MemLong: Memory-Augmented Retrieval for Long Text Modeling

Anonymous ACL submission

Abstract

Recent advancements in Large Language Models (LLMs) have yielded remarkable success across diverse fields. However, handling long contexts remains a significant challenge for LLMs due to the quadratic time and space complexity of attention mechanisms and the growing memory consumption of the key-value cache during generation. This work introduces **MemLong**: **Mem**ory-Augmented Retrieval for **Long** Text Modeling (MemLong), a method designed to enhance the capabilities of long-context language modeling by utilizing an external retriever for historical information retrieval. MemLong combines a non-differentiable *ret-mem* module with a partially trainable decoder-only language model and introduces a fine-grained, controllable retrieval attention mechanism that leverages semantic-level relevant chunks. Comprehensive evaluations on multiple long-context language modeling benchmarks demonstrate that MemLong consistently outperforms other state-of-the-art LLMs. More importantly, MemLong can extend the context length on a single 3090 GPU from 4k up to 80k[1].

1 Introduction

Large Language Models (LLMs) have achieved remarkable success in various fields. However, due to the quadratic time and space complexity of vanilla attention mechanisms (Vaswani et al., 2017), it is challenging to extend the context length considerably, which poses significant limitations for applications involving long-sequence tasks, such as long-document summarization (Koh et al., 2022) and multiple rounds of dialogue (Wang et al., 2024a). As a result, LLMs are often expected to maintain a long working capability (a.k.a. long context LLMs) to effectively handle these demanding scenarios.

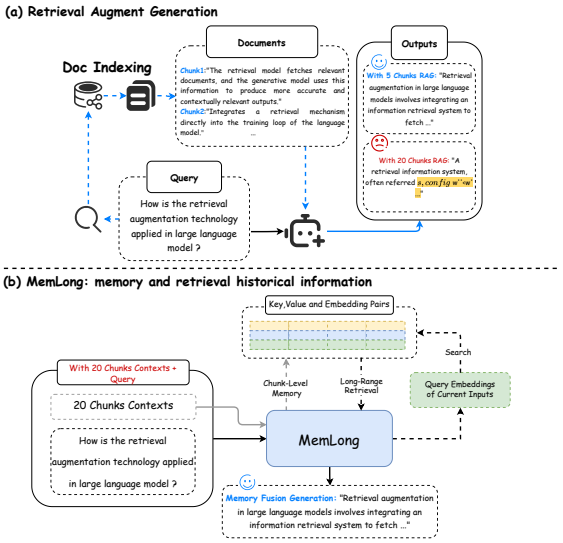

Figure 1: **Illustration of Retrieval-Augment Generation(RAG) and Memory-Retrieval flow of MemLong.** (a) RAG can even degrade the generation performance (yellow) when the length of the retrieved information exceeds the model's processing capacity. (b) Our approach utilizes an external retriever to fetch historical information, which is then passed into the model as K-V pairs rather than in text form.

To tackle the computational bottleneck, numerous efforts have been made. The first line of work focuses on reducing the computation of vanilla attention mechanisms (Vaswani et al., 2017) by employing sparse attention operations (Beltagy et al., 2020; Wang et al., 2020; Kitaev et al., 2020; Xiao et al., 2023a; Chen et al., 2023b; Lu et al., 2024). Although these types of works can reduce computational complexity to approximately $\mathcal{O}(n)$, it often comes with trade-offs in model capability. Therefore, Some works shift their focus to memory selection (Dai et al., 2019; Bertsch et al., 2024; Yu et al., 2023). These approaches, as token-level memory selection, can result in the truncation of semantic information. Another recent line of work is Retrieval-Augment Language Modeling (Wu et al., 2022; Wang et al., 2024b; Rubin and Berant, 2023).

