LLoCO: Learning Long Contexts Offline

Sijun Tan*, Xiuyu Li*, Shishir Patil, Ziyang Wu, Tianjun Zhang,
Kurt Keutzer, Joseph E. Gonzalez, Raluca Ada Popa
UC Berkeley
{sijuntan,xiuyu}@berkeley.edu

Abstract

Processing long contexts remains a challenge for large language models (LLMs) due to the quadratic computational and memory overhead of the self-attention mechanism and the substantial KV cache sizes during generation. We propose LLoCO, a novel approach to address this problem by learning contexts offline through context compression and in-domain parameter-efficient finetuning with LoRA. Our method enables an LLM to create a concise representation of the original context and efficiently retrieve relevant information to answer questions accurately. Our approach extends the effective context window of a 4k token LLaMA2-7B model to handle up to 128k tokens. We evaluate our approach on several long-context question-answering datasets, demonstrating that LLoCO significantly outperforms in-context learning while using $30\times$ fewer tokens during inference. LLoCO achieves up to $7.62\times$ speed-up during inference and $11.52\times$ higher throughput during finetuning, substantially reduces the cost of long document question answering. This makes it a promising solution for efficient long context processing.[1]

1 Introduction

Large language models (LLMs) have demonstrated remarkable performance across a wide range of tasks (Touvron et al., 2023; Jiang et al., 2023a). Many of these tasks require LLMs to understand and reason about long contexts. For instance, document question answering is one of the most common applications of LLMs, where the model is presented with a document as context and asked to respond to related questions or summarize the text. These documents may range from lengthy articles to entire books, potentially surpassing the context window limit of LLMs. Consequently, there is a growing trend in both academia and industry to enhance LLMs' capability to process longer contexts effectively (Chen et al., 2023b; Jiang et al., 2023a; Peng et al., 2024; Chen et al., 2024). This need has driven innovation among LLM providers like OpenAI and Anthropic to develop models that can handle increasingly lengthy texts consisting of several thousands of tokens.

Despite the impressive progress made by the LLM model providers, scaling these models to adeptly manage extended contexts remains a formidable challenge, both technically and financially. Due to the self-attention mechanism, transformer-based LLMs incur a quadratic computational and memory overhead as sequence length increases. Many long-context tasks require reusing the same context many times, which incurs extra latency overhead and substantial costs, as most commercial LLMs operate on a pricing model that is directly tied to the number of tokens processed. For example, a single inference run with a document that has 100k tokens would take 1.5 USD on Claude 3 Opus[2] and 1 USD on GPT-4-turbo[3].

Orthogonal to the exciting efforts of expanding the context window limit, our study introduces an innovative strategy to tackle the long context challenge. We propose a method where context information is condensed offline through finetuning, enabling the model to provide accurate responses during inference with streamlined contextual representations. To illustrate our idea, consider an analogy: *envision an LLM as a student preparing for an exam, where we, the researchers, are the examiners providing study materials and questions*. Traditional in-context learning with full context or Retrieval-Augmented Generation (RAG) resembles an open-book exam, where the LLM has access to all materials while answering questions. In con-

*Equal contribution

[1]Our code is publicly available on https://github.com/jeffreysijuntan/lloco.

[2]https://www.anthropic.com/api

[3]https://openai.com/pricing

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 17605–17621
November 12-16, 2024 ©2024 Association for Computational Linguistics

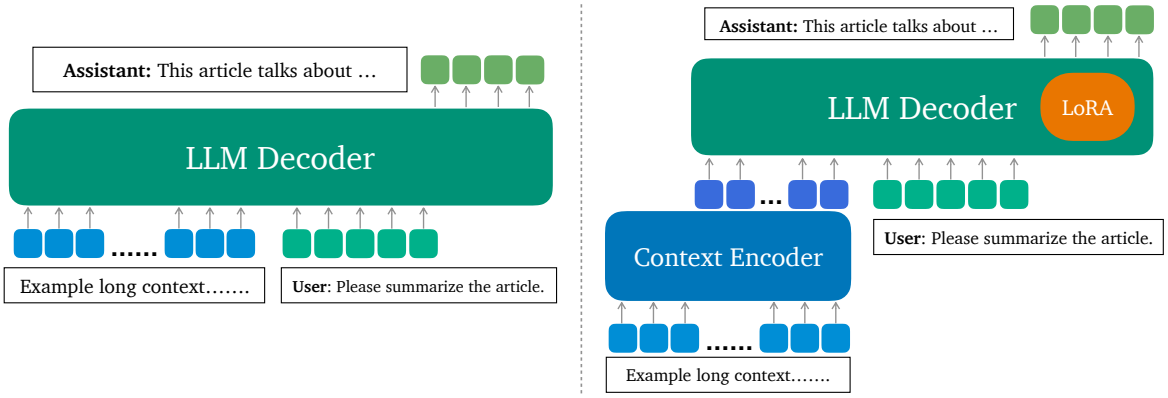

Figure 1: The architecture of regular LLM (left) vs LLoCO (right). In regular LLMs, long contexts are appended directly to the prompt. In contrast, LLoCO first processes these contexts through a context encoder. The resulting summary token embeddings are then prepended to the LLM's prompt, which are significantly shorter. LLoCO instruction finetunes on embeddings of targeted document groups using a LoRA module. This aligns the LLM's embedding space with the summary embeddings while keeping both the LLM and context encoder unchanged.

trast, our approach is akin to a semi-closed-book exam, where the LLM cannot bring the entire book but is allowed to bring a cheat sheet. To excel in the exam, the student must 1) study efficiently to distill a concise yet informative cheat sheet, and 2) effectively retrieve relevant information from the cheat sheet to accurately answer exam questions. Namely, 1) How can we train a model to produce a compact representation of the original context that the LLM can interpret and utilize effectively? 2) How to enable the LLM to proficiently navigate and extract pertinent details from this representation during inference?

Regarding the first question, we find the formulation closely resembles the existing research direction on context compression. Prior works (Chevalier et al., 2023; Mu et al., 2023; Ge et al., 2023) have proposed compressors that aim to distill the essence of the original texts into compact representations that are aligned with LLMs. Our work primarily focuses on the second question – we've observed that despite the progress in context compression, LLMs often struggle to accurately read such "cheat sheets" and tend to hallucinate when applying them to answer queries. To address this issue, we employ in-domain parameter-efficient finetuning directly on the compacted context (cheat sheet) without altering its content, which significantly improves the LLM's ability to accurately extract and utilize information from these compressed representations. In this way, we can steer the LLM to process long context more efficiently and accurately. This approach allows us to extend the effective context window of a 4k LLaMA2-7B model to handle up to 128k tokens. Moreover, we achieve state-of-the-art results that match or even surpass the performance of a LLaMA2-7B-32k model with full context on long context benchmarks, while using 30× fewer tokens.

