LIFT: Improving Long Context Understanding of Large Language Models through Long Input Fine-Tuning

Anonymous Authors[1]

Abstract

Long context understanding remains challenging for large language models due to their limited context windows. This extended abstract presents **Long Input Fine-Tuning (LIFT)**, a novel framework for long-context modeling that can improve the long-context performance of arbitrary short-context LLMs by dynamically adapting model parameters based on the long input. Importantly, LIFT, rather than endlessly extending the context window size to accommodate increasingly longer inputs **in context**, chooses to store and absorb the long input **in parameter**. By absorbing the long input into model parameters, LIFT allows short-context LLMs to answer questions even when the required information is not provided in the context during inference. Furthermore, we introduce Gated Memory, a specialized attention adapter that automatically balances long input memorization and the original in-context learning (ICL) capabilities. We provide a comprehensive analysis of the strengths and limitations of LIFT on long context understanding, offering valuable directions for future research.

1. Introduction

Large Language Models (LLMs), such as GPT-4 (Achiam et al., 2023), have revolutionized the field of natural language processing. Long input, which can span up to millions of tokens, is common in real-world applications, including long books (Kočiskỳ et al., 2018), accounting documents (Li et al., 2024), high-resolution videos (Wu et al., 2024; Tapaswi et al., 2016), and audio signals (Yang et al., 2024). However, it's hard for LLMs to capture the overall information within long input due to the quadratic complexity of the attention mechanism and the heavy memory overhead caused by storing KV cache. Moreover, it is challenging to capture long dependencies (Li et al., 2023) among pieces of information scattered throughout long inputs.

To address these challenges: long-context post-training (Chen et al., 2023; Peng et al., 2023) adapts LLMs to long context via fine-tuning them on corpora of long sequences, but both the post-training and the inference of these models are resource-intensive; Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Xu et al., 2023) and prompt compression (Jiang et al., 2023; El-Kassas et al., 2021) preprocess long input via retrieval or compression, but the effectiveness is limited by the quality of the retrieved/compressed information; memory-augmented LLMs (Wang et al., 2023; 2024b; 2025) usually memorize previous hidden states with external modules. Please refer to Appendix A for a detailed discussion of related work.

In this extended abstract, we present a novel framework, **L**ong **I**nput **F**ine-**T**uning (LIFT), designed to enhance the long-context capabilities of arbitrary short-context models by adapting model parameters to the long input at test time. With our proposed Gated Memory adapter, LIFT allows LLMs to efficiently fine-tuning and decoding, and balancing in-parameter and in-context knowledge. Empirically, it improves LLMs' performance on popular long-context tasks.

2. Method

In this section, we introduce LIFT, a framework improving LLMs' long context understanding through long input fine-tuning (Figure 1 (a)). The details are in Appendix B.

2.1. Training with segmented long inputs

We propose a novel way to memorize long inputs by storing them into LLMs' parameters via fine-tuning. We formulate the memorization task as a language modeling task. However, it is challenging for short-context LLMs to adapt to a long sequence. Besides, it leads to quadratic complexity w.r.t. the input length. Let the tokenized long input be $\mathbf{x}$ of length L. One straightforward way is to cut $\mathbf{x}$ into non-overlapping short segments. However, it fails to capture the

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

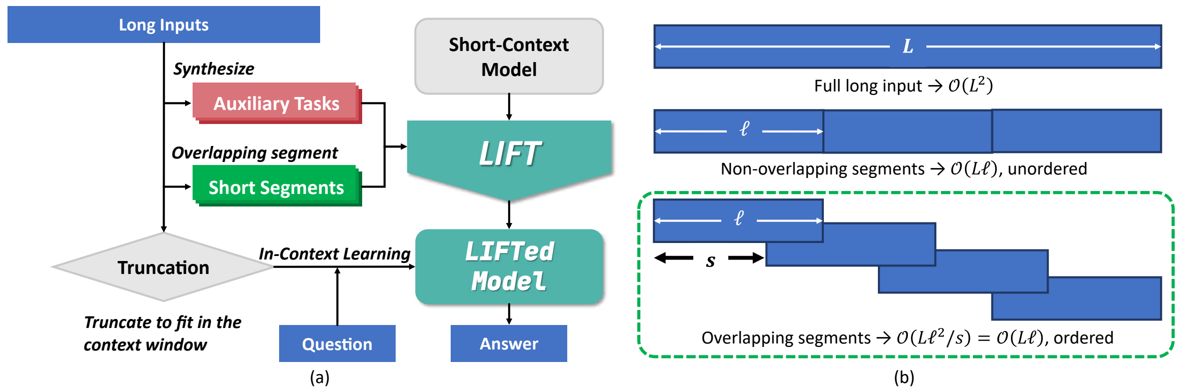


Figure 1: (a) The workflow of LIFT. (b) A comparison among the segmentation methods. LIFT adopts overlapping segmentation to avoid quadratic complexity while preserving the order.

sequentiality of the long input since the model cannot infer the *correct order* of the segments.

To address this, we alter the long-input segmentation with certain overlaps between the adjacent segments as illustrated in Figure 1 (b). By overlapping the tail of one segment with the head of the next, the model can better preserve the sequential structure of the input while maintaining linear complexity w.r.t. the input length. Ideally, *if the model learns to generate the tail of a segment, it should be able to seamlessly continue into the next segment*. Let the segments be $\mathbf{x}_{l_1:r_1}, \ldots, \mathbf{x}_{l_K:r_K}$. The objective is

$$\mathcal{L}_S = -\sum_{i=1}^{K}\sum_{j=l_i}^{r_i} \log P(x_j|\mathbf{x}_{l_i:j-1};\theta). \tag{1}$$

2.2. Joint training with auxiliary tasks

While training on the long input helps the model memorize the input, it probably degrades other abilities, such as instruction-following. Moreover, successfully memorizing the input does not necessarily indicate that the model can *reason effectively based on it*.

It is shown that *reading while questioning* is an effective method for improving the comprehension of knowledge for humans (Robinson, 1946). Motivated by this, we propose *synthesizing auxiliary question-answering (QA) tasks* based on the long input, denoted as $(\mathbf{q}_i, \mathbf{a}_i)_{i=1}^m$, and training the LLM on them. These QAs can be simple details such as specific people, time, locations of events, or more general reading comprehension ones. The objective function of the auxiliary tasks is

$$\mathcal{L}_{AT} = -\sum_{i=1}^{m} \log P(\mathbf{a}_i \mid \mathbf{q}_i;\theta). \tag{2}$$

Following the mechanism of mix training (Allen-Zhu & Li, 2023), which asserts that LLMs can only learn to perform inference based on $\mathbf{x}$ when trained simultaneously on both $\mathbf{x}$ and $(\mathbf{q}_i, \mathbf{a}_i)_{i=1}^m$, we propose jointly optimizing the two objective functions, i.e.,

$$\mathcal{L}_{MIX} = \mathcal{L}_S(\mathbf{x};\theta) + \mathcal{L}_{AT}((\mathbf{q}_i, \mathbf{a}_i)_{i=1}^m;\theta) \tag{3}$$

In our experiments, we extract several short segments from $\mathbf{x}$ and use a pretrained LLM to generate QA pairs based on the segments. The prompts to generate such QAs are given in Appendix B.

To align the formats of segmented language modeling and auxiliary tasks as well as maximally leverage the in-context knowledge, we further propose a Contextualized Training (CT) method which provides a piece of context from the original long input for both the language modeling and auxiliary tasks. The details are included in Appendix D. CT further improves the downstream performance.

2.3. Gated Memory Architecture

To efficiently apply LIFT, we aim to use a parameter-efficient fine-tuning (PEFT) rather than full-parameter fine-tuning. Existing representative PEFT methods such as LoRA (Hu et al., 2021) and PiSSA (Meng et al., 2024) are not specifically designed for long context tasks. Therefore, we propose a novel **Gated Memory** adapter working very well in the LIFT framework.

One key intuition behind LIFT is to store long input that cannot fit into the context window into model parameters. To achieve this, we hypothesize that the model has access to the long input $\mathbf{x}'$ and the prompt $\mathbf{x}$ during inference and the tokens of $\mathbf{x}$ can attend to the tokens of $\mathbf{x}'$ in the attention modules. We decompose the attention output in the hypothetical theme into parts related to $\mathbf{x}$ only and the other parts related to both $\mathbf{x}$ and $\mathbf{x}'$.

$$\begin{aligned}\mathrm{attn}(\mathbf{q},\mathbf{k}',\mathbf{k},\mathbf{v}',\mathbf{v}) = g_{\mathbf{k}'}(\mathbf{q}) \odot m_{\mathbf{k}',\mathbf{v}'}(\mathbf{q}) + \\ \big(1-g_{\mathbf{k}'}(\mathbf{q})\big) \odot \mathrm{attn}(\mathbf{q},\mathbf{k},\mathbf{v}),\end{aligned} \tag{4}$$

where $(\mathbf{q}',\mathbf{k}',\mathbf{v}')$ and $(\mathbf{q},\mathbf{k},\mathbf{v})$ are the query-, key- and value-vectors of $\mathbf{x}'$ and $\mathbf{x}$, respectively.

