

Cognitive Chunking for Soft Prompts: Accelerating Compressor Learning via Block-wise Causal Masking

Guojie Liu^{1*}, Yiqi Wang^{1*}, Yanfeng Yang¹, Wenqi Fan²,
Songlei Jian¹, Jianfeng Zhang¹, Jie Yu^{1†},

¹College of Computer Science and Technology,
National University of Defense Technology, China

²The Hong Kong Polytechnic University
yj@nudt.edu.cn

Abstract

Providing extensive context via prompting is vital for leveraging the capabilities of Large Language Models (LLMs). However, lengthy contexts significantly increase inference latency. To mitigate this issue, context compression—particularly soft prompt compression—has emerged as a widely studied solution, which converts long contexts into shorter memory embeddings via a trained compressor. Existing methods typically compress the entire context indiscriminately into a set of memory tokens, requiring the compressor to capture global dependencies and necessitating extensive pre-training data to learn effective patterns. Inspired by the chunking mechanism in human working memory and our empirical observation of spatial specialization within memory embeddings, we propose Parallelized Iterative Compression (PIC). By simply modifying the Transformer’s attention mask, PIC explicitly restricts the receptive field of memory tokens to sequential local chunks, thereby significantly reducing the difficulty of compressor training. Experiments across multiple downstream tasks demonstrate that PIC outperforms competitive baselines while dramatically expediting the training process. Finally, we provide an investigation into the emergence of spatial specialization, revealing how tuning strategies and explicit structural priors shape the latent structure of the memory embeddings.

1 Introduction

To enhance the generation quality of Large Language Models (LLMs), incorporating extensive contextual information has become a common practice in paradigms like Retrieval-Augmented Generation (RAG) (Gao et al., 2023) and In-Context Learning (ICL) (Li et al., 2024). However, extending context length incurs significant computational



Figure 1: An illustration of unconstrained global attention in a typical soft prompt compression method: both memory tokens need to attend to all the input words simultaneously.

overhead and inference latency, posing critical challenges for real-time applications.

Context compression mitigates this issue by reducing the effective sequence length while preserving core information. Existing compression approaches can be broadly categorized into two types: hard prompt compression and soft prompt compression (Li et al., 2025). Hard prompt compression relies on heuristic rules to prune redundant tokens from the original context or employs another LLM to produce a discrete, natural language summary, fundamentally retaining a symbolic representation (Li et al., 2023; Jiang et al., 2023; Chuang et al., 2024; Pan et al., 2024; Jiang et al., 2024). In contrast, soft prompt compression learns a compact, continuous representation by distilling the context information into a small set of dense embedding vectors (Chevalier et al., 2023; Ge et al., 2024; Rau et al., 2025; Cheng et al., 2024; Louis et al., 2025; Dai et al., 2025), which can be viewed as a novel modality offering the potential for higher compression rates (Kuratov et al., 2025). Therefore, this work focuses on soft prompt compression. Typically, a compressor maps the long input into short memory embeddings, which are then passed to a downstream decoder to guide generation.

However, most existing methods rely on unconstrained global attention to map the entire context into fixed memory slots, as illustrated in Figure 1. Lacking a structural inductive bias to establish explicit correspondences between memory tokens and local subsequences, the compressor is

*These authors contributed equally to this work.

†Corresponding author

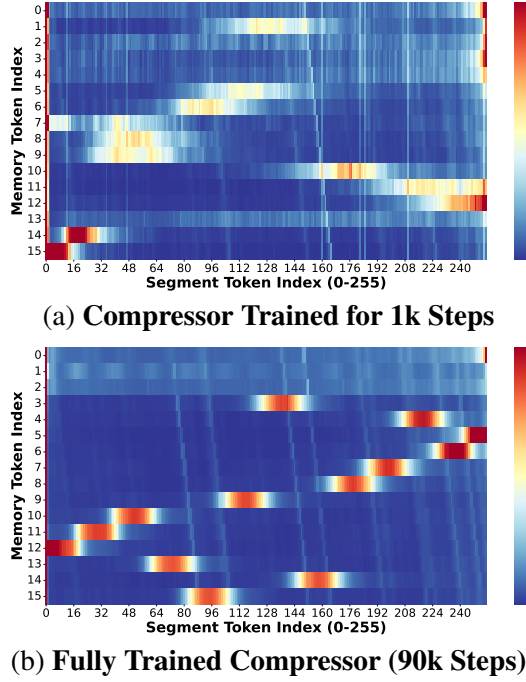


Figure 2: Attention weight heatmap between memory tokens and original context tokens in PCC. Red indicates higher attention weight, while blue represents lower attention weight. As training progresses, the highly diffuse attention distribution gradually evolves into a definitive block-wise mapping, demonstrating the emergence of spatial specialization.

forced to attend to the entire lengthy context (Yang et al., 2018a). This unconstrained mechanism requires the model to implicitly learn complex global routing patterns, rendering the training process highly optimization-heavy, data-intensive, and time-consuming (Goyal and Bengio, 2022).

To demystify how a typical global compressor (Dai et al., 2025) allocates its limited memory tokens, we analyze both the cross-attention weights and the embedding cosine similarities between memory tokens and original context tokens. As shown in Figure 2a, early in training (1k steps), the attention distribution is diffuse and unstructured. However, upon full convergence (90k steps), a definitive block-wise pattern emerges (Figure 2b): each memory token strongly aligns with a specific, contiguous segment of the context. (Embedding similarities exhibit an identical block-wise structure; see Appendix Figure 12).

This observed spatial specialization bears a resemblance to the cognitive principle of chunking in human working memory (Baddeley, 1992; Barak and Tsodyks, 2014). It is evident from cognitive studies that working memory, constrained by a limited capacity (Cowan, 2010), compensates by orga-

nizing information into coherent, higher-order units or “chunks”—such as grouping digits in a phone number or phrases in a sentence—thereby facilitating efficient storage and recall (Zhong et al., 2024). Whereas human comprehension naturally unfolds sequentially by interpreting each new chunk in light of previously integrated history, existing compressors are forced to discover this partitioning implicitly from scratch. There are no explicit mechanisms to enforce a sequential correspondence between memory tokens and text blocks.

Motivated by this gap, we explicitly impose a sequential, block-wise alignment, mandating each memory token to attend exclusively to a designated ordered text segment. This structural prior fundamentally simplifies the learning objective from complex global dependency modeling to localized information extraction. A straightforward approach to achieve such strict alignment is through iterative compression (i.e., processing the text chunk by chunk); however, this inevitably entails prohibitive inference latency. To enforce this constraint without sacrificing the inference speed of global attention, we propose Parallelized Iterative Compression (PIC). By leveraging a novel block-wise causal mask, PIC simulates incremental, chunk-by-chunk processing in a single parallelized forward pass, rigidly preserving sequential dependencies while bypassing the severe latency bottleneck of traditional iterative methods.

Extensive experiments on representative downstream tasks (RAG-based QA and ICL) demonstrate that PIC outperforms competitive baselines. Crucially, a data-efficiency analysis confirms that our structural prior substantially reduces learning difficulty, enabling the compressor to surpass baseline peak performance with 67% less training time under a $4\times$ compression setting. Finally, we provide an in-depth investigation into the learned memory embeddings, and reveal that the emergence of spatial specialization is also closely related to the tuning strategies. The key contributions of this paper are summarized as follows:

- By investigating the relation between memory embeddings and input context, we identify a distinct spatial-specialization phenomenon, observing that unconstrained memory representations naturally evolve to compress specific, localized blocks of the original text.
- Inspired by cognitive chunking in human working memory, we introduce a structural inductive

bias to accelerate compressor training. This approach simplifies global modeling into localized extraction and is simply implemented via a block-wise causal mask.

- Extensive experiments demonstrate that our method achieves strong performance improvements on RAG and ICL downstream tasks. Additionally, we conduct empirical analysis to unveil the underlying mechanisms of spatial specialization. Source code is released at https://github.com/cinuo/cognitive_chunking_for_soft_prompts.