This insight has led us to introduce LLoCO, a pipeline that learns contexts offline through the combination of context compression and parameter-efficient finetuning. It consists of three stages: preprocessing, finetuning, and serving. First, we preprocess the documents into "cheat sheets". Then, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) to perform parameter-efficient finetuning on these "cheat sheets" in groups. For serving system design, we use a standard RAG retriever to retrieve the compressed document as well as the most relevant LoRA module, and apply them to the LLM for inference. The contributions of our work could be summarized as follows:

- We introduce a novel method for effectively modeling long contexts by combining context compression with instruction finetuning. Our approach extends the context window of a 4k LLaMA2-7B model to handle up to 128k tokens, achieving performance that significantly surpasses in-context learning while using 30× fewer tokens.

- We propose LLoCO, a novel framework that combines context compression, retrieval, and parameter-efficient finetuning. This pipeline could be deployed to significantly speed up and reduce the cost of long document question answering. With the compressed context during inference and instruction finetuning, we demonstrate that LLoCO achives a

7.62× speedup over inference latency, and a 11.52× higher throughput compared to finetuning over the original context.

2 Related work

Long-context LLMs Recently, there have been efforts to increase the LLMs' context window sizes efficiently with continual pretraining or finetuning. One line of work focuses on scaling the Rotary Position Embeddings (RoPE) (Su et al., 2021), achieving longer contexts up to 128k (Chen et al., 2023b, 2024; Peng et al., 2024). Mistral (Jiang et al., 2023a) proposes sliding window attention that only attends to part of tokens from the previous layer, reducing compute and enabling pretraining with long-context to 30k. Nevertheless, as the auto-regressive generation in LLMs is largely memory-bound (Kwon et al., 2023), storing the KV cache of longer contexts slows down the inference and requires large GPU VRAMs.

Context compression A closely related topic is context compression, which aims to train a general compressor that can compress any input prompts. GIST (Mu et al., 2023), AutoCompressor (Chevalier et al., 2023), and ICAE (Ge et al., 2023) finetune LLMs in a "soft prompt-tuning" manner, either applying specific regularization in attention masks or utilizing dedicated "memory tokens" to compress contexts into embeddings with significant shorter lengths. LLMLingua family (Jiang et al., 2023c,b; Pan et al., 2024) proposes a question-aware framework to compress prompts in natural languages for black-box LLM APIs. Another line of work employs KV cache compression by eviction (Zhang et al., 2024b; Xiao et al., 2024), which only keeps informative keys and values for generation during inference, or quantization (Sheng et al., 2023b; Hooper et al., 2024). However, all those approaches aim to compress any inputs, which usually incurs rapid performance drops as the compression ratio exceeds a certain limit (e.g. 2-4×), especially for out-of-distribution texts, hindering their practical applications. In this work, we mainly focus on extreme compression of in-distribution documents of interest, going up to 30× compression rates.

Retrieval-augmented Generation Retrieval enhances the performance of LLMs in knowledge-intensive tasks such as question answering with long documents or in open-domain. RAG techniques are effective both in finetuned scenarios and when applied to off-the-shelf LLMs (Guu et al., 2020; Lewis et al., 2020; Zhang et al., 2024a), and have seen many advancements recently, emphasizing improved external knowledge integration, broader query support, and enhanced content clarity (Jiang et al., 2023d; Schick et al., 2023; Asai et al., 2023). Despite the improved performance from retrieval, challenges still remain in terms of runtime efficiency (Mallen et al., 2022) and effective filtering of irrelevant parts (Shi et al., 2023; Xu et al., 2024). Our proposed method opens up new opportunities for capturing important information while ensuring efficient generation.

Parameter-efficient Finetuning Parameter-efficient finetuning methods (Hu et al., 2022; Lester et al., 2021; Liu et al., 2024b) freeze the majority of the model weights and only update a small subset to finetune large models efficiently. LoRA (Hu et al., 2022) is one of the most widely adopted techniques, and recent advancements (Sheng et al., 2023a; Chen et al., 2023a) have focused on enhancing the system performance for deploying LoRA adaptors. In particular, (Sheng et al., 2023a) has achieved the deployment of thousands of LoRA instances on a single NVIDIA A100 GPU. LLoCO, which utilizes LoRA for finetuning, could benefit from those improvements to deploy LoRA adaptors efficiently in long context scenarios.

3 Method

3.1 Architecture Overview

Figure 1 illustrates our proposed LLoCO architecture, which consists of two components: a context encoder and an LLM decoder. In a typical instruction-finetuned LLM, the prompts can be categorized into system, user, and assistant prompts. The system prompt contains task instructions and rules the LLM should follow, as well as relevant contexts. The user prompt is a question asked by the user, and the assistant prompt is the answer generated by the LLM. The context information in the system prompt can include the user's interaction history or documents related to the user's question. These contexts can become very long, potentially surpassing the LLM's context window limit.

To overcome the context window limitation, we propose using a context encoder to compress the original long contexts into a much more compact representation. The context encoder itself is a language model that takes a sequence of tokens as input and outputs a sequence of token embed-

dings. These output token embeddings, which we call summary embeddings, should be significantly shorter than the original context. The summary embeddings are then prepended to the LLM decoder and serve as the LLM's system prompt. The user's prompt is processed normally through the LLM decoder's tokenizer, and the LLM generates answers (the assistant prompt) conditioned on the summary embeddings and user prompts.

In our design, the context encoder can be any model capable of producing a compact representation aligned with the LLM decoder. The summary embeddings can be viewed as pseudo-words in the LLM decoder's text embedding space, representing abstract concepts or summaries. For our experiments, we select AutoCompressor for LLaMA2-7B (Chevalier et al., 2023), a context compressor finetuned on LLaMA2-7B with the dual ability to generate summary tokens from long contexts and perform text completions conditioned on the summary tokens. The compressor groups the document into chunks of 1536 tokens and recursively compresses each chunk to 50 summary embeddings. To ensure alignment between the summary embeddings and the decoder LLM, we choose LLaMA2-7B as our decoder LLM but use the AutoCompressor's finetuned weights as the model weights.