Table 1: Performance on LooGLE. We evaluate the accuracy of the methods on LooGLE short-dependency QA (ShortQA) and long-dependency QA (LongQA). Comprehension & reasoning, multiple info retrieval, computation, and timeline reorder are the subtasks in LongQA and we evaluate the accuracy on each of them.

Methods	ShortQA	LongQA	Comprehension & Reasoning	Multiple info retrieval	Computation	Timeline reorder
MemoryLLM	33.06	20.44	29.31	15.53	8.00	18.14
LlamaIndex$_{\text{(Llama-3)}}$	41.93	21.07	33.00	17.11	12.00	9.77
LlamaIndex$_{\text{(Gemma-2)}}$	42.95	22.98	31.03	18.16	11.00	21.86
ICL$_{\text{(Llama-3)}}$	44.49	15.44	25.37	15.26	5.00	1.86
LIFT$_{\text{(Llama-3)}}$	47.51	29.97	39.90	**27.89**	**17.00**	17.21
ICL$_{\text{(Gemma-2)}}$	37.37	29.79	36.95	21.58	10.0	**40.00**
LIFT$_{\text{(Gemma-2)}}$	**50.33**	**31.24**	**40.39**	27.11	12.00	30.23

Table 2: Performance on LongBench. The scores of MemoryLLM marked with * are the reused experimental results from the original paper.

Methods	Musique	NarrativeQA	Qmsum	GovReport	PassageRetrievalEN
MemoryLLM	13.47*	20.64*	20.98	20.89	9.03
LlamaIndex$_{\text{(Llama-3)}}$	18.23	17.21	21.95	28.03	59.38
ICL$_{\text{(Llama-3)}}$	**26.89**	19.45	21.64	30.18	58.33
LIFT$_{\text{(Llama-3)}}$	21.19	**23.33**	**23.07**	**33.62**	**62.50**

By absorbing $\mathbf{x}'$, as well as $\mathbf{q}', \mathbf{k}', \mathbf{v}'$, into the parameters of g and m, we derive the following Gated Memory adapter for attention modules:

$$f_\phi(\mathbf{q},\mathbf{k},\mathbf{v}) = g_\phi(\mathbf{q})\odot m_\phi(\mathbf{q}) + (1-g_\phi(\mathbf{q}))\odot \mathrm{attn}(\mathbf{q},\mathbf{k},\mathbf{v}),$$

where attn is the original attention module and ϕ are the parameters of the Gated Memory adapter. The architecture is illustrated in Figure 2. Please refer to Appendix C for the details of the derivation of Gated Memory.

One can use Equation (4) as the distillation targets to train g and m, yet this way is too expensive as it requires computing attention over the long input. Alternatively, we choose to train these adapters *end-to-end* using the LIFT objective in Equation (3), i.e., just use them normally as standard adapters.

Compared to LoRA, Gated Memory allows the model to recover the original model by setting $g_\phi(\mathbf{q})$ to 0 when the prompt is irrelative to the long input, thereby solving the task using only the in-context knowledge. In contrast, existing PEFT methods like LoRA and PiSSA fail to control the influence of adapters, risk overfitting the long input, and may damage the original capabilities too much.

3. Experiments

3.1. Setup

We evaluate LIFT on two popular long-context benchmarks, LooGLE (Li et al., 2023) and LongBench (Bai et al., 2023), covering a wide variety of application scenarios. LIFT is compared with baselines including truncated ICL, RAG (represented by LlamaIndex (Liu, 2022)), and memory-augmented models (represented by MemoryLLM (Wang et al., 2024b)). To evaluate the universality of LIFT across different models, we select two foundation models, Llama-3 and Gemma-2, both of which have 8K context windows. Please refer to Appendix G, F for the details about the evaluation metrics, the reproduction of the baselines,

We use **ICL** to denote truncating the long input by retaining only the beginning and end of texts within the context window of the base model, and use **LIFT** to denote first fine-tuning the base LLM using the Gated Memory adapter with the objective in Equation (3) and then use the above truncated ICL.

3.2. Results on LooGLE and LongBench

As shown in Table 1, LIFT consistently outperforms ICL often by large margins on the overall scores on both LongQA and ShortQA tasks across the two LLMs. Particularly, it shows notable improvement in GPT4-score from 15.44 to 29.97 on Llama-3 in LongQA and from 37.37 to 50.33 on Gemma-2 in ShortQA. Notably, Llama-3 benefits more from LIFT than from LlamaIndex, consistently outperforming LlamaIndex and improving the GPT4-score on LongQA from 21.07 to 29.97. Moreover, compared to ICL, LIFT significantly improves the LLMs performance on LongQA, where Llama-3 improves with over 50% gain on all the sub-

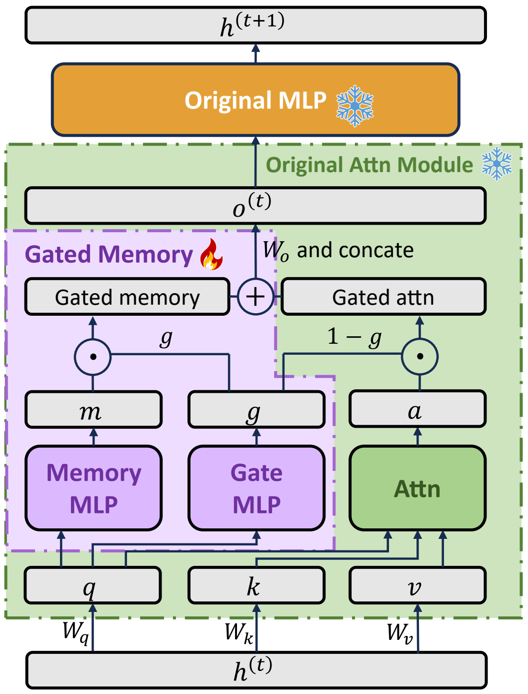


Figure 2: The architecture of **Gated Memory**. The purple part is the added adapter "gated memory" to fit the out-of-context attention; the green part is the original attention module. During training, only the gated memory part is trained. Other parameters are fixed.

tasks and Gemma-2 outperforms ICL on 3 out of 4 subtasks.

Table 2 presents the results across five representative tasks with extremely long inputs in LongBench. LIFT outperforms ICL and other baselines on 4 out of 5 subtasks. Please refer to Appendix I for additional analysis to the results on LooGLE and LongBench.

3.3. Efficiency and Ablation Study

We perform additional experiments to verify the efficiency of LIFT and the effectiveness of the designs.

To evaluate the efficiency of LIFT, we compare the following settings: LIFT with truncated ICL (denoted as `LIFT`), which is the same setting as that in LooGLE and LongBench test, LIFT without ICL (denoted as `LIFT (Llama 3.1)`), and long-context ICL (denoted as `ICL`). Empirically, as illustrated in Figure 3, LIFT starts to outperform ICL in decoding time when generating more than 1500 tokens with an input of 20K tokens. Please refer to Appendix J for a detailed analysis to the efficiency of LIFT.

For ablation study on Gated Memory, we conduct a comparison between Gated Memory, LoRA and PiSSA (Meng et al.). Specifically, using the same training framework (including task designs, objective functions, and scheduling

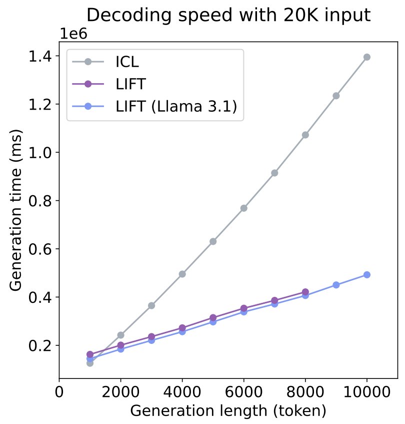


Figure 3: The efficiency test. The figure illustrates the decoding speed comparison between LIFT and ICL given inputs of length 20K.

configurations) of LIFT, we fine-tune the model separately with the Gated Memory, LoRA, and PiSSA adapters. To evalute the effectiveness of auxiliary tasks, we compare LIFT with 0, 10, and 20 auxiliary tasks on LooGLE. At last, we compare LIFT with contextualized training and LIFT with conventional language modeling. Empirically, the results suggest each of our design is effective. Please refer to Appendix E for the details of the ablation study.