2 Related Work

Existing context compression approaches fall into two broad categories: hard prompt compression and soft prompt compression (Li et al., 2025). Hard prompt compression, such as LLMingua-2 (Pan et al., 2024), produces compressed outputs in human-readable natural language, by formulating the task as token-level classification. Our work focuses on soft prompt compression, which condenses text into continuous memory representations. Early methods like AutoCompressor (Chevalier et al., 2023) introduced iterative summary generation, while PPC (Mu et al., 2023) demonstrated that simple attention masks could guide models to compress prompts into “gist tokens”. Some follow-up works, such as xRAG (Cheng et al., 2024), employ knowledge distillation as their primary training strategy, focusing on training only the modality bridge. Other frameworks, including ICAE (Ge et al., 2024), present a general compressor-decoder architecture pre-trained via reconstruction and prediction tasks. COCOM (Rau et al., 2025) further emphasizes that the downstream decoder requires additional fine-tuning to effectively utilize compressed memory. Subsequent analyses, such as PISCO (Louis et al., 2025), examine embedding behaviors in such settings. Building on these, PCC (Dai et al., 2025) explores design questions like scalability across compression ratios.

3 Method

Standard approaches compress long contexts globally, imposing a heavy burden on models to implicitly learn complex attention patterns. Motivated by our observation of spatial specialization and the sequential nature of human reading, we introduce a structured block-wise paradigm. This paradigm explicitly constrains each memory slot

to compress only a designated sequential segment, thereby reducing optimization difficulty. However, a naive iterative implementation of this strategy incurs unacceptable latency. To break this efficiency bottleneck, we propose Parallelized Iterative Compression (PIC). By leveraging a novel block-wise causal mask, PIC simulates iterative processing fully in parallel. This design strictly preserves the sequential inductive bias while maintaining the high inference speed of direct global compression.

3.1 Problem Definition

Soft prompt compression aims to distill a context $\mathbf{X} = (x_1, \dots, x_L)$ into a compact memory representation $\tilde{\mathbf{H}} = (h_1, \dots, h_N)$, where $N \ll L$, x_i is a discrete text token, and h_j is a continuous memory embedding. These embeddings $\tilde{\mathbf{H}}$ then replace $\mathbf{X}$ to condition the downstream LLM generation: $f_{\text{LLM}}(\tilde{\mathbf{H}}, \text{prompt}) \rightarrow \mathbf{Y}_{\text{output}}$.

As illustrated in Figure 3a, existing methods typically formulate compression as a global mapping problem (Dai et al., 2025). By appending N learnable memory tokens $\tilde{\mathbf{M}} = (m_1, \dots, m_N)$ as placeholders, the compressor processes the entire context $\mathbf{X}$ simultaneously. Thus, each memory embedding $h_t = \text{Compressor}(\mathbf{X}, m_{<t})$ depends globally on the full input $\mathbf{X}$ and preceding memory tokens. However, this unconstrained approach forces the compressor to learn complex, dense attention patterns over long sequences, which is optimization-heavy and sub-optimal.

3.2 Iterative Compression

To mitigate the difficulty of learning global dependencies, we draw inspiration from human cognitive chunking and introduce an iterative compression strategy (Figure 3b). We partition the input $\mathbf{X}$ into N equal-length, non-overlapping chunks $\mathbf{X} = (c_1, \dots, c_N)$. Compression proceeds sequentially: the t -th memory token h_t compresses only its designated chunk c_t while conditioned on the historical memory $h_{<t}$, formulated as $h_t = \text{Compressor}(c_t, h_{<t})$. This introduces a strong inductive bias, enforcing a one-to-one mapping between memory tokens and text chunks. Thus, the generation of h_t is local to the input text but global to the compressed history.

Although this sequential decomposition simplifies optimization by replacing global dependency modeling with localized extraction tasks, its strict serial dependency inherently prevents parallel computation. This results in inference latency scaling

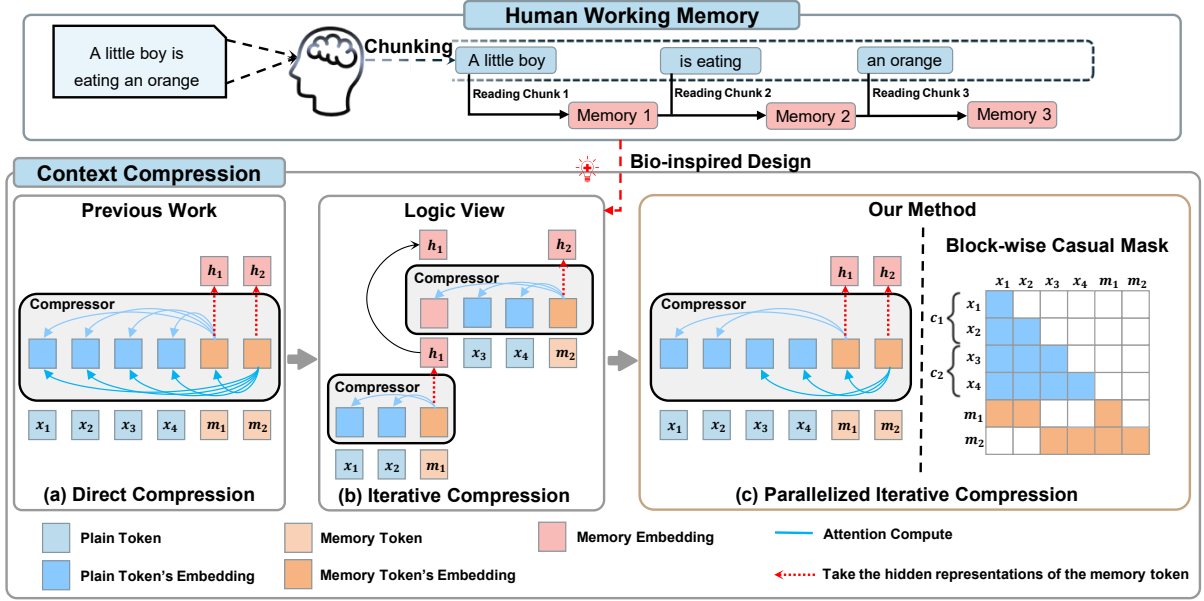


Figure 3: Comparison of different compression paradigms. (a) Direct Compression: Each memory token attends globally to the entire input context without structural constraints. (b) Iterative Compression: Inspired by the chunking mechanism in human working memory, we split the input context into chunks, and memory tokens are generated sequentially, with each token attending only to its specific chunk and previous memory. (c) PIC: Our proposed method allows all memory tokens to be processed in parallel within a single sequence, using a block-wise causal mask to enforce the same local dependency constraints as the iterative approach.

linearly with N , creating a severe bottleneck.

3.3 Parallelized Iterative Compression

To resolve the latency bottleneck while preserving the block-wise inductive bias, we propose PIC. As illustrated in Figure 3c, PIC processes all input tokens and memory placeholders simultaneously in a single forward pass, employing a block-wise causal attention mask to strictly simulate the receptive field.

3.3.1 Sequence Construction

Unlike step-by-step sequential execution, PIC concatenates all chunks and memory placeholders to enable parallelization. Assuming $\mathbf{X}$ is uniformly partitioned into N chunks ($c_1, \dots, c_N$) and $\widetilde{\mathbf{M}} = (m_1, \dots, m_N)$ denotes the learnable memory tokens, the unified input sequence $\mathbf{Z}$ is simply constructed as $\mathbf{Z} = [\mathbf{X}, \widetilde{\mathbf{M}}]$.