[1]Our code will be available at https://anonymous.com

These works usually introduce a retrieval mechanism to enhance the model's ability to handle long texts. However, these methods have several drawbacks. Firstly, the information stored in memory may experience *distribution shifts* due to changes in model parameters during training. Secondly, these methods often require retraining, which is impractical in the era of large models. Finally, these models are often prone to processing long text inputs at the expense of the original capabilities of the pre-trained model. To address the limitations of previous research, we posed the following question: **Can we utilize the explicit retrieval capabilities of a retriever to approximate the implicit retrieval processes within the model?**

In this work, we propose MemLong, an efficient and lightweight method to extending the context window of LLMs. **The key idea is to store past contexts and knowledge in a non-trainable memory bank and further leverages these stored embeddings to retrieve chunk-level key-value (`K-V`) pairs for input into the model.**. MemLong is applicable to any decoder-only pretrained language models by incorporating (1) an additional *ret-mem* component for memory and retrieval, and (2) a *retrieval causal attention* module for integrating local and memory information. The memory and retrieval process of MemLong is illustrated in Figure 1(b). During generation,one text that exceeds the model's maximum processing length is stored as context information in a Memory Bank. Subsequently, given a recently generated text chunk in a long document, we use the retriever to explicitly retrieve past information, obtaining additional context information through index alignment.

MemLong offers several benefits: (1) **Distributional Consistency**: Unlike previous models that experienced a *distribution shift* when information was stored in memory, MemLong ensures the distribution of cached information remains consistent. (2) **Training Efficient**: We freeze the *lower layers* of the model and only need to finetune the *upper layers* which greatly reduced computational cost. In our experiments, finetuning a 3B parameter version of MemLong on 0.5B tokens requires only eight 3090 GPUs for eight hours. (3) **Extensive Context Window**: Since only a single layer's `K-V` pairs need to be memorized, MemLong is capable of extending the context window up to 80k tokens easily on a single 3090 GPU.

Extensive experiments have demonstrated that MemLong exhibits superior performance in several aspects when compared with other leading LLMs. MemLong outperforms OpenLLaMA (Touvron et al., 2023) and other retrieval-based models on several long-context language modeling datasets. In retrieval-augmented in-context learning tasks, MemLong achieves an improvement of up to 10.2 percentage points over OpenLLaMA.

2 Preliminary

2.1 Task Definition

Language models are designed to define probability distributions over sequences of tokens, effectively predicting the likelihood of a sequence within a given language. Given such a sequence $x_1, \ldots, x_n$, the standard approach to modeling its probability is via the next-token prediction: $p(x_1, \ldots, x_n) = \sum_{i=0}^{n} p_\theta(x_i|x_{<i})$, where $x_{<i} := x_1, \ldots, x_{i-1}$ is the sequence of tokens proceeding x_i. Differently from the standard language modeling objective, we not only use the current context to make next-token predictions, but also utilize external retrieval to obtain relevant information and perform knowledge fusion at the upper layers of the model. Specifically, given a sequence consisting of l tokens and the size of each chunk τ, we partition it into a long sequence of $\nu = \frac{l}{\tau}$ non-overlapping chunks , which denoted as $\mathcal{C} = (c_1, \ldots, c_\nu)$. Correspondingly, its textual form is divided into ν text chunks, which denoted as $\mathcal{T} = (t_1, \ldots, t_\nu)$. In each step, we perform causal language modeling on c_i in the *lower layers*, while in the *upper layers*, we conduct fine-grained controllable retrieval on t_i for the fusion of additional information. After do this, our language modeling objective becomes

$$p(x_1, \ldots, x_n) = \sum_{i=0}^{n} p_\theta(x_i|\mathcal{R}(t_i), x_{<i}) \quad (1)$$

where $\mathcal{R}(t_i)$ denotes the retrieval of *neighboring chunks* of t_i where x_i is located.