While there are other available compressors for LLaMA2-7B (Ge et al., 2023; Mu et al., 2023), we find AutoCompressor to be most suited for our intended use cases given that it can 1) support compressing very long context due to its recursive training procedure, and 2) achieve a great compression ratio of 30:1. We consider the construction of context encoders (compressors) to be an important and orthogonal research direction. Developing more performant and universal context encoders in a streamlined fashion can further enhance the efficiency and effectiveness of LLoCO, denoting crucial future work.

3.2 Pipeline for Offline Context Learning

The pipeline of LLoCO consists of two primary stages: the preprocessing stage and the finetuning stage. We also outline a serving stage for real-world deployment in the system design.

Preprocessing First, we employ a preprocessing stage that leverages our context encoder to process the original documents. Since AutoCompressor is used as our context encoder, we follow its practice of dividing the documents into chunks of 1536 tokens and passing them through the context encoder. The context encoder outputs 50 summary embeddings for each chunk recursively, with each embedding having a dimension of 4096.

Our preprocessing stage is designed to be extensible, allowing seamless integration into a retrieval-augmented generation (RAG) system for document QA. In a typical RAG system, preprocessing involves building a vector database to index a collection of documents, where documents are chunked into passages, and each passage is associated with a key— a sentence embedding generated by a text embedding model.The design of LLoCO allows summary token embeddings to be stored alongside the original passages in the vector database, indexed by the same passage key.

Finetuning During the finetuning stage, we first segment the documents into groups based on their type (e.g., academic papers, news) or the tasks that users want to perform (e.g., QA, summarization). For each group of documents, we perform parameter-efficient finetuning using a LoRA adaptor. The finetuning data can be in-domain instruction pairs provided by the model provider. If such a dataset does not exist, it could also be generated using self-instruct (Wang et al., 2022) techniques or distilled from a more powerful model like GPT-4. At the end of the finetuning process, we obtain a finetuned LoRA adaptor for each group of documents. In the vector database system design, each passage entry will include an identifier for the corresponding LoRA module. Additionally, a separate database will be established to store all adaptors.

Concretely, given summary tokens $\mathbf{X}_m$ and instruction pairs $\{\mathbf{X}_q^1, \mathbf{X}_a^1, \ldots, \mathbf{X}_q^L, \mathbf{X}_a^L\}$ of length L for a document group g, we aim to maximize the probability of generating $\mathbf{X}_a$ defined as:

$$p(\mathbf{X}_a|\mathbf{X}_m, \mathbf{X}_q) = \prod_{i=1}^{L} p_{\theta_g}(\mathbf{X}_a^i|\mathbf{X}_m, \mathbf{X}_q^i) \quad (1)$$

where θ_g denotes the LoRA weight for group g.

Serving We also design a serving stage that can be naturally extended to a RAG system, where we leverage a standard RAG retriever to retrieve relevant documents, but instead prepend the compressed embeddings of these documents to the LLM decoder. Additionally, we apply the most relevant LoRA module to the model. Readers can refer to Appendix E for more details.

4 Experiments

In the experiment section, we aim to investigate the following aspects of LLoCO: (1) its effectiveness in comprehending compressed long contexts, (2) the extent to which summary embeddings can preserve essential information, and (3) the efficiency improvements on the associated system costs.

Datasets We select four datasets dedicated to question-answering tasks—**QuALITY** (Pang et al., 2021), **Qasper** (Dasigi et al., 2021), **NarrativeQA** (Kočiskỳ et al., 2018), and **HotpotQA** (Yang et al., 2018)—along with one for summarization, **QMSum** (Zhong et al., 2021). All datasets have long documents as contexts. For all the datasets, we use their validation set for evaluation. We follow the official metrics for each dataset. For QuALITY, we report the exact match (EM) score. For QMSum, we report the geometric mean of ROUGE scores. For the remaining datasets (Qasper, NarrativeQA, and HotpotQA), we report the F1 scores. More details on these datasets can be found in Appendix B.

Model Configuration In this study, we consider two base models. The first is the original LLaMA2-7B-chat (Touvron et al., 2023), which has a context window of 4096 tokens. The second is Longchat-7b-v1.5-32k (Li et al., 2023), a finetuned LLaMA2 model with an extended context window of 32,000 tokens via position interpolation. From now on, we will use LLaMA2-7B-4k to refer to the prior model and LLaMA2-7B-32k to denote the latter one. Unless otherwise specified, we set the LoRA rank to 8 for our experiments. All LLoCO's models are finetuned on AutoCompressor, which is itself finetuned on LLaMA2-7B.

4.1 Long Document QA

To evaluate the effectiveness of LLoCO on the aforementioned long context datasets, we consider the following scenarios:

1. **LLaMA-2-7B-4k/32k/128k with Original Context.** This is a baseline setting where we provide the LLMs with the ground truth document. We truncate the documents if their length exceeds the context window limit.

2. **LLaMA-2-7B-4k/32k with Retrieval.** Retrieval is a standard baseline compression method for long document question answering. For each document, we chunk it into passages of 512 tokens and use Contriever (Izacard et al., 2021) to retrieve the top 5 passages from this document and concatenate them to form the context.

3. **AutoCompressor.** In this setting, we use AutoCompressor (Chevalier et al., 2023) to compress the contexts into summary embeddings and prepend them to the LLM, without performing any in-domain finetuning, which is equivalent to using it straight out of the box. The AutoCompressor compresses a chunk of 1536 tokens into 50 summary tokens, resulting in an effective context window of roughly 128k tokens. We do not truncate the context unless it exceeds this limit.

4. **LLoCO (ours).** LLoCO is our proposed system for long document question answering. For each dataset we evaluate, we perform instruction finetuning using the question-answering pairs from the training set. During both finetuning and inference, we prepend the summary embeddings of the corresponding ground truth document to the LLM. We do not truncate the context unless it exceeds the 128k context window limit.

Our results are summarized in Table 1. When AutoCompressor is used without in-domain finetuning, its performance sometimes falls short of the baseline where no context is appended. However, by combining compression and finetuning, LLoCO significantly outperforms the baseline on all datasets, using $30\times$ fewer tokens.