4. Conclusion, Limitations, and Future Work

We proposed a novel framework, **L**ong-**I**nput **F**ine-**T**uning (**LIFT**), to enhance LLMs' long-context understanding. LIFT dynamically adapts LLMs to long inputs by efficiently fine-tuning the model parameters and utilizing the in-parameter knowledge to improve long-context performance. Experimental results across popular benchmarks like LooGLE and LongBench demonstrate that LIFT greatly improves short-context LLMs' ability to solve long-context tasks.

However, LIFT with truncated ICL is insufficient for tasks that demand precise information extraction from extended contexts, such as the Needle in a Haystack (NIAH) task (Appendix K). Although LIFT allows LLMs to memorize the input via contextualized training, the LLMs can not extract their parametric knowledge effectively, with accuracy lower than 50% on LooGLE. While auxiliary tasks help LLMs to extract parametric knowledge, the benefit comes with extensive computational costs and they must be aligned with test tasks to achieve optimal performance. LIFT is a fascinating concept because humans similarly transform short-term memory into long-term memory, much like LIFT converts in-context knowledge into in-parameter knowledge. We encourage the community to explore LIFT with broader training corpora, diverse models, advanced auxiliary task designs, and greater computational resources.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints, 2023. URL https://arxiv.org/abs/2305.13245.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.1, knowledge storage and extraction. *ArXiv*, abs/2309.14316, 2023. URL https://api.semanticscholar.org/CorpusID:262825178.

Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Chen, Y., Qian, S., Tang, H., Lai, X., Liu, Z., Han, S., and Jia, J. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

El-Kassas, W. S., Salama, C. R., Rafea, A. A., and Mohamed, H. K. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165: 113679, 2021.

Gandelsman, Y., Sun, Y., Chen, X., and Efros, A. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Hong, J., Lyu, L., Zhou, J., and Spranger, M. Mecta: Memory-economic continual test-time model adaptation. In *2023 International Conference on Learning Representations*, 2023.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*, 2023.

Jiang, Z., Ma, X., and Chen, W. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024.

Jin, H., Han, X., Yang, J., Jiang, Z., Liu, Z., Chang, C.-Y., Chen, H., and Hu, X. Llm maybe longlm: Self-extend llm context window without tuning. *arXiv preprint arXiv:2401.01325*, 2024.

Kamradt, G. Llmtest_needleinahaystack, 2023. URL https://github.com/gkamradt/LLMTest_NeedleInAHaystack/blob/main/README.md.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Kočiskỳ, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Li, H., Freitas, M. M. d., Lee, H., and Vasarhelyi, M. Enhancing continuous auditing with large language models: Ai-assisted real-time accounting information cross-verification. *Available at SSRN 4692960*, 2024.

Li, J., Wang, M., Zheng, Z., and Zhang, M. Loogle: Can long-context language models understand long contexts? *arXiv preprint arXiv:2311.04939*, 2023.

Liu, J. LlamaIndex, 11 2022. URL https://github.com/jerryjliu/llama_index.

Liu, Y., Kothari, P., Van Delft, B., Bellot-Gurlet, B., Mordan, T., and Alahi, A. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021.

Liu, Y., Yang, T., Huang, S., Zhang, Z., Huang, H., Wei, F., Deng, W., Sun, F., and Zhang, Q. Calibrating llm-based evaluator, 2023.

Meng, F., Wang, Z., and Zhang, M. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

Osowiechi, D., Hakim, G. A. V., Noori, M., Cheraghalikhani, M., Ben Ayed, I., and Desrosiers, C. Tttflow: Unsupervised test-time training with normalizing flow. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 2126–2134, 2023.

Peng, B., Quesnelle, J., Fan, H., and Shippole, E. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023.

Robinson, F. P. Effective study, rev. 1946.

Shen, Z., Zhang, M., Zhao, H., Yi, S., and Li, H. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Suri, G., Slater, L. R., Ziaee, A., and Nguyen, M. Do large language models show decision heuristics similar to humans? a case study using gpt-3.5, 2023.

Tapaswi, M., Zhu, Y., Stiefelhagen, R., Torralba, A., Urtasun, R., and Fidler, S. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4631–4640, 2016.

Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., and Wei, F. Augmenting language models with long-term memory, 2023. URL https://arxiv.org/abs/2306.07174.

Wang, W., Dong, L., Cheng, H., Liu, X., Yan, X., Gao, J., and Wei, F. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36, 2024a.

Wang, Y., Gao, Y., Chen, X., Jiang, H., Li, S., Yang, J., Yin, Q., Li, Z., Li, X., Yin, B., Shang, J., and McAuley, J. Memoryllm: Towards self-updatable large language models, 2024b. URL https://arxiv.org/abs/2402.04624.

Wang, Y., Ma, D., and Cai, D. With greater text comes greater necessity: Inference-time training helps long text generation. *arXiv preprint arXiv:2401.11504*, 2024c.

Wang, Y., Krotov, D., Hu, Y., Gao, Y., Zhou, W., McAuley, J., Gutfreund, D., Feris, R., and He, Z. M+: Extending memoryllm with scalable long-term memory, 2025. URL https://arxiv.org/abs/2502.00592.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Xu, P., Ping, W., Wu, X., McAfee, L., Zhu, C., Liu, Z., Subramanian, S., Bakhturina, E., Shoeybi, M., and Catanzaro, B. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*, 2023.

Yang, Q., Xu, J., Liu, W., Chu, Y., Jiang, Z., Zhou, X., Leng, Y., Lv, Y., Zhao, Z., Zhou, C., et al. Air-bench: Benchmarking large audio-language models via generative comprehension. *arXiv preprint arXiv:2402.07729*, 2024.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. Bertscore: Evaluating text generation with bert, 2020. URL https://arxiv.org/abs/1904.09675.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Table 3: Comparison of conventional long context understanding approaches with LIFT.

	RAG	ICL	LIFT
Knowledge storage	External data sources	Within context window	In parameters
Input length	Infinite	Limited	Infinite
Retrieval free	✗	✓	✓
Long-context adaptation free	✓	✗	✓

A. Related work

Long-context adaptation and efficient architectures. Existing LLMs mostly rely on pure ICL for long-context understanding. However, it is challenging for short-context models to process inputs longer than their context window sizes due to unseen positional encodings during pretraining, resulting in extremely poor performance on downstream tasks. Therefore, a common practice is to further post-train LLMs on a huge corpus of long texts (which we call long-context adaptation). Despite the effectiveness, long-context adaptation often requires tremendous computational cost. To cope with the problems, many works have been developed to accelerate the process of long-context training with efficient Transformer. Sparse attention (Kitaev et al., 2020; Wang et al., 2020; Beltagy et al., 2020) reduces memory and computation costs by using local windows or strided attention, allowing to focus on the most relevant inputs for given tasks. Linear attention (Shen et al., 2021) reduces the quadratic computation to linear by approximating self-attention with kernel functions or low-rank representations. Other alternatives for Transformer like state-space models (SSMs) (Gu & Dao, 2023) are recently proposed for efficient training based on dual representations. In this work, we focus on the conventional self-attention architecture (Vaswani, 2017) which is most widely used in current LLMs to validate the effectiveness of LIFT.

Retrieval-Augmented Generation (RAG). RAG (Lewis et al., 2020) improves the performance of long-context understanding by integrating LLMs with external data sources for retrieval (Xu et al., 2023; Jiang et al., 2024; Wang et al., 2024a; Jin et al., 2024), thereby avoiding the need to feed the entire long input. Its performance heavily relies on the quality of retrieved content, which must be relevant and concise enough to fit within models' short context windows. RAG can experience significant performance degradation or hallucination issues when the retrieved context is inaccurate or mismatched. A comparison of our LIFT with RAG and long-context adaptation is in Table 3.

Memory-augmented LLMs. A line of work (Wang et al., 2023; 2024b; 2025) explore augmenting LLMs with a memory module. Compared to RAG, which builds an offline database and retrieves from it during inference, memory-augmented LLMs emphasize continual updates of the memory module, enabling them to process long inputs sequentially. Wang et al. (2023) design a memory module that memorizes the hidden states as the LLM processes a long input and exponentially forgets past knowledge. While most memory-augmented LLMs memorize hidden states with an external module, our work explores directly storing incoming knowledge within model parameters.