3.3.2 Block-wise Causal Attention Mask

Our core innovation is the block-wise causal mask $\mathbf{M} \in \mathbb{R}^{|\mathbf{Z}| \times |\mathbf{Z}|}$, ensuring each memory token m_t selectively attends to its assigned chunk c_t and the memory history $m_{<t}$. For query z_i and key z_j in $\mathbf{Z}$, we define a visibility function $\text{Visible}(i, j) \in \{0, 1\}$ based on three rules:

1. Intra-Context: If $z_i, z_j \in \mathbf{X}$, $\text{Visible}(i, j) = 1$ when $i \geq j$.
2. Intra-Memory: If $z_i, z_j \in \widetilde{\mathbf{M}}$, $\text{Visible}(i, j) = 1$ when $i \geq j$.
3. Memory-to-Chunk: If query z_i is a memory token m_t and key z_j is a context token in chunk c_k , $\text{Visible}(i, j) = 1$ when $t = k$.

The mask entry is then naturally derived:

$$M_{i,j} = \begin{cases} 0 & \text{if } \text{Visible}(i, j) = 1 \\ -\infty & \text{otherwise} \end{cases} \quad (1)$$

By cutting off direct access to prior chunks $c_{<t}$, this mask forces m_t to extract local information from c_t and integrate global coherence solely via $m_{<t}$. Thus, PIC closely approximates iterative compression without serial latency.

3.4 Pre-training and Fine-tuning

We align our training pipeline with the PCC framework (Dai et al., 2025), utilizing identical pre-training tasks and fine-tuning objectives.

3.4.1 Pre-training Objectives

Text Reconstruction (TR). The model must autoregressively reconstruct the original context $\mathbf{X}$ con-

Table 1: Basic Statistics of Data Samples for Datasets

Stage	Source	Train	Test
Pre-training	FineWeb	11,520,000	576
Fine-tuning(RAG)	SQuAD	86,821	5,928
Fine-tuning(ICL)	GSM8K	6,725	748

ditioned on the compressed memory $\tilde{\mathbf{H}}$. Appending a special token $\langle \text{AE} \rangle$ to $\tilde{\mathbf{H}}$ indicates this reconstructive task. The objective is formalized as:

$$\mathcal{L}_{\text{TR}} = -\frac{1}{L} \sum_{i=1}^L \log f_{\text{LLM}}(x_i | \tilde{\mathbf{H}}, \langle \text{AE} \rangle, x_{<i}) \quad (2)$$

where $x_{<i}$ denotes the prefix $(x_1, \dots, x_{i-1})$.

Text Completion (TC). This objective encourages capturing semantic information essential for future generation. We partition the text into a prefix and a continuation, compressing the memory $\tilde{\mathbf{H}}$ from the prefix. The model then predicts the continuation starting from index $k + 1$:

$$\mathcal{L}_{\text{TC}} = -\frac{1}{L-k} \sum_{i=k+1}^L \log f_{\text{LLM}}(x_i | \tilde{\mathbf{H}}, x_k, \dots, x_{i-1}) \quad (3)$$

The total pre-training loss is their weighted sum: $\mathcal{L} = \lambda \mathcal{L}_{\text{TC}} + (1 - \lambda) \mathcal{L}_{\text{TR}}$, with $\lambda = 0.5$.

3.4.2 Fine-tuning

For downstream adaptation, we domain-specifically fine-tune the compressor on a single representative dataset per task category, following PCC’s identical settings.

4 Experiments

4.1 Experimental Details

4.1.1 Datasets

Table 1 provides a summary of the datasets utilized across different training stages. In the pre-training stage, we leveraged approximately 3 billion tokens from the FineWeb (Penedo et al., 2024) dataset. For the fine-tuning stage, two datasets were incorporated: SQuAD (Rajpurkar et al., 2018), comprising 86,821 training samples and 5,928 test samples, and GSM8K (Cobbe et al., 2021), containing 6,725 training samples and 748 test samples. Specifically, we employed the training set of SQuAD as the domain-specific dataset for the RAG-based QA task, and GSM8K for the ICL task.

4.1.2 Implementation Details and Baselines

Our experiments primarily focus on efficient context compression using accessible model scales. During the pre-training phase, we explored architectures from two different model families as compressors. Primarily, we engaged Qwen2.5-0.5B-Instruct as the compressor, trained via full-parameter fine-tuning, while the downstream decoder was Llama-3-8B-Instruct, which remained frozen. To eliminate potential biases arising from model family differences, we additionally trained a compressor based on Llama-3.2-1B-Instruct, also using full-parameter fine-tuning.

We compare our approach against six competitive context compression methods: AutoCompressor (Chevalier et al., 2023), xRAG (Cheng et al., 2024), COCOM (Rau et al., 2025), ICAE (Ge et al., 2024), LLMingua-2 (Pan et al., 2024), and PCC (Dai et al., 2025).

Following a two-tier evaluation strategy, PCC serves as our primary controlled baseline: for pre-training and data-efficiency analyses we re-train it under the identical Qwen2.5-0.5B-Instruct backbone and data, so that the attention mask remains the only varying factor. The remaining methods adopt their own backbones and training settings as reported in prior work, and are included primarily to position PIC within the broader context-compression landscape rather than as strictly controlled comparisons.

Further training hyper-parameters, hardware configurations, and detailed baseline descriptions are provided in Appendix A.

4.2 Main Results

4.2.1 RAG-based Question Answering

We evaluate our compressor on four diverse QA datasets: SQuAD, HotpotQA (Yang et al., 2018b), AdversarialQA (Bartolo et al., 2020), and Natural Questions (NQ) (Karpukhin et al., 2020), using standard F_1 and Exact Match (EM) metrics. **Reference (w/o Context)** and **Reference (w/ Context)** denote generating responses using the downstream decoder conditioned solely on the question, and on the question augmented with uncompressed context, respectively.

As shown in Table 2, PIC achieves the highest average performance across all evaluated compression ratios. Compared to the most competitive similarly-scaled baseline, PCC-Lite, PIC surpasses it on almost all metrics, with a marginal exception

Table 2: Comparative performance of various compression methods across four QA datasets. Bold values indicate the best performance for each dataset within the same compression ratio category, and second-best results are underlined. Unless explicitly stated otherwise, PIC utilizes Qwen2.5-0.5B-Instruct as the default compressor backbone.

Compression Rate	Methods	SQuAD		HotPotQA		AdversarialQA		NQ		Average	
		F_1	EM	F_1	EM	F_1	EM	F_1	EM	F_1	EM
$1\times$	w/o Context	14.84	3.41	24.80	14.90	12.05	6.43	27.48	15.02	19.79	9.94
	w/ Context	79.81	59.97	64.60	51.12	56.10	38.67	64.64	51.38	66.29	50.29
$40\times$ $> 64\times$	AutoCompressor	21.46	0.35	16.29	0.29	14.09	2.00	25.57	0.63	19.35	0.82
	xRAG	18.19	3.46	27.51	16.29	13.75	3.47	38.06	20.80	24.38	11.01
$4\times$	ICAE	45.69	21.63	35.16	26.68	27.98	11.70	59.15	47.35	42.00	26.84
	LLMLingua-2	51.20	32.18	55.72	<u>44.18</u>	35.41	24.80	68.44	55.85	52.69	39.25
	COCOM-Lite	21.70	9.17	40.07	32.32	19.45	13.90	50.45	41.87	32.92	24.32
	PCC-Lite	<u>75.83</u>	<u>57.44</u>	50.37	42.20	50.36	<u>37.83</u>	<u>70.51</u>	<u>61.97</u>	<u>61.77</u>	49.86
	OURS (PIC)	79.03	60.14	<u>54.66</u>	46.61	<u>50.28</u>	38.13	71.82	62.47	63.95	51.84
$16\times$	COCOM-Lite	19.23	8.13	31.94	25.27	19.45	13.90	26.36	20.66	24.25	16.99
	PCC-Lite	56.82	37.72	35.67	27.88	40.94	<u>28.07</u>	62.56	52.73	49.00	36.60
	OURS (PIC)	59.41	39.86	41.05	33.24	39.98	27.73	<u>67.61</u>	<u>58.37</u>	<u>52.01</u>	<u>39.80</u>
	OURS (PIC-Llama)	<u>58.93</u>	<u>39.69</u>	<u>40.61</u>	<u>32.47</u>	<u>40.53</u>	29.00	68.17	58.90	52.06	40.02
$64\times$	PCC-Lite	37.89	22.66	20.62	14.27	27.02	16.30	36.82	29.18	30.59	20.60
	OURS (PIC)	40.84	24.36	30.54	23.07	29.54	19.40	57.96	49.15	39.72	29.00

in F_1 on AdversarialQA (though still leading in EM). Additionally, our EM score on HotpotQA is only slightly below the explicit hard-prompt method LLMLingua-2. Notably, despite utilizing different backbones, the performance discrepancy between PIC and PIC-Llama is minimal. These results solidly demonstrate that introducing a block-wise structural inductive bias is highly effective for soft prompt compression.