For the QuALITY, Qasper, QMSum, and HotpotQA datasets, most samples fit within the 32k token limit of the LLaMA2-7B-32k model without truncation. Here, LLoCO does not have a longer context advantage but still uses $30\times$ fewer tokens than the baseline while achieving better performance. This shows that in-domain finetuning enhances the model's ability to interpret compressed embeddings and answer questions. Detailed sequence length statistics are in Table 6 in Appendix B.

For the NarrativeQA dataset, the average document length is 84,770 tokens, exceeding the LLaMA2-7B-32k context limit. LLoCO compresses these contexts to about 2,600 tokens, enabling comprehensive context utilization for question answering. To assess the impact of context length and ensure fair comparison, we conducted an ablation study with two new configurations: (1)

Setup	Ctx Size	τ	QuA	QAS	QM	NQA	HQA	Avg.
LLaMA2-7B								
4k w. Original Context	4k	1x	40.45	16.67	14.62	14.42	32.47	23.44
32k w. Original Context	32k	1x	38.88	21.72	14.58	16.76	31.58	24.70
128k w. Original Context	128k	1x	33.89	20.38	13.88	28.22	27.69	24.81
Compression Methods								
LLaMA2-7B-4k w. Retrieval	4k	1.6x	38.73	18.29	14.33	22.28	27.95	24.31
LLaMA2-7B-32k w. Retrieval	32k	12.8x	36.48	24.92	15.40	19.32	22.32	23.68
AutoCompressor	128k	30x	33.51	15.03	12.53	11.66	21.01	18.75
LLoCO w. Full Context (ours)	128k	30x	**41.51**	**29.01**	**16.68**	**28.34**	**37.83**	**30.67**

Table 1: Experiment results on long document QA datasets. τ indicates the compression ratio. For LLaMA2-7B-4k/32k with retrieval, the compression ratio is obtained by dividing the model's context window limit (4k/32k) by the length of the retrieved passages, which is fixed at 2560 tokens.

documents truncated to 32k tokens before compression and finetuning LLoCO on the resulting embeddings, and (2) testing another LLaMA2-7B extended for 128k context window with state-of-the-art long-context ability from (Fu et al., 2024). Our results in Table 2 show that LLoCO performs better than LLaMA-7B at both 32k and 128k context lengths.

Setup	NQA
LLaMA2-7B-32k (Li et al., 2023)	16.76
LLoCO-32k	27.99
LLaMA2-7B-128k (Fu et al., 2024)	28.22
LLoCO-128k	28.34

Table 2: Impact of context lengths on NarrativeQA.

4.2 Ablation Study

In this section, we present ablation studies comparing LLoCO under various setups, including finetuning LLaMA with the original context and evaluating different compression ratios. Additional studies, such as exploring alternative context encoders, are provided in Appendix D.

Finetuning LLaMA with Original Context One interesting question is how well LoRA finetuning works over original uncompressed context, and how does that compare to LLoCO's approach. To investigate this, we conduct additional finetuning experiments on LLaMA2-7B-4k/32k, where we append the original uncompressed context as a system prompt during finetuning. These models are finetuned following the same setup as LLoCO. We do not explore the impact of finetuning LLaMA2-7B-128k with uncompressed context due to constraints on compute resources.

As shown in Table 3, both LLaMA-7B-4k and LLaMA-7B-32k exhibit notable performance improvements after finetuning, with increases of 2.88% and 6.62% respectively. Despite these improvements, the performance of LLoCO remains comparable to that of the finetuned LLaMA-7B-32k model. This finding suggests that finetuning in a compressed context is just as effective as finetuning in the original context. Compared to full-context finetuning, LLoCO's finetuning step is considerably faster and more cost-efficient due to the use of a significantly shorter context (see 4.5). This makes LLoCO a more practical finetuning solution for long document question-answering tasks.

Exploring the Impact of Compression Ratios In this ablation study, we evaluate LLoCO under various compression ratios to analyze their effect on performance. The original LLoCO setup, adapted from AutoCompressor, applies a 30x compression ratio, reducing 1536 tokens to 50 tokens. To explore the impact of different ratios, we also compress 1024 tokens and 2048 tokens into 50 tokens, corresponding to compression ratios of 20x and 40x, respectively. We assess these configurations on the QuALITY, QMSum, and NarrativeQA datasets, with the results presented in Figure 2.

Our findings suggest that performance remains relatively stable across different compression ratios, with both 20x and 30x ratios consistently outperforming 40x across all datasets. Interestingly, the 30x ratio performs on par with, and occasionally slightly better than, the 20x ratio. We attribute this to the context encoder (AutoCompressor) being originally optimized for the 30x setting, allowing it

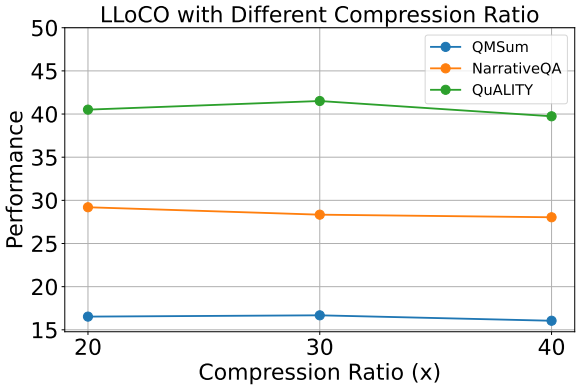

Figure 2: Impact of compression ratio on LLoCO's performance.

to maintain high performance even at higher compression rates.

4.3 Evaluation on LongBench

We further evaluate LLoCO on LongBench (Bai et al., 2023), which consists of 6 subtasks. Given that the primary applications of LLoCO are document question answering and summarization, our evaluation focuses on the SingleDoc QA, MultiDoc QA, and Summarization tasks. There exists overlap with some datasets (e.g. Qasper, QMSum) we evaluated in Section 4.1. However, the validation sets differ as LongBench samples a specific subset of examples for each dataset. We have rigorously ensured that LongBench does not feature any questions that overlap with those used in the LLoCO training, thereby completely removing any risk of data leakage. To achieve the best performance of LLoCO, we choose LoRA adaptors tailored for each particular category/dataset in our evaluation.

Specifically, for the SingleDoc tasks, we use the NQA LLoCO and Qasper LLoCo for the NQA and QAS tasks, respectively. For MultiField-En (MFE) (Bai et al., 2023), we use GPT-4-turbo to generate question-answering pairs from the training documents and finetune LLoCO using these pairs. For MultiDoc tasks, we use the combined finetuned LLoCO from Section D, which works well for all tasks. For the Summarization tasks, we create a training dataset by combining the entire QMSum (Zhong et al., 2021) training data with 5,000 randomly sampled examples from the MultiNews (Fabbri et al., 2019) training data. We then finetune LLoCO on this combined dataset.

