Z^{(L-1)}),\, p(Z^{(L-2)})\Big) + \mathcal{W}\Big(p(Z^{(L-2)}),\, p_{\text{dec}}\Big)\,, \tag{14}$$

and continuing this expansion down to layer 0 gives

$$\mathcal{M} \le \sum_{\ell=0}^{L-1} \mathcal{W}\Big(p(Z^{(\ell+1)}),\, p(Z^{(\ell)})\Big) + \mathcal{W}\Big(p(Z^{(0)}),\, p_{\text{dec}}\Big)\,. \tag{15}$$

This proves the claim. □

The proposition shows that the final mismatch is upper-bounded by two terms: the initial mismatch at the compressor input, and the cumulative layerwise drift accrued across the encoder. Therefore, even if each layer introduces only a modest shift, the total drift can still grow with depth, increase $\mathcal{M}$, and make the compression objective harder to optimize. Together with the extensive experiments in Sections 3 and Appendix A, this provides both theoretical and empirical support for the role of information dilution.

A.10 Training

A.10.1 Optimization Behavior

We plot the NTP training curves in Figure 8. ComprExIT starts from the lowest loss and converges faster to the best plateau, indicating a smaller compressor-decoder mismatch and easier optimization. This supports our core claim that layer aggregation and coordinated allocation produce compressed states that preserve useful information while remaining decoder-friendly.

A.10.2 Training Details

Tables 13 to 15 present the hyperparameters we use when training ComprExIT and the baseline methods.

Table 13: NTP Training configuration for ComprExIT.

Item	ComprExIT
Run & setup	
Dataset	SlimPajama-6B
Number of Tokens	1 Billion
Device	4× NVIDIA-A100
Precision	Bfloat16
Sequence & compression	
Context length	512
Generation length	128
Compression ratio	4
Optimal-Transport (OT)	
OT window size	128
OT iterations	30
OT projection dimension	256
Layerwise gate hidden size	256
Optimization	
Learning rate	1×10^{-4}
Warmup ratio	0.05
Max grad norm	20.0
Batch size (per device)	16
Gradient accumulation steps	32
Actual Batch Size	2048
Epochs	1

Table 14: NTP Training configuration for ICAE and 500x.

Item	ICAE	500x
Run & setup		
Dataset	SlimPajama-6B	SlimPajama-6B
Number of Tokens	1 Billion	1 Billion
Device	4× NVIDIA-A100	4× NVIDIA-A100
Precision	Bfloat16	Bfloat16
Sequence & compression		
Context length	512	512
Generation length	128	128
Compression ratio	4	4
Memory		
Number of memory tokens	128	128
Optimization		
Learning rate	3×10^{-5}	3×10^{-5}
Warmup ratio	0.05	0.05
Max grad norm	20.0	20.0
Batch size (per device)	16	16
Gradient accumulation steps	16	32
Actual Batch Size	2048	2048
Epochs	1	1
LoRA (compressor)		
Enabled	True	True
Rank r	128	128
Alpha α	32	32
Dropout	0.05	0.05
Bias	none	none
Task type	CAUSAL_LM	CAUSAL_LM

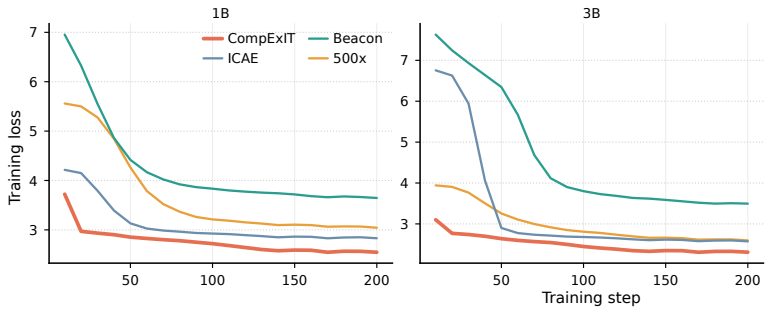


Figure 8: The training curves of baseline methods and ComprExIT under the next-token prediction task.

Table 15: NTP Training configuration for Activation Beacon.