Test-time training. Test-time training (TTT) (Liu et al., 2021; Gandelsman et al., 2022; Osowiechi et al., 2023; Hong et al., 2023) has emerged as a promising approach to adapt models to unseen data distributions during deployment, leveraging test data to fine-tune the model at inference time. Recent works have applied similar ideas to improve model adaptability when dealing with lengthy, context-rich inputs (Sun et al., 2024; Behrouz et al., 2024), yet focus on proposing new architectures to replace Transformer and require pretraining from scratch. Our work, in contrast, focuses on improving arbitrary pretrained models' long-context capabilities by fine-tuning them on the long input, which is not restricted to specific models or layers. Wang et al. (2024c) explore how TTT can enhance LLMs in long generation tasks such as novel writing and translation through iteratively fine-tuning a LoRA adapter to memorize the previously generated tokens. While sharing a similar idea to store context knowledge in LLM parameters, LIFT focuses on long-context understanding which poses different challenges, such as accurate memorization of details and capturing intricate long/short dependencies. Another important difference is that LIFT adopts a novel Gated Memory Adapter instead of LoRA. Traditional adapters like LoRA modifies model weights by directly adding delta weights, which unanimously changes the outputs for all inputs. In contrast, our Gated Memory Adapter uses a gate to control how much to use the original model and the adapter, dynamically determining whether to use pre-trained knowledge or new knowledge.

B. Implementation details of LIFT

B.1. Training data

The training data of LIFT includes contextualized input segments (Sections 2.1 and auxiliary tasks (Section 2.2). For contextualized input segments, the length of each input segment is $\ell = 2048$, and the offset between two segments is $s = \frac{3}{8}\ell = 768$ (Figure 1). We affiliate each segment with an extra context to simulate ICL during testing. We randomly select $r \in [0, 4096]$ (r is sampled independently for each segment) and concatenate the first r tokens and the last $(4096 - r)$ tokens of the long input as the context. The prompt is as follows:

> {*First* r *tokens*}...{*Last* $(4096 - r)$ *tokens*}Given above context, please recite following segment of the context: {*The segment*}

Only {*The segment*} is supervised during fine-tuning.

For each subtask of LooGLE and LongBench, except PassageRetrievalEN, to generate auxiliary tasks, we randomly select 16 consecutive sentences from the long input as the context and prompt LLM to synthesize a question and the corresponding answer based on the context. The prompts are as follows:

> **Instruction for LooGLE:**
> You are given a piece of text as the context. You should generate ONLY one question and the corresponding answer according to the context. You should also select one or more sentences directly from the original context as the evidence. The evidences must be EXACTLY SAME ADJACENT sentences retrieved from the context; KEEP the special tokens in the sentences. Please answer in the following format:
> Question: [question]
> Answer: [answer]
> Evidence: [evidence]
> Please DON'T output quotes when outputting evidences.
> The following is the piece of text: {*Context*}

> **Instruction for Musique:**
> You are given a piece of text as the context. You should generate ONLY one question and the corresponding answer according to the context. You should also select one or more sentences directly from the original context as the evidence. The evidences must be EXACTLY SAME ADJACENT sentences retrieved from the context; KEEP the special tokens in the sentences. Please answer in the following format:
> Question: [question]
> Answer: [answer]
> Evidence: [evidence]
> Please DON'T output quotes when outputting evidences. The question should focus on the details like names, dates, e.t.c., and the answer should be as brief as possible. The following is the piece of text: {*Context*}

> **Instruction for Narrativeqa:**
> You are given a piece of text as the context. You should generate ONLY one question and the corresponding answer according to the context. You should also select one or more sentences directly from the original context as the evidence. The evidences must be EXACTLY SAME ADJACENT sentences retrieved from the context; KEEP the special tokens in the sentences. Please answer in the following format:
> Question: [question]
> Answer: [answer]
> Evidence: [evidence]
> Please DON'T output quotes when outputting evidences. The question should focus on the details like names, dates, e.t.c. The following is the piece of text: {*Context*}

For PassageRetrievalEN in LongBench, we imitate the generation process of the test set by randomly selecting a passage and summarize the passage by LLM. The auxiliary task is to answer the index of the passage given the generated summary.

Instruction for passage summarization:
Please summarize the following text in 4 to 6 sentences: {*Context*}

B.2. Training process and hyperparameters

We design a two-stage training paradigm for both Gated Memory and PiSSA. In the first stage, the model is trained solely on contextualized input segments and optimizes the loss function $\mathcal{L}_S$ (Equation 1). In the second stage, auxiliary tasks (Section 2.2) are incorporated with contextualized input segments as the training data, and the model optimizes the loss function $\mathcal{L}$ (Equation 3).

We adopted different sets of hyperparameters during testing on LooGLE and LongBench. When testing on LooGLE, empirically, the Gated Memory architecture causes small updating steps and requires a higher learning rate and more training steps than PiSSA. The important hyperparameters for both methods are detailed in Table 4. When testing on LongBench with Gated Memory, we carefully select hyperparameters for each subtask, detailed in Table 5.

Besides, we put all the samples including the context segments and the auxiliary tasks into a single batch through gradient accumulation to stabilize gradients. The batch size per device is 1 to reduce memory costs. The other hyperparameters are kept the same for all the experiments: the context window lengths are limited to 8000 to guarantee fair comparison, which is the context window lengths of Llama 3 and Gemma 2, but shorter than that of GPT-3.5.

During generation, we adopt greedy decoding for Llama 3 and Gemma 2 to avoid randomness, while adopt sampling for GPT-3.5. For GPT-3.5, the temperature is set to 0, top p is set to 1.0, and we adopt no frequency nor presence penalty.

Table 4: The hyperparameters employed during testing on LooGLE.

Hyperparameter	Gated Memory	PiSSA
learning rate	1.0×10^{-3}	3.0×10^{-5}
weight decay	1.0×10^{-4}	1.0×10^{-4}
max grad norm	1.0	1.0
β_1	0.9	0.9
β_2	0.98	0.98
ϵ	1.0×10^{-8}	1.0×10^{-8}
stage 1 #epochs	3	1
stage 2 #epochs	5	3

Table 5: The hyperparameters employed during testing on LongBench with Gated Memory. #QA denotes the number of auxiliary tasks used. * We adopt 4 warmup steps to adjust the corresponding learning rates.

Hyperparameter	Musique	Narrativeqa	Qmsum	GovReport	PassageRetrievalEN
learning rate	3.0×10^{-3} *	3.0×10^{-3} *	3.0×10^{-3}	3.0×10^{-3}	3.0×10^{-3} *
weight decay	1.0×10^{-4}	1.0×10^{-4}	1.0×10^{-4}	1.0×10^{-4}	1.0×10^{-4}
max grad norm	1.0	1.0	1.0	1.0	1.0
β_1	0.9	0.9	0.9	0.9	0.9
β_2	0.98	0.98	0.98	0.98	0.98
ϵ	1.0×10^{-8}	1.0×10^{-8}	1.0×10^{-8}	1.0×10^{-8}	1.0×10^{-8}
#QA	10	30	0	0	60
stage 1 #epochs	3	3	8	8	3
stage 2 #epochs	5	5	0	0	5

Table 6: Performance of GPT-3.5 on LooGLE. FT stands for ”fine-tuned”.

Mothods	ShortQA	LongQA	Comprehension & Reasoning	Multiple info retrieval	Computation	Timeline reorder
ICL(GPT-3.5)	66.82	44.82	52.67	**40.77**	**27.55**	45.19
FT(GPT-3.5)	**69.66**	**45.76**	**53.44**	40.50	26.53	**49.52**

Table 7: Performance of GPT-3.5 on LongBench. FT stands for ”fine-tuned”.

Methods	Musique	NarrativeQA	Qmsum	GovReport	PassageRetrievalEN
ICL(GPT-3.5)	26.33	25.67	22.09	**25.30**	**79.17**
FT(GPT-3.5)	**27.20**	**26.53**	**22.23**	25.01	**79.17**

B.3. Gated Memory Architecture

Gated Memory is implemented as an adapter attached to the attention modules. Consider a multi-head attention module with H_q query heads (Ainslie et al., 2023). We attach a memory projection and a gate projection, which are token-wise MLPs as illustrated in Figure 4, to each of the query heads. These projections are independent across heads and do not share parameters. Denote the dimension of a query vector in a query head as d_h. The memory projection has a hidden dimension of $2d_h$ and an output dimension of d_h, while the gate projection has a hidden dimension of $\lfloor\sqrt{d_h}\rfloor$ and an output dimension of 1.

The memory and gate projections are implemented as simple MLPs to ensure low latency. We consider exploring advanced architectures for gate and memory projections as future work.

B.4. Hardwares

All the experiments, including the main experiments on LooGLE (Section 3.2) and LongBench (Section 3.2), the efficiency test (Section 3.3), and the Needle-in-A-Haystack task (Section K), are conducted on a single NVIDIA A800 Tensor Core GPU. We intentionally select this resource-constrained hardware setup, where full-parameter fine-tuning is impractical. This necessitates the use of parameter-efficient fine-tuning (PEFT) methods, which optimize both time and memory efficiency.

The resource costs (GPU hours) of the experiments, which are mainly dependent on the PEFT methods (the Gated Memory architecture or PiSSA), the sizes of the models, and the sizes of the datasets, are presented in Table 8.