4.2.2 Cross-Chunk Reasoning on Multi-hop QA

Explicit block-wise masking inherently constrains direct cross-chunk attention during compression. To evaluate whether this impairs long-range dependency modeling, we additionally evaluate on multi-hop QA tasks (MultiQA (Ho et al., 2020) and MuSiQue (Trivedi et al., 2022)), which require aggregating evidence across multiple text spans and therefore naturally involve cross-chunk reasoning. As shown in Table 3, PIC outperforms the PCC baseline across all compression ratios on EM and nearly all F_1 comparisons. This demonstrates that causal memory-to-memory token passing effectively propagates sufficient context for downstream reasoning, overcoming the perceived limitation of local chunking.

Table 3: Performance on multi-hop QA tasks (MultiQA and MuSiQue).

Method	Rate	MultiQA		MuSiQue	
		EM	F_1	EM	F_1
w/o Context	1x	15.65	23.06	2.01	7.29
w/ Context	1x	32.05	44.61	19.80	30.02
PCC	4x	26.20	32.13	10.02	15.05
	16x	22.85	28.99	3.34	8.50
	64x	18.56	23.53	1.78	6.54
PIC (Ours)	4x	29.98	36.82	13.24	19.55
	16x	26.04	31.77	6.51	13.16
	64x	19.64	24.97	2.01	6.41

4.2.3 In-Context Learning

Unlike RAG, which emphasizes precise evidence retrieval and local information extraction, ICL relies on analogical reasoning from demonstration examples to infer latent task patterns. To evaluate this distinct capability, we assess our method on GSM8K (Cobbe et al., 2021), WSC (Levesque et al., 2012), and seven GLUE subsets (Wang et al., 2018). We compare PIC against zero-shot generation, uncompressed few-shot ICL (constrained to 150 and 750 tokens), and the $16\times$ compressed PCC baseline.

As detailed in Table 4, although PIC inevitably incurs some performance degradation relative to the uncompressed full-context baselines, PIC out-

Table 4: Performance Comparison on ICL Datasets (Accuracy %). “Comp.” denotes using compressed memory context. Bold indicates the highest accuracy between PCC and PIC.

Methods	GSM8K	SST-2	WSC	WNLI	RTE	QQP	CoLA	MRPC	QNLI
Zero-shot	77.81	92.32	57.69	56.34	53.07	50.41	81.21	66.67	55.32
ICL 150 Tokens	83.56	95.30	50.00	60.56	81.59	57.99	81.50	75.00	79.57
ICL 750 Tokens	86.10	95.30	46.15	70.42	86.64	65.96	70.37	33.09	75.53
16x-PCC Comp. 150 Tokens	90.11	89.45	43.27	57.75	52.71	63.00	73.63	35.54	49.06
16x-PCC Comp. 750 Tokens	91.44	86.12	47.12	45.07	53.79	57.59	70.18	31.62	54.00
OURS (16x Comp. 150 Tokens)	89.84	90.37	57.69	66.20	54.15	63.17	71.72	43.38	56.36
OURS (16x Comp. 750 Tokens)	90.64	85.09	65.38	56.34	56.32	61.79	70.57	41.67	55.92

performs the PCC baseline across most evaluated configurations. The gains are most pronounced on tasks that hinge on local relational reasoning, such as WSC, WNLI, QQP, and MRPC: for instance, on WSC PIC reaches 65.38% accuracy, exceeding PCC by a large margin and even surpassing the uncompressed 750-token baseline (46.15%). On a few holistic sentence-level tasks, such as CoLA and the 750-token setting of SST-2, PIC is marginally below PCC; these tasks require a whole-sentence judgment, and under block-wise compression a sentence and its label may fall into different chunks, so their relationship must be conveyed through compressed memory tokens. Overall, these results indicate that block-wise compression effectively preserves localized structural patterns and intra-context relations, with the remaining differences confined to holistic sentence-level judgments.

4.3 Block-wise Causal Mask Accelerates Compressor Learning

4.3.1 Pre-training Dynamics

To rigorously evaluate the learning efficiency and stability of the compressor, we compare the pre-training evaluation loss trajectories across 3 random seeds. As illustrated in Figure 4, PIC exhibits a noticeably faster convergence rate and a smaller standard deviation compared to the PCC baseline.

Further investigation (detailed in Appendix C) reveals that PCC’s high loss variance stems from its unconstrained global attention, which converges to substantially different spatial specialization patterns under different random seeds. Notably, runs where the implicitly learned mapping more closely resembles sequential chunking achieve a lower loss. This empirical evidence demonstrates that explicitly introducing our block-wise inductive bias not only accelerates training but also substantially reduces the architectural instability across different runs. For the complete, long-term pre-training loss

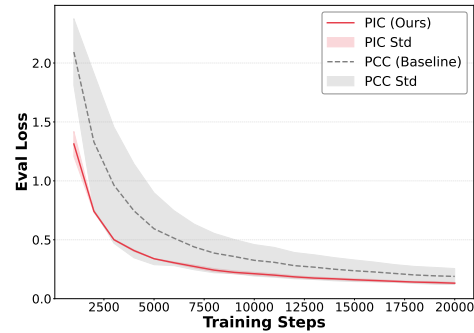


Figure 4: Comparison of evaluation loss during the first 20k pre-training steps across 3 random seeds. Lines represent the mean loss, while shaded regions indicate the standard deviation. PIC (red) achieves faster convergence and remarkably higher stability compared to the PCC baseline (gray).

trajectory of a single representative run, please refer to Appendix B Figure 9.

4.3.2 Data Efficiency Analysis

As observed in Appendix B Figure 9, after 30k training steps, both PIC and the PCC baseline approach convergence, rendering it challenging to discern capability differences based solely on pre-training reconstruction loss. Consequently, to thoroughly investigate data efficiency, we perform domain-specific fine-tuning on checkpoints saved at various pre-training intervals and evaluate their performance on downstream QA tasks. To ensure a fair comparison, we utilize the identical Qwen2.5-0.5B-Instruct backbone for both methods, isolating the attention mechanism as the sole independent variable.

Taking the $4\times$ compression setting as a representative example (Figure 5), PIC trained with merely 0.98 billion tokens (30k steps) already surpasses the peak performance of the PCC baseline trained with nearly 3 billion tokens (90k steps). By maintaining a substantial and consistent performance

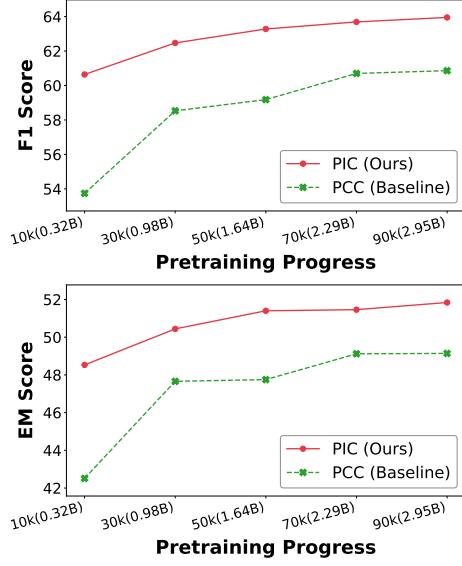


Figure 5: Relationship between Compressor Performance on Downstream QA Tasks and Pre-training Progress under $4\times$ compression. The x-axis represents the training steps alongside the corresponding consumed tokens. PIC demonstrates superior data efficiency compared to the baseline.

lead throughout the entire training process, these results clearly validate that explicitly enforcing a block-wise inductive bias significantly enhances data efficiency. (Detailed data efficiency results for $16\times$ and $64\times$ compression rates are provided in Appendix D.)