Our evaluation results show that LLoCO outperforms the baselines on 5 out of the 9 datasets. In particular, LLoCO excels in the MultiDoc QA task, achieving much better results on all three datasets. LLoCO also demonstrates comparable performance in two datasets (MultiNews, Qasper), but falls short in the GovReport (Huang et al., 2021) and MultiField-En datasets. The GovReport dataset is a summarization task that requires the model to generate a one-page summary. We found that LLoCO does not perform well when the generated content is lengthy, which currently is a limitation of our approach. The lower performance on the MultiField-En dataset could be attributed to the data being out-of-distribution compared to our training data, as this dataset does not provide any training examples. Despite these limitations, the average score of LLoCO across all datasets is higher than that of the LLaMA2 baseline, highlighting the overall effectiveness of our approach.

4.4 Needle In A Haystack

We further investigate LLoCO's proficiency to retrieve information across different positions of context window with varying lengths using the renowned Needle In A Haystack task (gkamradt, 2023). Tailoring this to our pipeline, we randomly select a long article exceeding 32k tokens from the NarrativeQA dataset as the "haystack". This article is used to test the LLoCO model, which has been finetuned on this dataset as discussed in Section 4.1.

Single Fixed Needle Our initial evaluation focuses on a straightforward scenario: the insertion of a consistent fixed "needle". Figure 3 shows that our LLoCO successfully retrieves the needle in $\sim$80% across all tested context sizes, indicating robust performance with no degradation at longer contexts, while LLaMA2-7B-32k exhibits substantially lower effectiveness in this task.

Random Needles Additionally, we examine LLoCO's capability in a more complex scenario by employing a unique "needle" in each trial. Following (Liu et al., 2024a), we randomly choose a designated city to serve as the needle for each position. We enhance the NQA LLoCO model through continual finetuning with a synthetic small-scale dataset comprising cities not encountered during evaluations. Figure 4 reveals that although the NQA-finetuned model struggles initially, further finetuning with the LLoCO pipeline substantially elevates success rates to $\sim$80%. This improvement underscores the efficacy of our method in handling the Needle In A Haystack task.

Setup	QuA	QAS	QM	NQA	HQA	Avg.
LLaMA2-7B-4k	40.45	16.67	14.62	14.42	32.47	23.44
LLaMA2-7B-32k	38.88	21.72	14.58	16.76	31.58	24.70
LLaMA2-7B-4k w. Finetuning	35.00	17.80	15.49	21.41	**41.89**	26.32 (+2.88%)
LLaMA2-7B-32k w. Finetuning	40.12	**29.71**	16.36	**28.72**	41.68	**31.32** (+6.62%)
LLoCO	**41.51**	29.01	**16.68**	28.34	37.83	30.67

Table 3: Comparison of LLoCO with LLaMA-7B baselines after in-domain finetuning.

	SingleDoc			MultiDoc			Summarization			
	NQA	QAS	MFE	HQA	WMQA	MSQ	Gov	QMS	MNews	Avg.
LLaMA-2-7B-4k	18.7	19.2	36.8	25.4	32.8	9.4	27.3	20.8	25.8	24.0
LLaMA-2-7B-32k	16.9	**27.7**	**41.4**	32.5	20.6	9.7	**30.8**	22.7	26.4	25.4
LLoCO	**23.1**	26.1	26.3	**46.2**	**35.6**	**27.3**	17.6	**23.4**	25.0	**27.8**

Table 4: Evaluation results on LongBench for SingleDoc, MultiDoc and Summarization. Numbers for LLaMA-2-7B-4k/32k are taken from the official LongBench's repository (THUDM, 2023)

4.5 Inference Latency

In this section, we evaluate the inference latency improvements of our LLoCO method. The experiments are run on a single A100-80G-SXM GPU and a RTX A6000 GPU, both with a batch size of 1 and a generation token count set to 16. We measured the per-token latency with various context sizes. As illustrated in Figure 5, LLoCO realizes speed-ups of up to 7.62× on A100 and 7.19× on A6000 when compared to the LLaMA2-7B baseline without compression, under identical context conditions. While the baseline exceeds GPU VRAM limits for sequences longer than 32k tokens, LLoCO maintains efficient generation for sequences up to 128k tokens. Notably, LLoCO achieves the baseline's 4k latency for sequences that are **16× longer (64k) on the A100** and **32× longer (128k) on the A6000**. Furthermore, LLoCO can process 32k sequences on A6000 as fast as baseline's 4k on A100.

We additionally measure LLoCO's finetuning throughput improvement over finetuning LLaMA with the original context, where LLoCO achieves a up to 11.52× higher throughput. The detailed results can be found in Appendix C.

5 Conclusion

We propose LLoCO, a new paradigm that addresses the long-context challenge by preprocessing the context before inference through parameter-efficient finetuning. Our approach extends the effective context window of a LLaMA2-7B model with a 4k tokens context size to handle up to 128k tokens. Evaluations on extensive long-context benchmarks show that LLoCO significantly outperforms in-context learning while using 30× fewer tokens during inference, making it a promising solution for efficient long-context processing.

6 Limitations

One limitation of our work is that the context encoder is tied to a specific LLM model. It compresses the original context into a representation aligned with the model. Therefore, for a different model, a separate context encoder must be trained to ensure alignment. Training a context encoder such as AutoCompressor requires data at the scale of 1 billion tokens, which is relatively expensive. Developing a model-agnostic context encoder that can adapt to different models without additional training poses an important challenge and is an interesting future direction.

In LLoCO's pipeline, we separate documents into groups and finetune a LoRA adapter for each group. In a standard RAG pipeline, top k documents relevant to the user's query are retrieved. However, these k documents may come from different groups, each corresponding to a different LoRA adapter, and we can apply only one adapter at a time, which is a limitation. An intriguing future direction would be to investigate the composition of multiple LoRA adapters simultaneously to further enhance QA performance.