Table 8: Resource costs (GPU hours) of the experiments.

Models	Methods	LooGLE ShortQA	LooGLE LongQA	LongBench
Llama3	ICL	3	3	2
	PiSSA	20	42	24
	Gated Memory	15	33	21
Gemma2	ICL	2	3	\
	PiSSA	44	64	\

C. Details of the derivation of Gated Memory

To recover the attention output when the long input is accessible during inference, we propose Gated Memory. For a **hypothetical** complete input $(\mathbf{x}', \mathbf{x})$ where $\mathbf{x}'$ is the long input that we aim to absorb into model parameters and $\mathbf{x}$ represents the in-context tokens (such as downstream questions/prompts about the long input), we let their hidden states after the $(t-1)$-th layer be $(\hat{\mathbf{h}}'^{(t-1)}, \hat{\mathbf{h}}^{(t-1)})$, where the length of $\mathbf{x}'$ is l' and the length of $\mathbf{x}$ is l. An ideal model with a long-enough context window will just take $(\mathbf{x}', \mathbf{x})$ as input and output answers. However, a practical model may only be able to take $\mathbf{x}$ as

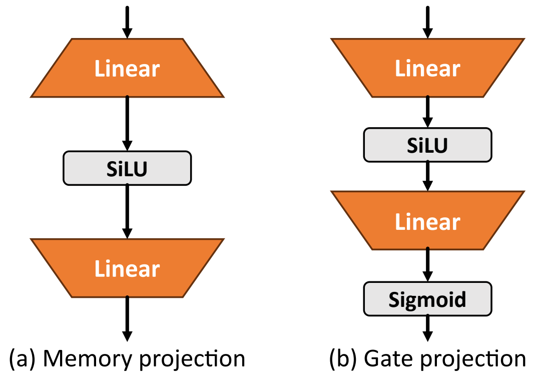


Figure 4: Architectures of the memory projection (a) and gate projection (b). Both projections are applied token-wise to the query vectors of a query head. The memory projection outputs a vector of the same dimensionality as its inputs, while the gate projection produces a scalar.

input—the goal of LIFT is to store $\mathbf{x}'$ into adapter parameters and make the adapted model have similar behaviors to the original model with complete (hypothetical) input. Let the hidden states of the real input $\mathbf{x}$ after the $(t-1)$-th layer be $\mathbf{h}^{(t-1)}$. We expect the following equation to hold in each layer:

$$\phi_{\mathbf{x}'}^{(t)}(\mathbf{h}^{(t-1)}) = f^{(t)}(\hat{\mathbf{h}}'^{(t-1)}, \hat{\mathbf{h}}^{(t-1)}),$$

where f is the original layer that takes the complete input $(\mathbf{x}', \mathbf{x})$ as the context. The adapted layer $\phi_{\mathbf{x}'}$, on the other hand, takes only $\mathbf{x}$ as the context while absorbing $\mathbf{x}'$ into its parameters.

Let's first examine the hypothetical complete attention over $(\hat{\mathbf{h}}'^{(t-1)}, \hat{\mathbf{h}}^{(t-1)})$, where $\hat{\mathbf{h}}'^{(t-1)}$ is positioned from 1 to l' and $\hat{\mathbf{h}}^{(t-1)}$ is positioned from $l'+1$ to $l'+l$. The attention output at position L ($l'+1 \le L \le l'+l$) is:

$$\text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{1:L}, \hat{\mathbf{v}}_{1:L}) = \frac{\sum_{i=1}^{L} \exp(\langle \hat{q}_L, \hat{k}_i \rangle) \hat{v}_i}{\sum_{j=1}^{L} \exp(\langle \hat{q}_L, \hat{k}_j \rangle)}, \tag{5}$$

We aim at splitting the attention output into two components: one corresponding to the out-of-context $\hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'}$, and the other corresponding to the in-context $\hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L}$. Define the gate function $g(\hat{q}_L, \hat{\mathbf{k}}_{1:L})$ and the memory function $m(\hat{q}_L, \hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'})$:

$$g(\hat{q}_L, \hat{\mathbf{k}}_{1:L}) = \frac{\sum_{i=1}^{l'} \exp(\langle \hat{q}_L, \hat{k}_i \rangle)}{\sum_{i=1}^{L} \exp(\langle \hat{q}_L, \hat{k}_i \rangle)}, \quad m(\hat{q}_L, \hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'}) = \frac{\sum_{i=1}^{l'} \exp(\langle \hat{q}_L, \hat{k}_i \rangle) \hat{v}_i}{\sum_{i=1}^{l'} \exp(\langle \hat{q}_L, \hat{k}_i \rangle)}. \tag{6}$$

$g(\hat{q}_L, \hat{\mathbf{k}}_{1:L})$ determines the proportion of attention allocated to the out-of-context part at position L, and $m(\hat{q}_L, \hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'})$ is the out-of-context representation which can be understood as performing cross attention between the current in-context token $\hat{q}_L$ and all the out-of-context tokens $\hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'}$. Then the attention output in Equation (5) can be reformulated as:

$$\text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{1:L}, \hat{\mathbf{v}}_{1:L}) = g(\hat{q}_L, \hat{\mathbf{k}}_{1:L}) \cdot m(\hat{q}_L, \hat{\mathbf{k}}_{1:l'}, \hat{\mathbf{v}}_{1:l'}) + \big(1 - g(\hat{q}_L, \hat{\mathbf{k}}_{1:L})\big) \cdot \text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L}),$$

where $\text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L})$ is the attention output with the same attention parameters operated on the in-context part's hidden states $\hat{\mathbf{h}}^{(t-1)}$ (instead of the complete hidden state $(\hat{\mathbf{h}}'^{(t-1)}, \hat{\mathbf{h}}^{(t-1)})$). Let g and m be implemented as neural networks. When the out-of-context input $\mathbf{x}'$ is considered a constant and has been absorbed into the parameters of g and m, $\hat{\mathbf{k}}_{1:l'}$ and $\hat{\mathbf{v}}_{1:l'}$ *can be removed from g and m*. We further adopt an approximation to let g only depend on $\hat{q}_L$. Consequently, both $g(\hat{q}_L, \hat{\mathbf{k}}_{1:L})$ and $m(\hat{q}_L, \hat{\mathbf{k}}_{1:L}, \hat{\mathbf{v}}_{1:L})$ become functions of $\hat{q}_L$ only. The attention output simplifies to:

$$g(\hat{q}_L) \cdot m(\hat{q}_L) + \big(1 - g(\hat{q}_L)\big) \cdot \text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L}).$$

Interestingly, the above formula decomposes the hypothetical attention output for position L's query into two parts: 1) $g(\hat{q}_L) \cdot m(\hat{q}_L)$ which retrieves in-parameter knowledge (i.e., those memorized out-of-context tokens) from memory m according to $\hat{q}_L$, and 2) $\big(1 - g(\hat{q}_L)\big) \cdot \text{attn}(\hat{q}_L, \hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L})$ which computes the standard attention of $\hat{q}_L$ only with the in-context tokens $\hat{\mathbf{k}}_{l'+1:L}, \hat{\mathbf{v}}_{l'+1:L}$, where a gate function $g(\hat{q}_L)$ controls the proportion of contribution from the two parts.

Table 9: Ablation study on contextualized training on LooGLE.

Datasets	w/o CT	w/ CT
ShortQA	43.98	**47.51**
LongQA	27.07	**29.97**

Table 10: Ablation study on contextualized training on LooGLE.

Datasets	w/o QA	10 QA	30 QA
ShortQA	47.21	47.51	**48.84**
LongQA	29.25	29.97	**30.70**

D. Contextualized Training and Task Alignment

As discussed in Sections 2.1 and 2.2, we adapt an LLM to handle a long input through two objectives: language modeling on segments of the long input and auxiliary QA tasks. While these tasks align with our objectives of memorizing the long input and enhancing reasoning based on the long input, the model may still struggle with the semantic divergence (memorization vs. reasoning) and structural divergence (language modeling vs. supervised fine-tuning) between different tasks. To address these challenges, we propose a contextualized training (CT) method for long input segments, shifting from the language modeling paradigm to a supervised fine-tuning paradigm and more closely aligning the task of input segment memorization and the auxiliary QA tasks.