2.2 Module and Operation Definitions

As shown in Figure 2, the *Ret-Mem* module comprises a *Retriever* and a *Memory* component for information exchange. Initially, we define the Memory component as $\mathcal{M}$ and the Retriever as $\mathcal{R}$, and their corresponding operations $\mathcal{M}(\cdot)$ and $\mathcal{R}(\cdot)$. Furthermore, we specify the dimension of the model as d_{model} , the dimension of the retriever as d_{ret}. The *Memory* module includes two segments: `K-V` pairs and corresponding Representation

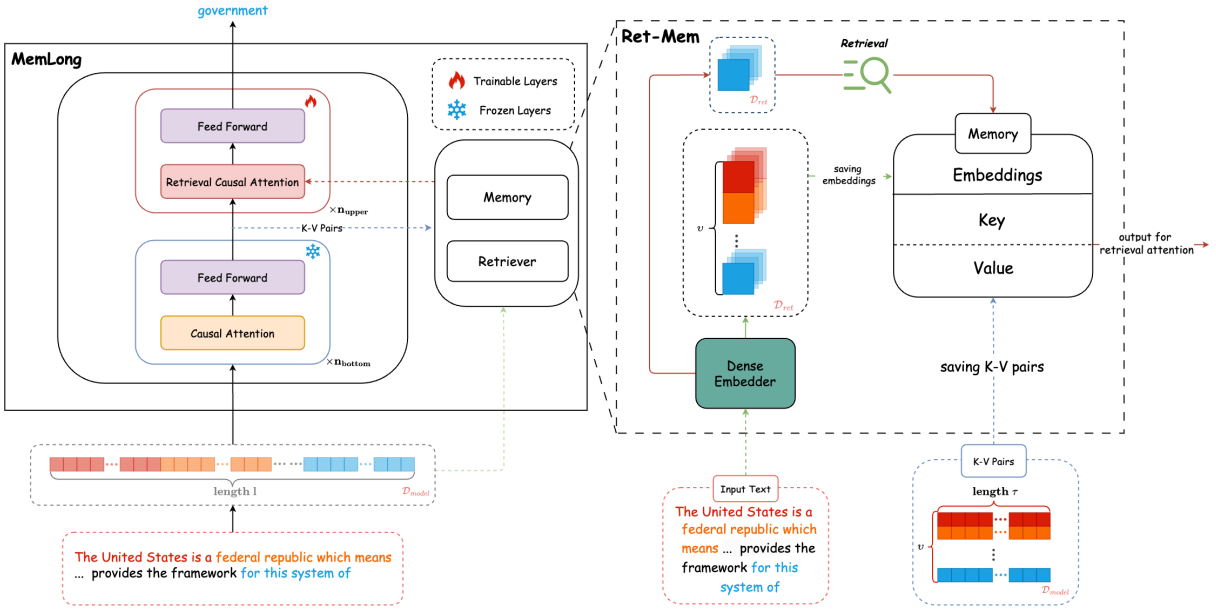

Figure 2: An example of MemLong : In the *lower layers*, where the model remains static, causal language modeling is performed on the entire chunk c_i, and subsequently, c_i is cached in both embedding and K-V pair forms. Lastly, the *upper layers* are finetuned to harmonize retrieval preferences and integrate the retrieved content.

Embeddings. The dimension for both keys and values is represented as $\mathbb{R}^{d_{model}}$ and for Embeddings as $\mathbb{R}^{d_{ret}}$. It is crucial to emphasize that the actual retrieval process involves the embeddings representing the chunks, not the K-V pairs. The *Retriever* is essentially a pretrained dense embedder with excellent representation capabilities. MemLong use it to encode each chunk into Representation Embeddings. Since it produces a one-dimensional representation vector for one chunk, the memory footprint remains minimal even if the memory size is substantial.