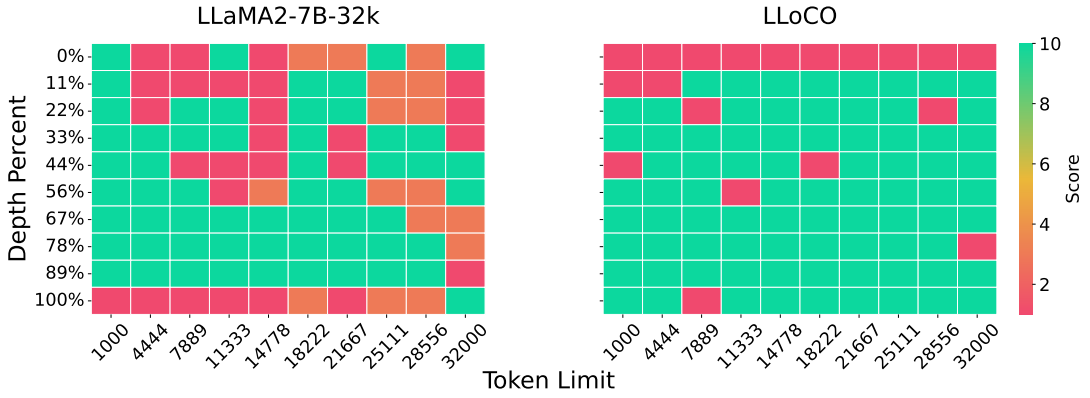

Figure 3: Fixed needle retrieval task. The sampled article ("haystack") starts with "`Mary, ..., a gentle, fashionable girl...`", and a context-relevant needle was curated as "`Mary's favorite fashion designer was Coco Chanel when she was a teenager. Who was Mary's favorite fashion designer when she was a teenager?`"

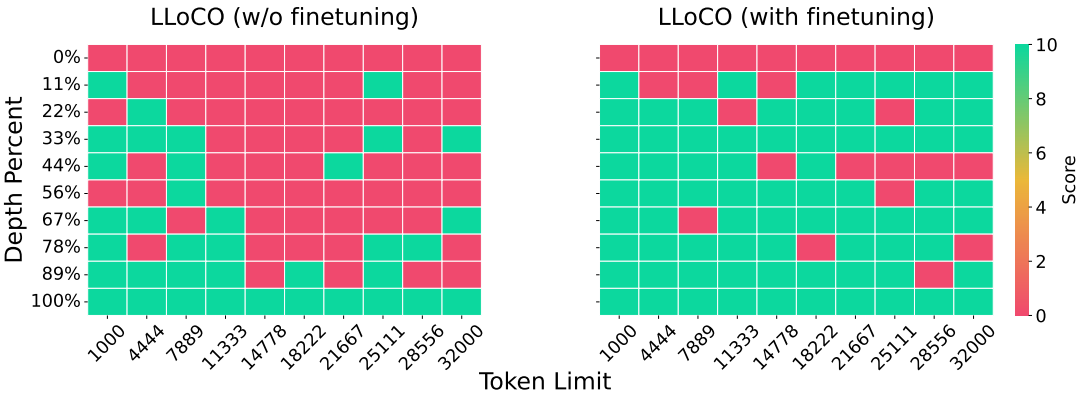

Figure 4: Random needle retrieval with city-word pairs.

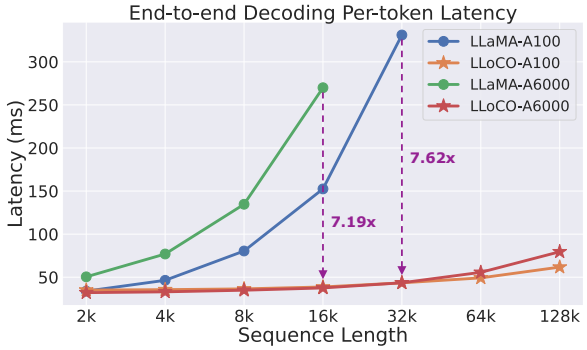

Figure 5: End-to-end decoding per-token latency (ms) on A100 and A6000 GPUs. LLaMA2 without compression runs out of VRAM for sequences of 64k and 128k on A100, and for 32k sequences on A6000.

There are differences in the effectiveness of LLoCO across various datasets and tasks. As shown in Section 4.3 and 4.4, our performance on GovReport and MultiField-En does not surpass the baselines, and we do not achieve perfect results on the Needle-In-A-Haystack task. We attribute these issues to the limitations of the training data quality and the capabilities of the context encoder and LLaMA2 base model. We believe that improving these components will address these shortcomings.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv: 2310.11511*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. 2023a. Punica: Multi-Tenant LoRA Serving.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv: 2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. *Preprint*, arXiv:1906.01749.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context Autoencoder for Context Compression in a Large Language Model. *arXiv preprint*. ArXiv:2307.06945 [cs].

gkamradt. 2023. Needle in a haystack - pressure testing llms. [Accessed 26-03-2024].

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *International Conference on Machine Learning*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv: 2401.18079*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv: 2307.13269*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. *arXiv preprint arXiv:2104.02112*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *arXiv preprint arXiv: 2310.06825*.

Huiqiang Jiang, Qianhui Wu, , Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *ArXiv preprint*, abs/2310.06839.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023c. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023d. Active retrieval augmented generation. *Conference on Empirical Methods in Natural Language Processing*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Neural Information Processing Systems*.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023. How long can context length of open-source LLMs truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with blockwise ringattention. *arXiv preprint arXiv: 2402.08268*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024b. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *Annual Meeting of the Association for Computational Linguistics*.

Jesse Mu, Xiang Lisa Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. *Neural Information Processing Systems*.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv: 2402.05859*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *ArXiv preprint*, abs/2403.12968.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Sam Bowman. 2021. Quality: Question answering with long input texts, yes! *North American Chapter of the Association for Computational Linguistics*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Neural Information Processing Systems*.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. 2023a. S-LoRA: Serving Thousands of Concurrent LoRA Adapters. *arXiv preprint*. ArXiv:2311.03285 [cs].

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023b. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, E. Chi, Nathanael Scharli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *International Conference on Machine Learning*.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *NEUROCOMPUTING*.

THUDM. 2023. Longbench: A benchmark for long-range language models. `https://github.com/THUDM/LongBench`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. *arXiv preprint arXiv:2402.16617*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024a. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2024b. H2o: Heavy-hitter

oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

A Extended Experimental Settings

In this part, we provide details of our experimental settings in the main text. For the experiments presented in Sec 4.1, we summarize the hyperparameter settings in Table 5. We use all samples from the train split of each dataset for finetuning, except for HotpotQA, from which we randomly select $20,000$ samples from its train split. All finetuning experiments are conducted with either 4 or 8 NVIDIA RTX A6000 GPUs or A100-80G-SXM GPUs.