4.4 Analysis of Learned Embeddings

By calculating the cosine similarity between the memory embeddings generated by PIC and the original context token embeddings, we find that the learned attention patterns align closely with our imposed receptive field constraints: each memory token sequentially focuses intensively on the specific text chunk it is designated to compress. The detailed visual results are shown in Appendix Figure 13.

4.4.1 Impact of Tuning Strategy on Spatial Specialization

Louis et al. (2025) claim that the spatial specialization does not occur when the downstream decoder remains completely frozen. However, our findings indicate that with a completely frozen decoder, this phenomenon is fundamentally dictated by the compressor’s tuning method and structural bias. While both PCC and PIC achieve spatial specialization via full-parameter tuning, this specialization disappears under LoRA in standard global compressors,

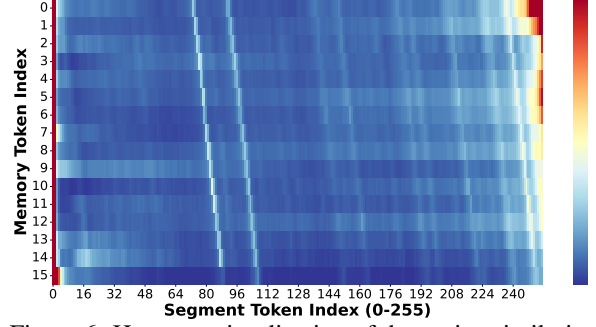


Figure 6: Heatmap visualization of the cosine similarity between generated memory embeddings generated by the LoRA tuning compressor and the embeddings of original context tokens. Red indicates higher similarity, while blue represents lower similarity.

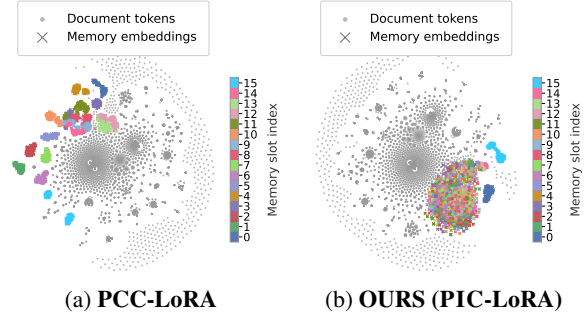


Figure 7: t-SNE visualization of compressed memory embeddings and original document tokens in the latent space of the downstream LLM.

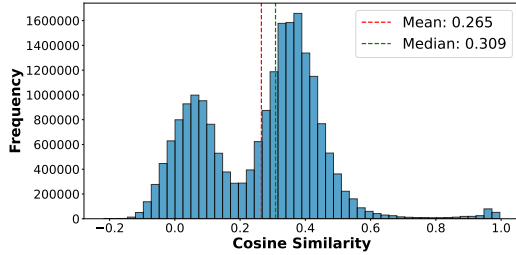
as visualized in Figure 6. Crucially, PIC’s explicit block-wise bias successfully recovers this structure under LoRA.

This restoration translates into massive efficiency gains: as shown in Table 5, after merely 10k pre-training steps PIC-LoRA drastically outperforms PCC-LoRA across all downstream QA metrics.

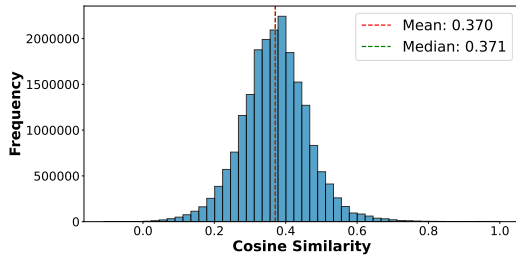
Furthermore, t-SNE visualizations of the LoRA-tuned compressors (Figure 7) reveal stark representational differences in the latent space. PCC’s embeddings severely fragment into isolated clusters dictated primarily by their memory slot index. Conversely, PIC maintains a significantly more centralized and cohesive distribution. This visual evidence further confirms that our block-wise inductive bias helps preserve essential representational structures, allowing LoRA to maintain comparable performance and spatial characteristics to full-parameter tuning.

Table 5: Performance comparison on downstream QA tasks. Compressors are fine-tuned using LoRA based on checkpoints saved after only 10k pre-training steps. The metrics reflect F_1 and EM scores (%).

Method	SQuAD		HotpotQA		AdversarialQA		NQ	
	F_1	EM	F_1	EM	F_1	EM	F_1	EM
PCC-LoRA	28.08	13.80	27.50	20.39	20.57	13.87	44.93	36.56
PIC-LoRA	52.63	33.22	38.55	30.34	37.32	24.90	62.52	52.43



(a) PCC



(b) OURS (PIC)

Figure 8: Distribution of pairwise cosine similarities between memory embeddings within the same memory slot. Compressors are fine-tuned on QA tasks.

4.4.2 Distributional Characteristics of Memory Embeddings

Analyzing pairwise cosine similarities between memory tokens (Figure 8) reveals stark distributional differences. PCC exhibits a problematic bimodal distribution: a heavy right tail implies token redundancy, while negative values and a low-similarity peak (0 - 0.2) suggest conflicting semantics and uninformative noise. Conversely, PIC approximates a normal distribution, structurally suppressing both redundancy and noise. This indicates that our block-wise inductive bias naturally enables more orthogonal, distinct, and information-rich memory representations.

5 Conclusion

Drawing inspiration from the chunking mechanism in human working memory and the observed spatial specialization of memory embeddings, we introduced PIC. By implementing a straightforward modification to the Transformer attention mask,

PIC explicitly constrains the receptive field of memory tokens to sequential local blocks. This structural inductive bias significantly reduces the optimization complexity inherent in compressor training. Empirical results across diverse downstream tasks demonstrate that PIC outperforms competitive baselines while simultaneously accelerating the training process by a substantial margin. Finally, we shed light on the intrinsic properties of the memory embeddings, revealing that the emergence of spatial specialization is closely tied to both tuning strategies and explicit structural priors. We anticipate these insights will inspire and guide future explorations in efficient context compression.

6 Limitations

While PIC demonstrates superior efficiency and performance in soft prompt compression, it inherently carries several limitations.

First, PIC adopts a fixed, uniform, non-overlapping chunking strategy based on a pre-defined compression ratio. This choice is deliberate: fixed chunking yields a static attention mask that enables stable and highly parallel compressor training, whereas overlapping chunks degrade downstream performance. The strategy nevertheless does not dynamically allocate more memory tokens to information-dense regions or fewer to redundant spans, which could result in sub-optimal memory utilization for highly heterogeneous texts. We leave adaptive allocation and learned routing to future work.

Second, explicitly constraining the receptive field via the block-wise causal mask prevents a memory token from directly attending to the raw tokens of preceding segments ($c_{<t}$). Although historical context is successfully propagated via the condensed memory states ($m_{<t}$), this indirect routing architecture might impose a theoretical ceiling on the model’s capacity for extremely fine-grained, cross-chunk exact string matching compared to full-

context global attention.

Finally, our empirical evaluation is currently constrained to model scales up to 1B parameters and context lengths up to 16K tokens due to compute limits. Exploring the scaling laws of PIC on massive context windows (e.g., 100K+ tokens) and substantially larger LLMs remains a crucial subject for future research.