Our contextualized training method involves 1) providing the model with a piece of context when asking it to memorize the segments, typically selected from the beginning and ending portions of the long input, and 2) prompting the model to generate the target segments based on the provided context. Formally, we modify the objective function (1) for the long input memorization part to the following:

$$\mathcal{L}_S(\mathbf{x};\theta) = -\sum_{k=1}^{K} \log \mathbb{P}(\mathbf{x}_{l_k:r_k} \,|\, concat(\mathbf{c}_k, \mathbf{p}); \theta), \tag{7}$$

where $\mathbf{c}_k$ represents the given context, and $\mathbf{p}$ is a prompt instructing the model to recite the segment based on $\mathbf{c}_k$. For the QA tasks, we also modify the objective (2) by concatenating the questions with a context $\mathbf{c}_q$:

$$\mathcal{L}_{AT}((\mathbf{q}_i, \mathbf{a}_i)_{i=1}^m; \theta) = -\sum_{i=1}^{m} \log \mathbb{P}(\mathbf{a}_i \,|\, concat(\mathbf{c}_q, \mathbf{q}_i); \theta) \tag{8}$$

where $\mathbf{c}_q$ keeps the same during training on different segments, which is only related to the test question. In this way, both the input memorization and QA tasks share a similar SFT format. In addition, they both align better with the real testing scenario, where given a LIFTed LLM, we can still fill the context window with the long input as much as possible to maximally leverage the in-context knowledge, instead of only filling in the testing question. Such a technique greatly improves practical performance of LIFT.

To mitigate the risk of overfitting, instead of using the same $\mathbf{c}_k$ for all the segments $\mathbf{x}_{l_k:r_k}$, we further regularize $\mathbf{c}_k$ by randomly sampling $\mathbf{c}_k$ for each segment $\mathbf{x}_{l_k:r_k}$ from both the beginning and ending of the long input with a total length of L. Specifically, we select consecutive sentences from the beginning and ending respectively compositing $\mathbf{c}_k$ with a fixed length l to align with the usages of contexts in real testing scenarios.

By employing CT, we align the input memorization task with the auxiliary QA tasks better within a closer semantic space, and unify the training and testing formats, thereby greatly enhancing the generalization capabilities of LIFT, as evidenced by our ablation study in Table 9.

Table 11: Ablation study on Gated Memory on LooGLE.

Datasets	LoRA	PiSSA	Gated Memory
ShortQA	43.31	42.03	**47.51**
LongQA	29.15	29.06	**29.97**

E. Ablation study

Ablation on contextualized training. We evaluate the effectiveness of contextualized training (CT) (discussed in Appendix D) on LooGLE. By providing a piece of input context for both the language modeling and auxiliary QAs, CT aligns the two tasks within the same semantic space and task format. As demonstrated in Table 9, the inclusion of contextualized training significantly enhances model performance on both the LooGLE ShortQA and LongQA tasks compared to the version without this component. This improvement underscores the critical role of contextualized training in achieving robust and effective long-context understanding.

Ablation on number of auxiliary QAs. Another important technique to improve LIFT's effectiveness is the auxiliary QA task introduced in Section 2.2. Here, we compare three settings: no auxiliary QA, 10 auxiliary QA pairs (default), and 30 pairs for each long input article. The results, shown in Table 10, suggest that increasing the number of auxiliary QA pairs improves performance. However, more QA pairs also mean more forward passes, and the 30 QA pair setting consumes roughly twice the training time of the 10 QA pair setting. Therefore, we choose 10 pairs as the default, balancing performance and efficiency.

Ablation on Gated Memory. As discussed in Section 2.3, our Gated Memory module acts as a specialized attention adapter, parallel to the original attention mechanism. Here, we compare it with standard LoRA (Hu et al., 2021) and PiSSA (Meng et al., 2024) adapters on LooGLE. The hyperparameters (learning rate and early-stop epochs) for both models are individually tuned to achieve optimal performance. Table 11 shows that Gated Memory outperforms both LoRA and PiSSA, demonstrating its superior ability to balance in-parameter and in-context information.

F. Baseline reproduction details

F.1. MemoryLLM

Generally, we adopt the official checkpoint `memoryllm-8b-chat` and the same method to process the documents in LooGLE and LongBench as the official implementation of MemoryLLM. For LooGLE, we split the tokenized document into consecutive segments of length of 512 tokens and inject the segments sequentially into the model memory, and prompt the model to answer the question without providing the document in the context. The prompt is as follows:

> Please answer the following question: {*Question*}

For Musique and NarrativeQA of LongBench, we reuse the scores reported in Wang et al. (2024b). For Qmsum, GovReport, and PassageRetrievalEN, following the official implementation, the entire input (The input format is assigned by LongBench) is split into the context and the prompt as follows:

> **For Qmsum:**
> **Context:**
> You are given a meeting transcript and a query containing a question or instruction. Answer the query in one or more sentences.
>
> Transcript:
> {*Context*}
> **Prompt:**
> Now, answer the query based on the above meeting transcript in one or more sentences.

Query: {*Input*}
Answer:

For GovReport:
Context:
You are given a report by a government agency. Write a one-page summary of the report.

Report:
{*Context*}
Prompt:
Now, write a one-page summary of the report.

Summary:

For PassageRetrievalEN:
Context:
Here are 30 paragraphs from Wikipedia, along with an abstract. Please determine which paragraph the abstract is from.
{*Context*}
Prompt:
The following is an abstract.

{*Input*}

Please enter the number of the paragraph that the abstract is from. The answer format must be like ”Paragraph 1”, ”Paragraph 2”, etc.

The answer is:

The context is injected into the model memory the same as the process of the LooGLE documents and the model respond to the prompt without access to the context.

F.2. LlamaIndex

We adopt `bge-small-en-v1.5` as the embedding model and `Llama-3-8B-Instruct` as the generator for LlamaIndex. Since each task of LooGLE and LongBench are based on a single context, we provide the context (without prompts) to LlamaIndex as a single document, and evaluate its ability to answer questions given only the prompt.

G. Evaluation metrics

In the reported results, the evaluation metrics are consistent with those used in the original benchmarks. For LongBench, the evaluation metrics are task-specific (Zhang et al., 2020). For LooGLE, since most automatic evaluation metrics are sensitive to semantic expression, output format, and length, we utilize GPT4-0613 (Achiam et al., 2023) as recommended in the paper to judge whether the two answers are semantically the same or not, noted as GPT4-score. It has been proven to exhibit high consistency with human evaluation and can serve as a reliable annotator to a great extent (Suri et al., 2023; Liu et al., 2023; Zheng et al., 2023). The prompt is as follows:

Given one question, there is a groundtruth and a predict_answer. Please decide whether they are the same or not in semantic. Please only output ’True’ or ’False’ .
Question: {*Question*}
groundtruth = {*Ground-truth answer*}
predict_answer = {*LLM response*}

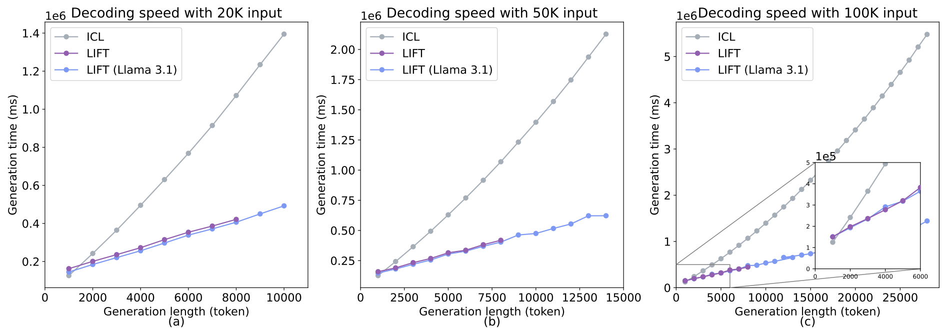


Figure 5: Subfigures (a)-(c) illustrate the decoding speed comparison between LIFT and ICL given inputs of length 20K, 50K, and 100K.

H. Empirical validation of intuition

The intuition behind LIFT is that storing in-context knowledge allows models to better understand the long input. We empirically validate our intuition by fine-tuning GPT-3.5 on the input segments (Section 2.1) and the auxiliary tasks (Section 2.2) with its API, as illustrated in Tables 6 and 7.

Similar to the standard LIFT training flow, the dataset consists of overlapping input segments and auxiliary tasks. However, we did not incorporate the Gated Memory adapter or the two-stage training flow, since the API provides no control over the PEFT adapter or the training pipeline. Overall, GPT-3.5 fine-tuned on the input outperforms the pretrained GPT-3.5 on both LongQA and ShortQA of LooGLE, as well as on most subtasks of LongBench, validating that storing the in-context knowledge within model parameters via fine-tuning improves the model's understanding of the input.

I. Additional analysis to the main results

For LooGLE, we further investigate the performance on the four LongQA subtasks including comprehension & reasoning, multiple info retrieval, computation and timeline reorder introduced in LooGLE in Table 1. As we can see, LIFT greatly enhances the two base LLMs in most subtasks. For example, LIFT improves the performance of Llama-3 on all the four subtasks with over 50% gain. These results demonstrate that LIFT enhances ICL across different models and tasks by facilitating a more holistic understanding of the entire lengthy input, which is effectively captured in the model parameters.