3 MemLong

3.1 Overview

As illustrated in Figure 2, each step involves an input of a chunk c_i, where the original text for that chunk is t_i. In the *lower layers* where the model is frozen, the standard causal attention is applied to the entire c_i. For the final layer of the *lower layers*, we refer to it as the *memory layer*. Following each traversal of the *memory layer*, two key operations are performed. The first operation is retrieval, depicted by the red line, where t_i is utilized to fetch the most pertinent K-V pairs. The second operation, indicated by the blue line, involves caching the acquired K-V pairs along with their associated chunk representation. Within the model's *upper layers*, the retrieved K-V pairs are integrated with the current input context, subsequently tuning the model parameters to calibrate the retrieval reference. Subsequent sections will explore the various facets of the MemLong framework and their intricacies, encompassing Retriever and Dynamic Memory Management (§ 3.2), Attention Reformulation (§ 3.3), and Inference with MemLong (§ 3.4).

3.2 Retriever and Dynamic Memory Management

We offer a comprehensive explanation of the retrieval process and the dynamics of memory management.

Retrieval Process. Given our objective to replace traditional kNN retrieval based on K-V pairs with explicit retrieval, we aim to pre-fetch the desired information when feasible before each model input. Specifically, for each potential query block $c^q = c_i$ and its corresponding text block $t^q = t_i$, we first pass it through *Retriever* and then obtain a representation embedding $r^q = \mathcal{R}(t^q)$, where $r^q \in \mathbb{R}^{d_{ret}}$. Subsequently, we use this representation embedding to perform retrieval against the embeddings in $\mathcal{M}$ to obtain the required k chunk-level indices. We compute the cosine similarity between the retrieval representation r^q and the embeddings stored in Memory $\mathcal{M}$. Finally , we get the top-k indices $z^q = \texttt{TopK}\{\texttt{Cos}\,(r^q)\}$ for the c^q, where $z^q \in \mathbb{R}^k$. Due to the contiguous nature within the blocks, we can easily extend the obtained in-

dices to cover the entire relevant range for retrieval. Finally, we retrieve the corresponding K-V pairs $\tilde{z}^q \in \mathbb{R}^{k\times\tau\times d_{model}}$ from Memory based on these indices and used for the *upper layer*. It is noteworthy that we have equipped the *Memory* with a counter mechanism to record the frequency of retrievals for each index contained therein. This frequency data will subsequently serve as a basis for dynamic memory updating, allowing for the prioritization of more frequently retrieved information.

Memory Process. The memory process synchronously stores the K-V pairs from the *memory layer* and the *representation embedding* previous calculated for retrieval , ensuring that indices for K-V pairs correspond accurately to their representation embeddings (see Figure 2, right, blue line). For every possible chunk memory $c^m = c_i$, and its corresponding text chunk $t^m = t_i$, we divide the memory process into two parts: the first part details how to cache the K-V pairs, and the second part explains how to store the corresponding representations. Firstly, we input c^m into the MemLong and get the output from the *memory layer*. It is worth noting that, since the lower layers are frozen during training, we can ensure that the distribution of the output K-V pairs is consistent. This consistency is crucial for avoiding the distribution shift issue, which was previously observed in models like MemTrm (Wu et al., 2022). Our memory operation is highly efficient because it only involves storing the representations needed for retrieval, $r^m = r^q$, thereby avoiding redundancy. After the retrieval for all chunk pairs is complete, the memory operation—denoted as $\mathcal{M}(k, v; r^m)$—synchronously updates the memory with both the Key-Value pairs and their corresponding representations.

Dynamic Memory Update. When memory overflows, we use the Counter to update memory intelligently. In our experiments, we keep the latest 10% of memory content due to its potential relevance, discard the oldest 10% as likely outdated, and prioritize the middle 80% based on retrieval frequency, deleting the least accessed entries until memory usage drops to 50%. This selective pruning balances recency and relevance, retaining valuable information and removing less pertinent data. Unlike traditional FIFO strategies, our method focuses on retrieval frequency to efficiently prune redundant information, maintaining a high-quality dataset. The decision to dynamically update the datastore is a trade-off between effectiveness and efficiency. For tasks requiring long-term dependencies, storing all information can enhance comprehensive processing, but for shorter-term tasks, dynamic updates are more suitable. Dynamic updates control memory size to prevent out-of-memory issues, discard stale information, and reduce retrieval overhead, ensuring efficiency without significantly compromising performance.