7 Acknowledgments

This work is supported by Hunan Provincial Natural Science Foundation - Young Scientists Fund (Category B) (No. 2026JJ40096), Hunan Provincial Foundation for Innovative Research Groups (No. 2026JJ10007), the National Natural Science Foundation of China (Project No.62406336) and the Research Foundation from NUDT (Grant No. ZK2023-12). Jie Yu is the corresponding author.

References

- Alan Baddeley. 1992. Working memory. *Science*, 255(5044):556.
- Omri Barak and Misha Tsodyks. 2014. Working models of working memory. *Current opinion in neurobiology*, 25:20–24.
- Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the ai: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.
- Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Yu Neng Chuang, Tianwei Xing, Chia Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. 2024. Learning to compress prompt in natural language formats. In *2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2024*, pages 7749–7760. Association for Computational Linguistics (ACL).
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Nelson Cowan. 2010. The magical mystery four: How is working memory capacity limited, and why? *Current directions in psychological science*, 19(1):51–57.
- Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28715–28732.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).
- Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.
- Anirudh Goyal and Yoshua Bengio. 2022. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 478(2266):20210068.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmllingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.
- Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmllingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.
- Yurii Kuratov, Mikhail Arkhipov, Aydar Bulatov, and Mikhail Burtsev. 2025. Cramming 1568 tokens into a single vector and back again: Exploring the limits of embedding space capacity. In *Proceedings of the*

- 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 19323–19339.
- Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. *KR*, 2012(13th):3.
- Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhui Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.
- Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 6342–6353.
- Zongqian Li, Yinhong Liu, Yixuan Su, and Nigel Collier. 2025. Prompt compression for large language models: A survey. In *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.
- Maxime Louis, Hervé Déjean, and Stéphane Clinchant. 2025. **PISCO: Pretty simple compression for retrieval-augmented generation**. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15506–15521, Vienna, Austria. Association for Computational Linguistics.
- Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352.
- 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. 2024. Llmllingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 963–981.
- Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.
- David Rau, Shuai Wang, Hervé Déjean, Stéphane Clinchant, and Jaap Kamps. 2025. **Context embeddings for efficient answer generation in retrieval-augmented generation**. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM ’25, page 493–502, New York, NY, USA. Association for Computing Machinery.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♪ musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.
- Baosong Yang, Zhaopeng Tu, Derek F Wong, Fandong Meng, Lidia S Chao, and Tong Zhang. 2018a. Modeling localness for self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4449–4458.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018b. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.
- Weishun Zhong, Mikhail Katkov, and Misha Tsodyks. 2024. Synaptic theory of chunking in working memory. *arXiv preprint arXiv:2408.07637*.

A Detailed Experimental Setup

A.1 Additional Implementation Details

In the pre-training stage, the learning rate was set to 1×10^{-4} for all models. Training the PIC model (with the Qwen backbone) for one full epoch on the pre-training dataset took approximately 170 hours on a single node with two 48GB RTX 4090 GPUs. For all inference experiments, greedy decoding was applied to the LLM decoder.

A.2 Baselines Description

We compare our approach against six competitive context compression methods:

AutoCompressor (Chevalier et al., 2023): Adapts a pre-trained Llama-2-7b-hf to generate summaries embeddings, achieving a high compression ratio of 40x.

xRAG (Cheng et al., 2024): Utilizes Mistral-7B-Instruct-v0.2 as the downstream LLM. It focuses on training the connector (Convert module) while keeping other components frozen, fine-tuned on multiple downstream datasets.

COCOM (Rau et al., 2025): Employs a compressor based on bert-base-uncased for pre-training and fine-tunes the downstream LLM (Mistral-7B-Instruct-v0.2).

ICAE (Ge et al., 2024): Features a compressor pre-trained on Mistral-7B to compress context into memory slots, utilizing Mistral-7B-Instruct-v0.2 as the downstream LLM.

LLMLingua-2 (Pan et al., 2024): A hard prompt compression method based on token classification (using XLM-RoBERTa-large). It is evaluated with Llama-3-8B-Instruct as the target LLM.

PCC (Dai et al., 2025): PCC-Lite utilizes a compressor pre-trained on GPT2-Large and uses Llama-3-8B-Instruct as the downstream decoder.

To ensure a fair comparison, we strictly adhered to the experimental settings established in PCC and directly compared our results against the metrics reported in their paper.

B Extended Pre-training Dynamics

We also provide the complete, long-term pre-training reconstruction loss trajectory for a single representative run. As shown in Figure 9, PIC maintains a sustained convergence advantage throughout the initial phase, achieving an effective acceleration of approximately $1.1 \times$ to $1.3 \times$ compared to the baseline before both models approach convergence.

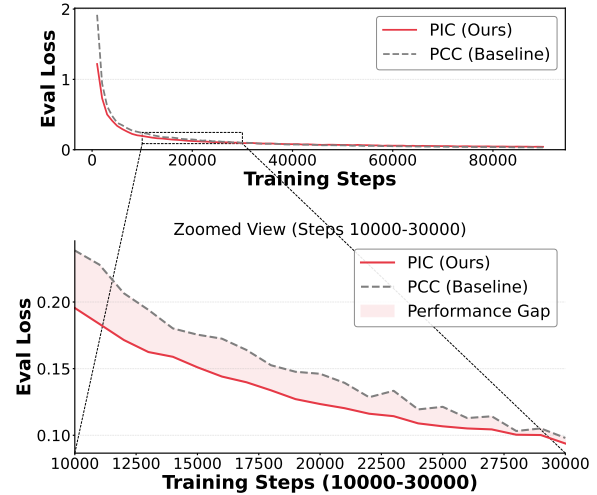


Figure 9: Complete trajectory of reconstruction loss during pre-training. The upper plot illustrates the overall process, while the lower plot presents a zoomed-in view of steps 10k to 30k, highlighting the sustained faster convergence of PIC (red) compared to the PCC baseline (dashed gray).

C Analysis of PCC Training Variance

As observed in the pre-training dynamics (Section 4.3.1), the PCC baseline exhibits significant variance in evaluation loss across different random seeds. To uncover the underlying architectural cause, we visualize the attention weight heatmap between memory tokens and original context tokens of converged PCC models trained under different random seeds.

As shown in Figure 10, different weight initializations drive PCC’s unconstrained global attention to settle into entirely different spatial specialization mappings. Crucially, we observe a strong correlation between the structural regularity of these mappings and the final evaluation loss: the closer the local regions of the implicitly learned mapping approach a strict, sequential block-wise chunking organization, the lower the resulting pre-training loss.

This phenomenon suggests that sequential block-wise extraction serves as a highly effective structural prior for soft prompt compression. Under standard global compression, the model expends significant optimization effort searching for a suitable layout, which can lead to sub-optimal local minima depending on the initialization. By explicitly enforcing this block-wise constraint in PIC, we mitigate this unstable search process, enabling the compressor to more consistently achieve lower loss and improved data efficiency.

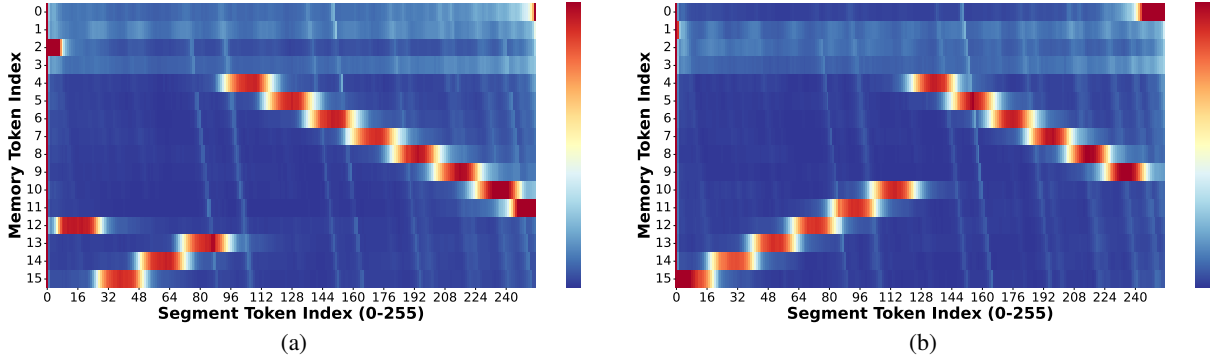


Figure 10: Comparison of memory-to-context attention heatmaps generated by converged PCC models trained under different random seeds. The model converges to more optimal checkpoints (lower loss) when its learned spatial specialization naturally approximates a sequential block-wise structure.