For LongBench, we make in-depth analysis to figure out the impact of LIFT on different subtasks. LIFT consistently outperforms the baselines on Qmsum, GovReport, and PassageRetrievalEN, all of which require the model to capture the overall gist of the article, instead of the details. LIFT stores new knowledge via fine-tuning, compressing long inputs into model parameters. As a consequence, it may generate a better overall understanding of the input, while ICL and LlamaIndex has access to only parts of the long input during inference, and MemoryLLM may forget the previous knowledge due to its updating mechanism. In contrast, Musique and NarrativeQA focus on the details of the inputs. LIFT outperforms the baselines on NarrativeQA underperforms ICL on Musique, probably because the input of Musique consists of multiple passages and LIFT fails to memorize the details of all the passages.

J. Additional efficiency analysis

By transferring input tokens into LLM parameters, LIFT alleviates the need to compute the attention score over all the input tokens when generating a token. Consequently, the decoding speed of LIFT is expected to be much faster than that of long-context ICL. We measure the total time cost (including fine-tuning) of generating y tokens with x tokens as the input. Empirically, as illustrated in Figure 5 (a)-(c), LIFT starts to outperform ICL in decoding time when generating more than 1500 tokens. This is because, LIFT only needs to fine-tune on the long input once, and later *fully becomes a short-context model* with very short decoding time per token. In contrast, ICL puts all the long input in the context, and every new token generation needs to compute the attention of the last token to all the previous tokens, incurring great latency in every new token generation.

While we deliberately design LIFT–particularly the efficient Gated Memory adapter–to support training and inference on a

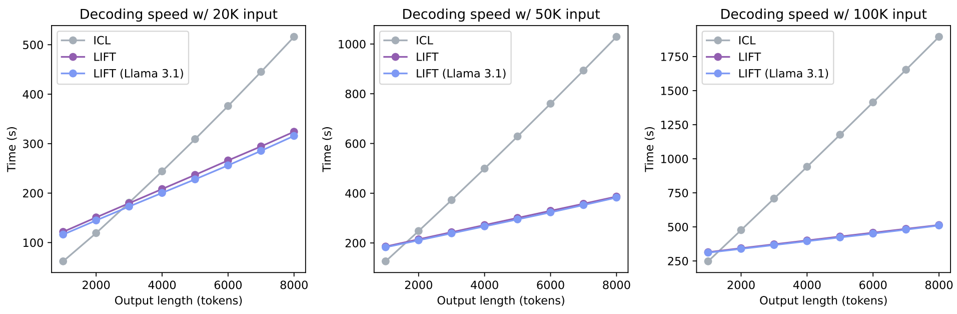


Figure 6: The generation time cost of ICL, LIFT, and LIFT (Llama 3.1), given input length of 20K, 50K, and 100K tokens.

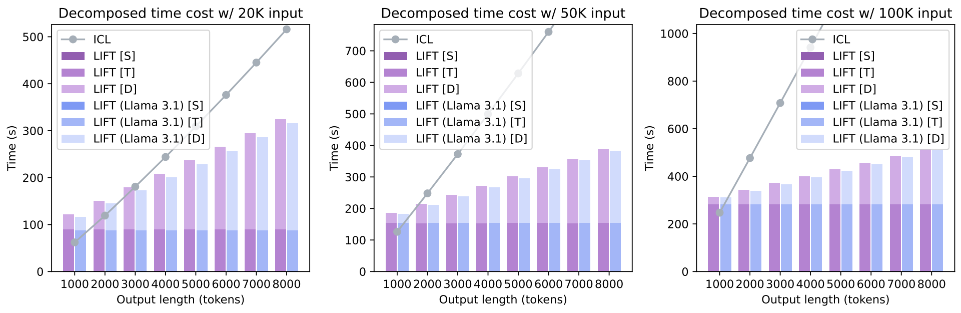


Figure 7: Breakdown of the time cost of LIFT. [S], [T], and [D] represent auxiliary task synthesizing, training, and decoding, respectively. The cost of auxiliary task synthesizing is negligible (less than 3 seconds) and the figure mainly illustrate the cost of training and decoding.

single GPU, LIFT can also be accelerated using multiple GPUs. Following the same setting as the previous efficiency test, except that LIFT and LIFT (Llama 3.1) are evaluated on 4 A800 GPUs with data parallelism and ICL is evaluated on 4 A800 GPUs, we conduct additional experiments to demonstrate that LIFT is even more efficient than long-context ICL in terms of decoding speed. The results are illustrated in Figure 6.

Empirically, with data parallelism, LIFT yields an even faster decoding speed as the generation length increases. Since ICL is also accelerated by tensor parallelism with multiple GPUs, the overall trend remains the same as that illustrated in Figure 3. However, when the generation length is short (e.g., 1K tokens), the total time cost for both LIFT and ICL becomes higher than that on a single GPU, due to the synchronization overhead introduced by multi-GPU inference.

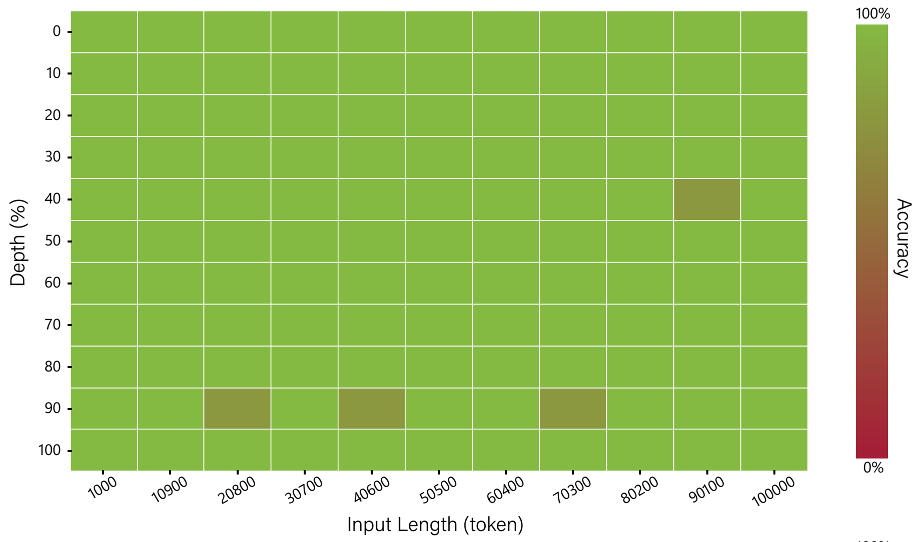
Moreover, we analyze the time cost of each component of LIFT, as illustrated in Figure 7. The workflow of LIFT can be decomposed into three stages: auxiliary task synthesis, training, and decoding. The cost of synthesizing auxiliary tasks is negligible, since it is accelerated by vLLM. As the input length increases, the training stage occupies a larger proportion of the total time cost, suggesting that improving the training efficiency is the key to further accelerating LIFT. We leave exploration of faster-converging training methods and advanced parallelism techniques for future work.

K. Results on Needle-in-a-Haystack (NIAH)

We present the experimental results in the NIAH (Kamradt, 2023) task in Figure 8, as further analysis of the pros and cons of LIFT and directions for future works. The task requires accurate retrieval from the contexts. We adopt a strong long-context model, Llama-3.1-8B-Instruct, as the baseline and apply the LIFT framework to the model.

The maximum context length of our test is 100K, which is within the 128K context window of Llama-3.1-8B-Instruct. As expected, the baseline achieves nearly perfect performance. However, LIFT slightly degrades the performance and the degradation seems irregular.

The reason for the degradation may be that LIFT introduces more noise to the model. While most parts of the context are



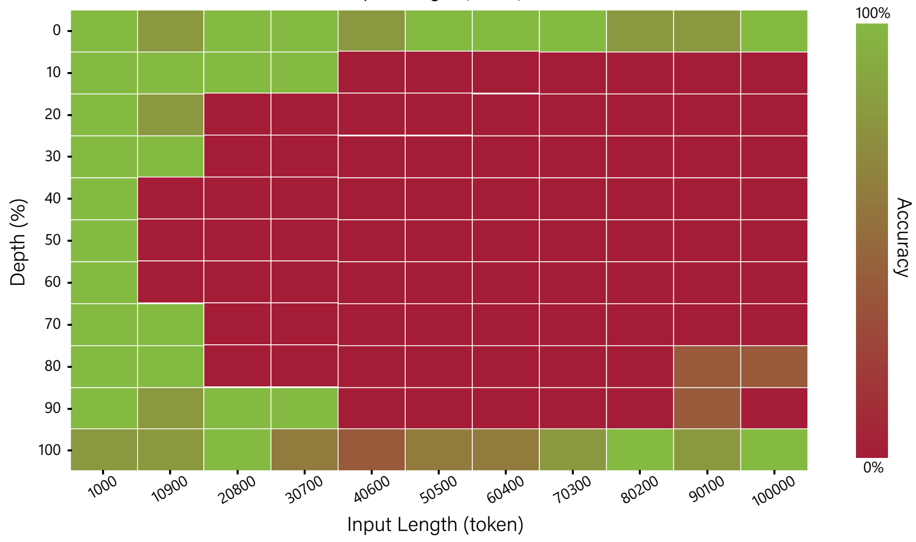


Figure 8: Performance on NIAH: ICL (top) vs. LIFT (bottom).

irrelevant to the answer, LIFT asks the model to memorize all the context. The model is likely to be misled by the large amount of irrelevant information.