Item	Activation Beacon
Run & setup	
Dataset	SlimPajama-6B
Number of Tokens	1 Billion
Device	4× NVIDIA-A100
Precision	Bfloat16
Sequence	
Context length	512
Generation length	128
Activation Beacon configuration	
Beacon enabled	True
Beacon window / stride	64 / 64
Beacon ratio	4
Beacon attention	full-coverage
Attend previous beacons	True
Trainable beacon params	q, k, v
Beacon position	interleave
Grouping by stride	strict
Optimization	
Learning rate	1×10^{-4}
Max grad norm	20.0
Batch size (per device)	16
Gradient accumulation steps	32
Actual Batch Size	2048
Epochs	1

A.11 Broader Impact

This work studies learned context compression for long-context language models. Its primary potential benefit is improved efficiency: effective compression can reduce memory usage, latency, and computational cost while preserving salient information from the original context. In turn, this may make document-grounded models more accessible to researchers and practitioners with limited hardware resources, and may reduce the energy required to deploy such systems at scale. More efficient long-context modeling may also broaden the practical use of language models in settings where rapid processing of long documents is important, such as literature review, writing assistance, and educational tools. Gist tokens can also function as encrypted memory or a communication medium for LLM agents while remaining unreadable to humans.

At the same time, context compression introduces important risks. Any compression method may discard rare but critical details, qualifiers, or provenance signals, which can lead to overconfident but incomplete answers. Such failures may be especially concerning in high-stakes applications such as medicine, law, science, or public policy, where omitted evidence can materially affect the

correct conclusion. In addition, compressed-context models can still inherit social biases, factual errors, and privacy risks from their pretrained backbone and training data, and efficient processing of long documents could be misused for large-scale analysis of sensitive text. We therefore recommend task-specific evaluation, preserving access to the original context whenever possible, and avoiding deployment as a substitute for expert judgment in safety-critical settings.

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: The abstract and introduction clearly state the paper's main claims: the two structural bottlenecks, the proposed explicit-transmission framework, and the empirical gains. These claims are supported by the method and experiments in Section 5 and Section 6.1.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: The paper explicitly discusses limitations in Section 7: we do not test much larger backbones, we evaluate task performance only up to 8k-token contexts and measure efficiency alone at 16k and 32k, ComprExIT has a higher peak memory than ICAE unless the context is compressed window by window, and our experiments focus on text-only models.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: The appendix Section A.9 includes a formal theoretical result (the proposition on accumulation of layerwise drift), states its assumption explicitly (finite first moments for the relevant distributions), and provides a complete proof based on the triangle inequality for Wasserstein distance.

Guidelines:

- The answer [N/A] 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 detailed description of our method in Section 5. We provide details of our experiment setups in Section 6.1 and include the specifications to reproduce our training process in Section A.10.2.

Guidelines:

- The answer [N/A] 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: [Yes]

   Justification: The evaluation datasets and base models are public, and the paper provides detailed experimental instructions. Our code is publicly available at `https://github.com/Jiangnan0522/ComprExIT`, with instructions to reproduce the main experimental results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6.1 specifies the base models, training/evaluation protocol, datasets, metrics, context length, and compression ratio. The appendix Section A.10.2 provides dataset statistics, implementation details, and per-method hyperparameter tables for training. Together, these details are sufficient to understand and interpret the reported results.

   Guidelines:

   - The answer [N/A] 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: The reported QA metrics are aggregate EM/F1 scores computed over evaluation splits containing many validation examples, and the latency analysis in the appendix reports the mean over 200 timed iterations. However, the main experimental results are still presented without run-to-run error bars, confidence intervals, or formal significance tests, so we answer this item conservatively.

Guidelines:

- The answer [N/A] 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 provide the key compute details needed to reproduce the reported experiments in the paper: training is described with device information (e.g., 4×NVIDIA A100), precision and optimization settings in Sections A.10.2 and 6.1. The latency study specifies the hardware, memory, workload configuration, warm-up runs, and timed iterations (NVIDIA L40S 48GB GPU; 50 warm-up and 200 timed iterations) in Section A.1.2. Together, these disclosures provide sufficient information about the compute workers and execution setup for reproduction.

   Guidelines:

   - The answer [N/A] 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: The work complies with the NeurIPS Code of Ethics, with particular attention to data integrity, transparency, and reproducibility.

   Guidelines:

   - The answer [N/A] 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: Section A.11 discusses both positive impacts, such as lower memory, latency, and energy costs, and negative impacts and risks, such as omission of critical details, bias, privacy, and misuse.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The submission does not release a high-risk generative model or scraped dataset, so controlled-release safeguards are not a central issue for this work.

    Guidelines:

    - The answer [N/A] 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: All assets used in this paper, including code, datasets, and pre-trained models, are appropriately attributed to their original creators or owners.

Guidelines:

- The answer [N/A] 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: [Yes]

Justification: We release our code and trained checkpoints at `https://github.com/Jiangnan0522/ComprExIT` under the Apache 2.0 license. The README documents the setup, training, evaluation, and reproduction of the baselines.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work does not involve crowdsourcing or research with human participants.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: The work does not involve crowdsourcing or research with human participants, so IRB-style review is not applicable.

Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

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

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.