D Data Efficiency Analysis on Other Compression Rate

We further investigated the data efficiency of our method under different compression regimes. Figure 11 presents the performance trajectories for 16x and 64x compression rates. Consistent with the 4x results discussed in the main text, PIC demonstrates a superior learning curve. For both compression ratios, our method achieves higher downstream performance (EM and F_1 scores) at earlier pre-training steps compared to the baseline, confirming that the block-wise constraint accelerates the acquisition of effective compression capabilities regardless of the compression rate.

E Detailed Analysis of Spatial Specialization

We also analyze the t-SNE visualizations of the memory embeddings produced by the fully fine-tuned compressors. As illustrated in Figure 14, PIC exhibits more compact clusters compared to PCC. Furthermore, compared with our reproduced PISCO model (Louis et al., 2025), we find that PIC achieves a similar spatial structure in the latent space.

F Inference Efficiency

F.1 End-to-End Latency Analysis

To quantify the computational efficiency of our method, we measured the end-to-end inference latency across different context lengths and compression ratios. As shown in Table 6, PIC significantly reduces inference latency (by 2-4 $\times$) compared to the uncompressed baseline. Using a smaller compressor minimizes overhead, accelerating inference without adding any extra cost compared to baseline

Table 6: End-to-end latency (milliseconds) across varying input lengths and compression rates.

Batch $\times$ Input	Rate	Comp. Time	Prefill	Decode	Total Time
8×1024	Origin	-	1067.67	1022.29	2089.96
	4x	138.80	258.61	746.99	1144.40
	16x	116.16	68.61	668.37	853.14
	64x	112.57	28.21	655.03	795.81
16×1024	Origin	-	2181.19	1516.71	3697.90
	4x	262.44	548.79	870.34	1681.57
	16x	225.47	133.67	752.76	1111.90
	64x	209.07	38.26	718.02	965.35
8×2048	Origin	-	2200.18	1453.37	3653.55
	4x	276.35	550.52	816.52	1643.39
	16x	233.52	132.11	696.58	1062.21
	64x	224.47	36.44	649.59	910.50
4×4096	Origin	-	2266.72	1438.84	3705.56
	4x	340.39	552.60	791.15	1684.14
	16x	287.78	130.69	688.38	1106.85
	64x	283.22	34.67	636.66	954.55
2×8192	Origin	-	2396.49	1434.53	3831.02
	4x	569.88	543.64	791.68	1905.20
	16x	516.51	125.37	688.53	1382.88
	64x	518.70	34.36	647.45	1200.51

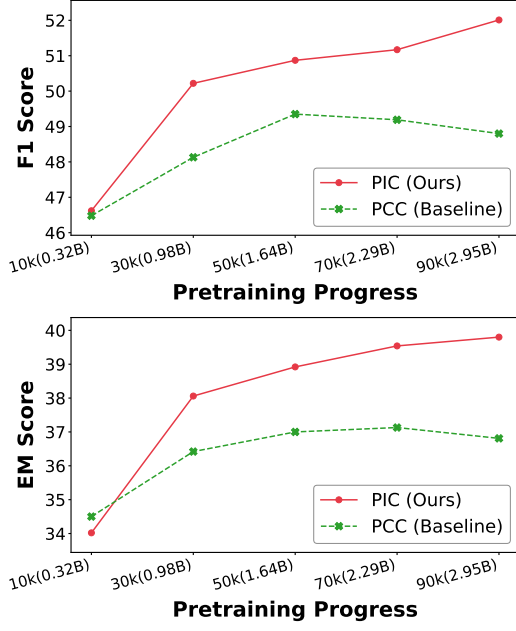
methods like PCC. All latency measurements were conducted on a single NVIDIA RTX 4090 GPU with 48GB of memory, and the downstream LLM’s generation was capped at 32 tokens to standardize throughput comparisons.

F.2 GPU Memory Footprint Analysis

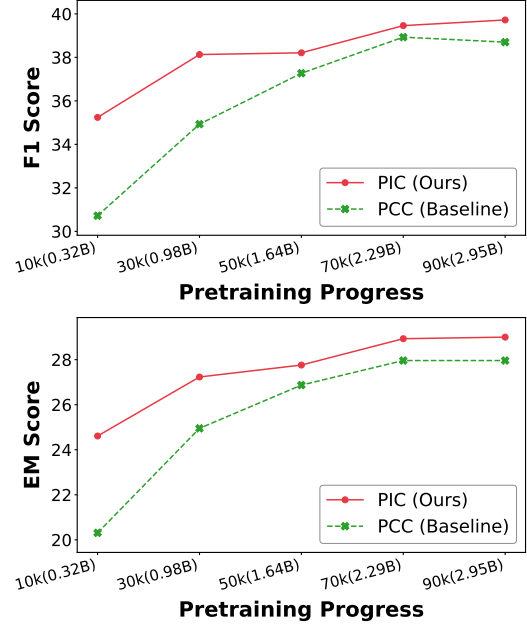
We further evaluated the peak GPU memory footprint. As shown in Table 7, compression successfully prevents Out-Of-Memory (OOM) errors on lengthy inputs (e.g., 8×8192 and 8×16384), enabling high-throughput deployment under severe memory constraints.

G Ablations on Chunk Granularity

To further validate our design choices regarding chunk granularity, we conducted an ablation study using the 16 $\times$ compression rate. In standard



(a) 16x Compressor



(b) 64x Compressor

Figure 11: Relationship between Compressor Performance on Downstream QA Tasks and Pre-training Steps. The x-axis represents the number of tokens corresponding to the pre-training steps (converted to 1 epoch equivalent), and the y-axis represents the EM or F_1 score.

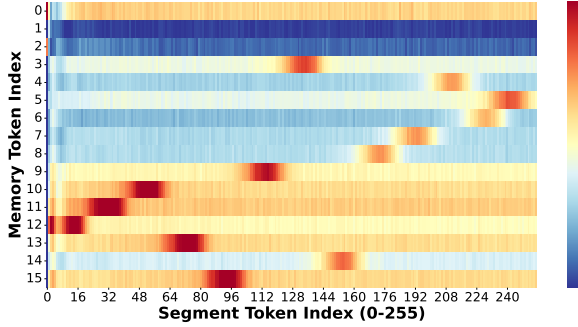


Figure 12: Heatmap visualization of the cosine similarity between generated memory embeddings generated by the fully trained compressor (90k steps) and the embeddings of original context tokens. Red indicates higher similarity, while blue represents lower similarity.

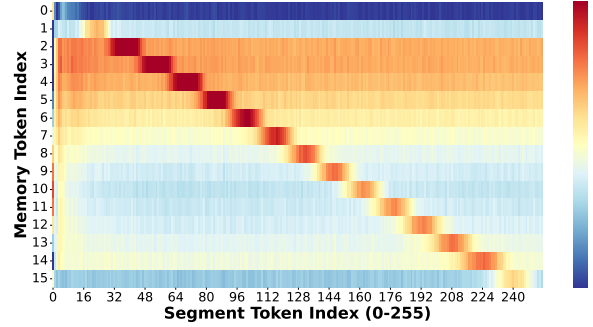


Figure 13: Heatmap visualization of the cosine similarity between memory embeddings (PIC) generated by the Fully Trained Compressor (90k Steps) and original token embeddings. Red indicates higher similarity, while blue represents lower similarity.