B More Details on Datasets

Here we provide more detailed descriptions of the 5 main datasets used for evaluation:

- **QuALITY** (Pang et al., 2021) is a multiple-choice question-answering dataset over long contexts. It contains 150 articles with an average length of 5000 tokens, with 6737 questions in total. This dataset is particularly suited to evaluate models in a long-context setting.

- **Qasper** (Dasigi et al., 2021) is a dataset for answering questions about NLP research papers, sourced from the Semantic Scholar Open Research Corpus. It includes various types of questions, ranging from detailed explanations to simple yes/no answers, based on provided documents.

- **QMSum** (Zhong et al., 2021) features summaries and transcripts from meetings across various sectors, including academia and industry, focusing on query-based summarization. Participants are tasked with condensing dialogue transcripts based on specific queries.

- **NarrativeQA** (Kočiskỳ et al., 2018) is a question-answering dataset derived from complete texts of books from Project Gutenberg and movie scripts from various sources. The challenge here involves generating a concise answer from long and potentially disorderly texts sourced from books.

- **HotpotQA** (Yang et al., 2018)) is a Wikipedia-based dataset that challenges users to derive answers through multi-hop reasoning across several documents, featuring a broad range of questions beyond specific knowledge bases.

We additionally provide the statistics of these five datasets in Table 6.

C Finetuning Throughput

We assess the finetuning throughput for both LLoCO and LLaMA2-7B-32k with the original context on the NarrativeQA dataset, which mainly consists of samples exceeding 32k tokens. We compare the 7B 32k baseline finetuned on 8 A100s and our LLoCO finetuned on both 8 A100s and 8 A6000s, all with a per-device batch size of 1 and a global batch size of 32, using 4 gradient accumulation steps. For the 7B 32k baseline finetuning, samples of more than 32k tokens were truncated to 32k. Figure 5 shows that our LLoCO achieves $11.52\times$ and $6.82\times$ throughput on A100s and A6000s, respectively. This highlights that LLoCO not only achieves competitive results, but also improves performance with much greater efficiency compared to finetuning the baseline with full context (as shown in Section 4.2).

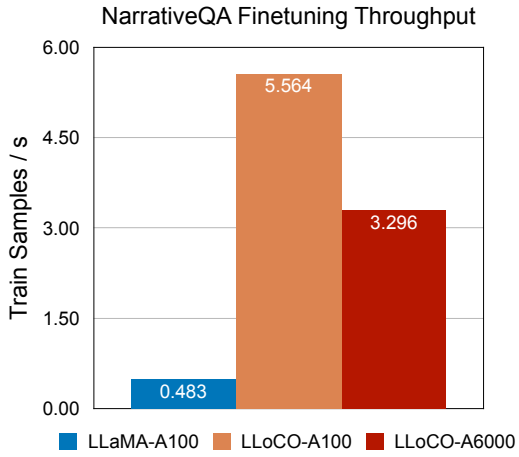

Figure 6: Samples per second when finetuning on NarrativeQA for LLaMA2-7B-32k without compression and LLoCO.

D Additional Ablation Study

Combined Instruction Finetuning In our previous experiments, we finetuned LLoCO on each dataset separately. In this ablation study, we investigate LLoCO's performance when we combine all the training data and finetune a general model. To balance the dataset, we sample 10,000 examples from NarrativeQA and HotpotQA and use all examples from the other three datasets to form our final training set.

As shown in Table 7, LLoCO with combined finetuning surpasses baseline methods on all

Hyperparameters	QuA	QAS	QM	NQA	HQA
LoRA Rank r	8				
Optimizer	AdamW				
Weight Decay	0.00				
Learning Rate	2×10^{-5}				
LR Scheduler	cosine annealing				
Batch Size	8	8	8	32	32
Warmup Ratio	0.04				
Epoch	3	3	3	1	3

Table 5: Hyperparameter settings of experiments.

	QuA	QAS	QM	NQA	HQA
# of samples	2086	1726	272	5878	7405
# of unique articles	115	273	35	115	7306
Avg word count	4122	3572	10422	51925	952
Max word count	5967	14640	24573	353480	2694
Max token length	9953	24205	34478	556249	4000

Table 6: Statistics of the datasets. The maximum token length is counted using LLaMA2-7B's tokenizer.

datasets except QMSum. The lower performance on QMSum can be attributed to the fact that it is a summarization task that favors longer, more detailed answers, but the combined finetuning process likely shifts the LLM's behavior towards generating shorter, more focused answers, which may not be optimal for the summarization task.

Compared to in-domain finetuning, combined finetuning generally yields lower performance, with the exception of QuALITY. QuALITY questions heavily rely on the LLM's reasoning abilities, whereas the other datasets primarily focus on the LLM's capacity to retrieve relevant information from long contexts. We hypothesize that the additional finetuning on diverse datasets enhances the LLM's ability to reason about long contexts, leading to improved performance on QuALITY. Overall, the combined finetuning approach demonstrates the potential for knowledge transfer across different tasks.

Exploring other Context Encoders By default, LLoCO utilizes AutoCompressor as its context encoder. However, LLoCO offers flexibility in its choice of context encoders and is not limited to a single method. In this ablation study, we evaluate LLoCO using another state-of-the-art context compression method, ICAE (Ge et al., 2023), as the context encoder for LLoCO. Since ICAE was trained on contexts up to 4096 tokens, we truncated the inputs to this length, resulting in compressed embeddings of 128 tokens, following the approach outlined in the original paper.

Our results in Table 8 show that ICAE performs comparably to LLaMA2-7B-4K across benchmarks, while achieving a 40x compression ratio. This highlights that our proposed pipeline, which combines context compression with PEFT, is not restricted to AutoCompressor and can be generalized to other context encoders.

However, there remains a notable performance gap between ICAE and LLoCO. A key limitation of ICAE is its inability to extend the context window beyond 4K tokens, making it a less suitable option compared to AutoCompressor as the context encoder for our approach.

E LLoCO's Serving Stage

Serving In a traditional RAG system, when a user asks a question, the system employs a retriever to fetch the top k relevant documents/passages from the vector database and concatenates them to the prompt to serve as the relevant context.