As summarized in Section 4, precise memorization can be challenging for LIFT. On the one hand, LIFT can't accurately memorize the context while avoiding overfitting. On the other hand, LIFT is likely to be misled when most information is irrelevant to the answer. Future works may improve the LIFT framework from these two aspects.

L. Analysis to the end-to-end training of Gated Memory

To train Gated Memory, we learn two MLPs, $g(q)$ and $m(q)$. Suppose we have a complete input $(\mathbf{x}', \mathbf{x})$, where $\mathbf{x}'$ denotes the full long input (e.g., a long document) and $\mathbf{x}$ denotes the prompt (e.g., a question). Let the hidden states of the last layer be $(\hat{\mathbf{h}}'^{(l-1)}, \hat{\mathbf{h}}^{(l)})$, corresponding to $(\mathbf{x}', \mathbf{x})$, and let the queries, keys, and values be $\hat{\mathbf{q}}, \hat{\mathbf{k}}, \hat{\mathbf{v}}$. Equation 6 defines the ground truth for $g(q_L)$ and $m(q_L)$, where l and l' are the length of $\mathbf{x}$ and $\mathbf{x}'$, respectively. However, directly using Equation 6 as supervision (distillation) is infeasible as passing the entire long input $\mathbf{x}'$ through the model is too expensive. Instead, we treat Gated Memory as an adapter similar to LoRA and train it in a SFT manner (end-to-end training).

Empirically, the end-to-end training proves effective, outperforming ICL and PiSSA on most tasks (Tables 1, 2, and 11). However, it doesn't fully replicate the theoretical attention mechanism. We adopt Llama-3.1-8B-Instruct with a 128K

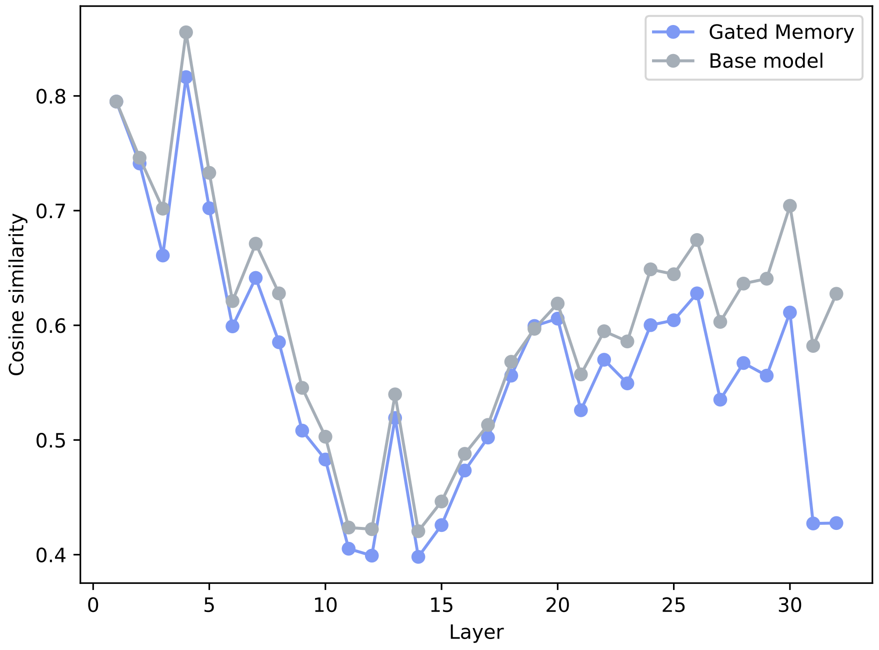


Figure 9: The cosine similarity between the **Ground-truth** attention output and the **Gated Memory** and **Base model** output respectively.

context window and compare the attention outputs under the following settings:

- **Ground-truth**. The model has access to the complete context.
- **Base model**. The model can only see the question.
- **Gated Memory**. The model is fine-tuned using the LIFT training process with the Gated Memory adapter. During evaluation, the model can only see the question.

In detail, the input text for each setting is as follows:

Ground-truth:
Given the article "{*Title*}": {*Context*} Based on the article {*Title*}, please answer the following question: {*Question*}
Answer: {*Answer*}
Base model and **Gated Memory**:
Based on the article {*Title*}, please answer the following question: {*Question*}
Answer: {*Answer*}

In the experiment, {*Title*} and {*Context*} are the title and the content of the first document in LooGLE ShortQA dataset, and {*Question*} and {*Answer*} are the questions and ground-truth answers provided by LooGLE. We collect only the attention outputs of the {*Answer*} tokens and compute the cosine similarity between the attention output of **Ground-truth** and those of **Base model** and **Gated Memory**, respectively, as illustrated in Figure 9. The similarity corresponding to **Gated Memory** is consistently lower than that corresponding to **Base model**, suggesting that the end-to-end trained Gated Memory learns to answer questions through mechanisms other than full-context ICL. We consider conducting more comprehensive experiments with full-attention distillation as an important direction for future work.

M. Examples of the auxiliary tasks

To enable LLMs to reason based on the learned long inputs, we synthesize auxiliary tasks related to the inputs with the prompts presented in Appendix B. We randomly sample several auxiliary tasks generated during test on LooGLE and

LongBench as examples:

LooGLE:
Q1: What was the original intention of the architect, Picardo, when designing the Hostería de Cáceres?
A1: To make the Hostería appear, through imitation, to be an integral historic part of the old city centre.

Q2: What was the year when Picardo's work for Paradores de Turismo de España ended?
A2: 1985

Q3: Who was commissioned to design a means of displaying the painting Guernica in the Museum's annexe, the Casón del Buen Retiro?
A3: Picardo and fellow architect José García María de Paredes.

Q4: Where is the Parador at Arcos de la Frontera located?
A4: The Parador at Arcos de la Frontera is located in the centre of the old town, at the top of the cliffs that overhang the Rio Guadalete.

Q5: What was the initial plan for the number of guest rooms in the Parador building?
A5: 23 double guest rooms and 10 singles.

Musique:
Q1: Who played the role of Syed Modi in the movie Sau Crore?
A1: Raman Kapoor

Q2: When was Grown Ups 2 released?
A2: July 12, 2013

Q3: What is the title of the Christian nu-metal band that inspired the film's title?
A3: P.O.D.

Q4: Who is Vasudevan's father?
A4: Kadikalingam

Q5: When did Big Head Todd's version of "Boom Boom" start being used as the opening theme of NCIS: New Orleans?
A5: 2014

NarrativeQA:
Q1: What is the reason for the narrator's desire to leave?
A1: The narrator wants to get away from his young lady because it would help him not to pretend to satisfy her.

Q2: What was Kent Mulville's face like when he came back downstairs?
A2: His face told as few tales as I had seen it succeed in telling on the evening I waited in the lecture-room with Miss Anvoy.

Q3: Who was Lady Coxon's husband?
A3: The late Sir Gregory.

Q4: When did Ruth ask Mrs. Mulville to ask the narrator to come and see her?
A4: It was not till a day or two ago.

Q5: Who is George Gravener's wife?
A5: His wife, whose fortune clears the property, is criminally dull.

PassageRetrievalEN:
Q: Here are 30 paragraphs from Wikipedia, along with an abstract. Please determine which paragraph the abstract is from.

{*Context*}

The following is an abstract.

In 887, a mutiny led by Xue Lang forced Zhou to flee to Chang Prefecture, where he sought the protection of Ding Congshi. Qian, the leader of the Eight Corps, responded by sending three commanders to attack Xue, but one of them, Du Leng, turned against Zhou and captured Chang Prefecture. Zhou was later escorted to Hang Prefecture, where he died soon after. Qian then captured Run Prefecture and executed Xue, and went on to capture Su Prefecture and establish himself as the dominant power in the region. Qian was rewarded by Emperor Zhaozong with various titles and honors, including the title of military governor of Zhenhai and the honorary chancellor title of Tong Zhongshu Menxia Pingzhangshi.

Please enter the number of the paragraph that the abstract is from. The answer format must be like ”Paragraph 1”, ”Paragraph 2”, etc.

The answer is:
A: Paragraph 22