PIC (granularity 1), there is a strict one-to-one mapping between each memory token and its corresponding text chunk. Here, we relaxed this constraint, testing granularities of 2 and 4.

We trained the $16\times$ PIC models using these granularities for 10k steps and fine-tuned them on downstream QA tasks. As shown in Table 8, increasing chunk granularity consistently leads to worse downstream performance.

Our internal visualization of spatial specialization patterns supports this outcome: within larger chunk granularity, the internal attention patterns lose their strict sequential alignment. Instead, they

randomly converge to local sub-patterns similar to the unconstrained PCC baseline. This disruption of structural sequentiality reintroduces optimization uncertainty, fundamentally validating our choice of fine-grained, one-to-one chunk mapping in the standard PIC design.

H Ablations on Overlapping Chunking

To further assess the flexibility of our fixed chunking design, we tested overlapping chunking, a straightforward extension in which adjacent chunks share boundary tokens. Under $16\times$ compression,

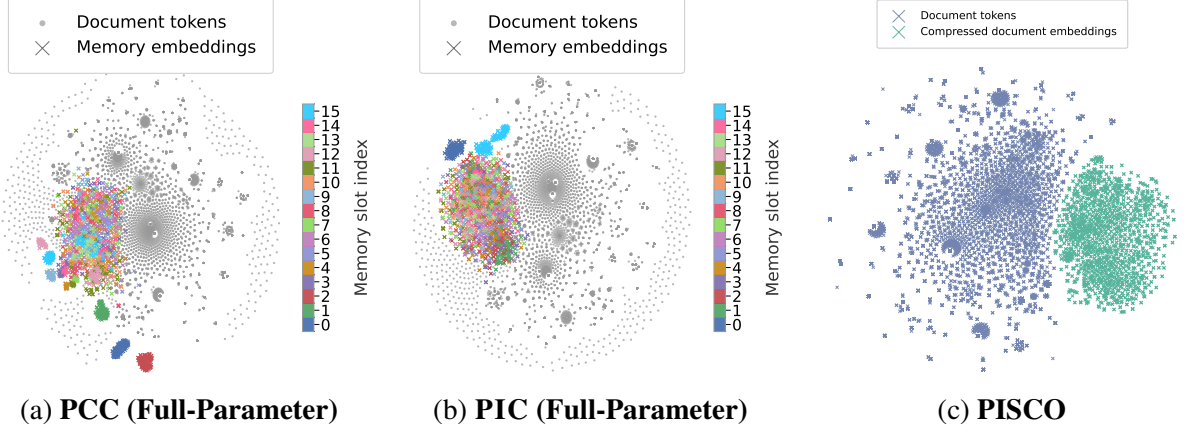


Figure 14: t-SNE visualizations of memory embeddings generated by different compressor architectures. PIC exhibits more compact clusters compared to PCC and achieves a spatial structure similar to PISCO in the latent space.

Table 7: GPU Peak Memory (GB) across varying input lengths and compression rates.

Batch $\times$ Input	Rate	GPU Peak Mem (GB)
8×512	Origin	20.13
	4x	18.91
	16x	18.56
	64x	18.47
8×1024	Origin	23.71
	4x	18.91
	16x	18.56
	64x	18.47
8×2048	Origin	30.87
	4x	20.23
	16x	18.56
	64x	18.47
8×4096	Origin	45.20
	4x	23.87
	16x	18.57
	64x	18.48
8×8192	Origin	OOM
	4x	31.16
	16x	20.22
	64x	18.48
8×16384	Origin	OOM
	4x	OOM
	16x	23.79
	64x	18.48

we evaluated 25% overlap (two additional tokens per side) and 50% overlap (four additional tokens per side). After 10k pre-training steps and downstream QA fine-tuning, overlapping chunks consistently degraded performance, as shown in Table 9.

These results indicate that overlapping chunks force each memory token to attend to more raw tokens, effectively raising the compression rate borne by a single memory embedding and thereby degrading downstream performance without any obvious compensating benefit. We therefore retain fixed,

Table 8: Ablations on chunk granularity at $16\times$ compression rate. Metrics are averaged across six QA datasets (SQuAD, HotpotQA, AdversarialQA, NQ, MultiQA, MuSiQue).

Granularity	Average (F_1)	Average (EM)
1 (Standard PIC)	38.14	27.56
2	36.28	26.94
4	34.85	25.40

Table 9: Performance of overlapping chunking under $16\times$ compression. Metrics are averaged across the four downstream QA datasets (SQuAD, HotpotQA, AdversarialQA, NQ).

Method	Average (F_1)	Average (EM)
PIC (no overlap)	46.62	34.02
PIC (overlap-2)	46.01	33.93
PIC (overlap-4)	43.10	30.37

non-overlapping uniform chunking as the standard PIC design.

I Deployment & Routing Strategies

It is critical to determine optimal scenarios for compression deployment. To this end, we first partition the QA datasets based on their context lengths, as detailed in Table 10. Through a decision boundary analysis spanning Context Length $\times$ Compression Ratio on the partitioned QA data (as shown in Table 11), we identified three operational regions:

- **Short Contexts** ($L \leq 512$): 4x compression consistently outperforms the uncompressed 1x baseline. PIC actively acts as a semantic information refiner/denoiser here.
- **Medium Contexts** ($512 < L \leq 4096$):

Table 10: Distribution of input lengths and sample counts.

length	sample_num
[0, 64]	222
(64, 128]	2697
(128, 256]	6289
(256, 512]	2646
(512, 1024]	2191
(1024, 2048]	4798
(2048, 4096]	749
(4096, 8192]	261
(8192, 16384]	76

We observe the expected accuracy-efficiency tradeoff where $1x \geq 4x \geq 16x$. PIC still provides significant latency reduction but incurs minor information loss compared to $1x$.

- **Long Contexts** ($L > 4096$): A higher compression ratio ($16x$) surprisingly outperforms $4x$, as condensing contexts to optimal decoder processing lengths avoids "lost-in-the-middle" attention degradation.

Based on these findings, we propose a lightweight, Rule-Based Length Routing heuristic dynamically executed at runtime for mixed workloads:

- 1) If $L \leq 512$, apply $4x$ compression to maximize accuracy and speed.
- 2) If $512 < L \leq 4096$, use $1x$ if accuracy is paramount, or gracefully fallback to $4x$ under high system load.
- 3) If $L > 4096$, apply $16x$ compression to prevent OOM errors and actively circumvent long-context degradation.

Table 11: PIC performance (F_1 / EM) across input lengths and compression rates.

Length Interval	Rate	PIC- F_1	PIC-EM
[0, 64]	1x	50.03	33.78
	4x	68.07	58.56
	16x	57.83	47.75
	64x	48.02	36.04
(64, 128]	1x	64.08	46.42
	4x	70.16	54.36
	16x	59.30	42.27
	64x	42.20	26.92
(128, 256]	1x	65.74	48.16
	4x	68.36	53.27
	16x	56.62	40.31
	64x	40.61	25.81
(256, 512]	1x	67.67	48.11
	4x	67.33	51.81
	16x	55.59	39.19
	64x	43.50	27.55
(512, 1024]	1x	74.09	61.43
	4x	66.80	57.46
	16x	59.06	49.38
	64x	50.46	40.94
(1024, 2048]	1x	72.21	60.80
	4x	58.94	50.90
	16x	47.47	39.66
	64x	37.60	29.93
(2048, 4096]	1x	71.87	58.34
	4x	68.70	59.01
	16x	61.25	53.00
	64x	48.39	41.66
(4096, 8192]	1x	71.68	54.02
	4x	64.35	52.11
	16x	66.01	54.02
	64x	43.45	36.40
(8192, 16384]	1x	61.78	44.74
	4x	51.05	36.84
	16x	51.68	36.84
	64x	33.42	27.63