In Figure 7, we illustrate LLoCO's serving pipeline. In a standard RAG, the retriever computes a text embedding for each passage, and use that as the key to retrieve passages relevant to the

Setup	QuA	QAS	QM	NQA	HQA	Avg.
LLoCO w. Separate Finetuning	41.51	**29.01**	**16.68**	**28.34**	**37.82**	30.67
LLoCO w. Combined Finetuning	**47.07**	24.95	12.77	27.93	35.55	29.65

Table 7: Comparison of LLoCO with in-domain finetuning and combined finetuning.

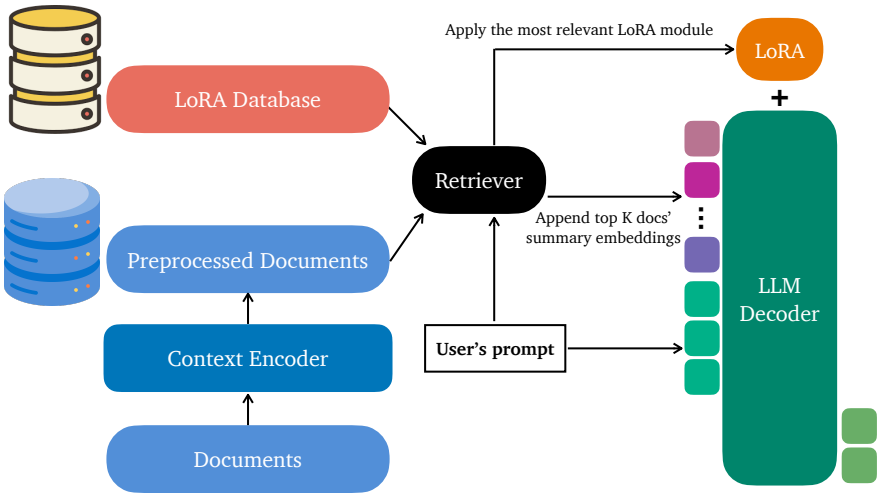

Figure 7: The serving stage of LLoCO's pipeline.

Setup	QuA	QAS	QM	NQA
LLaMA2-7B-4k	40.45	16.67	14.62	14.42
ICAE	36.5	16.14	15.69	17.38
LLoCO	**41.51**	**29.01**	**16.68**	**28.34**

Table 8: Comparison of LLoCO with ICAE (Ge et al., 2023)

user query. We use the same text embedding as key, but instead retrieve the compressed token embeddings of the passages, and prepend it to the decoder LLM. Additionally, the system searches for the corresponding LoRA adaptor in the database and applies it to the decoder LLM. Applying a LoRA adaptor incurs minimal overhead to the system cost (Hu et al., 2022). By leveraging recent work on serving multiple LoRA adaptors (Sheng et al., 2023a; Chen et al., 2023a), we can parallelize LoRA adaptor serving and simultaneously serve requests for documents associated with thousands of LoRA adaptors on a single GPU.

In our current system, we assume that all passages retrieved from the vector database for a given user query belong to the same document group, allowing us to use a single dedicated LoRA adaptor. However, LLoCO can be extended to handle cases where the retrieved passages span multiple document groups. For instance, it is feasible to design algorithms to dynamically select the most relevant LoRA adaptor based on the majority of the retrieved documents or weight the contributions of different adaptors by their relevance to the query (Muqeeth et al., 2024). Another interesting and orthogonal research direction is to explore composing multiple LoRA modules together (Huang et al., 2023), and we leave it as future work.

F Comparison with Concurrent Work

CEPE (Yen et al., 2024) and SnapKV (Li et al., 2024) are two concurrent work that also proposes methods for context compression. We provide a comparison with them in this section.

Comparison with CEPE Similar to LLoCO, CEPE requires training an encoder to generate embeddings for the context. During inference, LLoCO uses cross-attention to integrate these embeddings into the original LLM decoder. The main difference is that CEPE applies the encoder at runtime, allowing it to process context in parallel, thus speeding up the inference process. In contrast, LLoCO performs context encoding offline, retrieving the preprocessed embeddings during inference.

To compare with CEPE, we evaluate their model

(based on LLaMA2-7B-Chat) on QuaLITY, Qasper, QMSum, and NarrativeQA. We present two sets of results: the official evaluation performance provided by CEPE, and our reproduced results, obtained by running their publicly available code on these datasets.

Setup	QuA	QAS	QM (ROUGE-L)	NQA
CEPE (official)	30.2	20.5	19.6	21.9
CEPE (reproduced)	26.40	20.54	19.38	20.33
LLoCO	**41.51**	**29.01**	**20.92**	**28.34**

Table 9: Comparison of LLoCO with CEPE (Yen et al., 2024)

As shown in Table 9, LLoCO outperforms CEPE across all datasets. Note that for QMSum dataset, CEPE provided the ROUGE-L score, while LLoCO provides the geometric mean of ROUGE-1, ROUGE-2, and ROUGE-L scores as our metric. We take the ROUGE-L score of LLoCO here for a fair comparison.

Comparison with SnapKV Compared to LLoCO and CEPE (Yen et al., 2024), which requires first finetune a encoder beforehand, SnapKV is a KV cache compression approach that is finetuning free. The key insight behind SnapKV is that only a small subset of tokens in the prompt is crucial (those that other tokens attend to), allowing the less important tokens to be discarded.

Since both LLoCO and SnapKV provide evaluation results on LongBench, we offer a side-by-side comparison of their performance. We extract SnapKV's results using LLaMa-2-7B-32k (LongChat) as the base model and compare them with ours. SnapKV provides three settings (1024, 2048, and 4096 tokens), and we use their best results for comparison.

Overall, LLoCO and SnapKV show comparable performance across tasks. LLoCO outperforms SnapKV on five datasets, while SnapKV leads on four. Unlike SnapKV, which compresses input tokens to fixed lengths (1024, 2048, 4096), LLoCO applies a consistent 30x compression ratio, enabling the compression of up to 128k tokens into 4096 tokens. This generally results in a higher compression ratio than SnapKV.

	SingleDoc			MultiDoc			Summarization			
	NQA	QAS	MFE	HQA	WMQA	MSQ	Gov	QMS	MNews	Avg.
SnapKV	20.7	**29.3**	**42.2**	34.0	24.9	14.2	**28.6**	23.1	**26.5**	27.1
LLoCO	**23.1**	26.1	26.3	**46.2**	**35.6**	**27.3**	17.6	**23.4**	25.0	**27.8**

Table 10: Comparison between LLoCO and SnapKV (Li et al., 2024) on LongBench (THUDM, 2023)