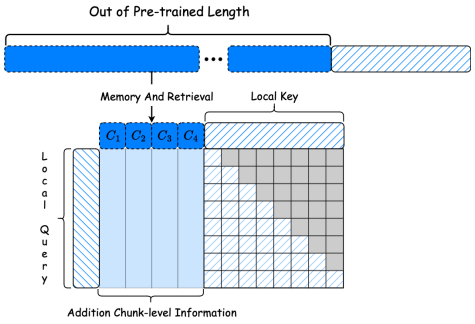

Figure 3: Illustration of retrieval causal attention. Local causal attention is applied to the recent context, while chunk-level K-V pairs, obtained through the retrieval method, enable bidirectional attention without information leakage due to their historical nature.

3.3 Attention Reformulation

In the trainable *upper layers* of the model, we revised the attentions to fuse with long-term memory. As illustrated in Figure 3, unlike the traditional Transformer decoder layers that utilize Multi-Head Attention (Vaswani et al., 2017), we propose a *Retrieval Causal Attention* to extend it to a joint-attention mechanism and propose a long-term memory fusion process to enable each token to attend on both local contexts and chunk-level past contexts which have complete and continuous semantics. With the head-wise hidden state output from previous layer $H^{l-1} \in \mathbb{R}^{|x|\times d_{model}}$ and the corresponding retrieved key-value pairs are $\tilde{z}^q = \{\tilde{K}_i, \tilde{V}_i\}_{i=1}^{\omega} \in \mathbb{R}^{k\times\tau\times d_{model}}$, the output hidden state for the next layer H^l is computed as:

$$S_a = \texttt{Softmax}\left(\frac{QK^T}{d}\right) \tag{2}$$

$$S_m = \texttt{Concat}\left\{\texttt{Softmax}(\tilde{\mathbf{z}}_\mathbf{i}^\mathbf{q})\right\}_{i=1}^{\omega} \tag{3}$$

To avoid the interference caused by the retrieval attention scores S_m at the initial stage of training, we adopt a multi-head attention mechanism following the approach of the LLaMA-adapter(Zhang et al., 2023b) :

$$S_l^g = [(S_m) \cdot g_l; (S_a)]^T \tag{4}$$

Finally, we concatenate the $\tilde{V}$ and V to obtain H^l:

$$V_l = \left[\tilde{V}_c; V_i\right], H^l = S_l^g V_l \quad (5)$$

3.4 Inference with MemLong

When MemLong receives an input exceeding the length, we treat it as two segments: the *prefix* and the *main*. We will separately describe the encoding of long inputs and the generation of long outputs during the inference phase. When MemLong receives long inputs, it first divides the *prefix* into multiple non-overlapping chunks and computes the from its *memory layer*, which ensures that the number of tokens involved in the attention is equal to the chunk size, which is much smaller than the length of the input. It is important to note that each chunk is interrelated (e.g., the t-th chunk needs to process the of the previous $t-1$ chunks).

The second step is to select the k most relevant chunks for the *main* based on chunk-level retrieval representations and to obtain their key and value representations. After this, for the upper retrieval layers, the attention window for retrieval is equivalent to $k * \tau$, which is also smaller than the input length. Finally, both length-restricted causal attention and retrieval attention is performed efficiently.

4 Experiments

We evaluate our proposed MemLong model on various tasks that require in-memory long-context processing: (a) long-context language modeling and retrieval-augmented language modeling; (b) scalable in-context learning capable of handling a large number of demonstration examples in memory.

4.1 Implementation Details

Training Details. We use OpenLLaMA-3B as the pre-trained backbone LLM with Rotation position coding (Su et al., 2024). Due to hardware constraints, we opted to train our models using the LoRA (Hu et al., 2021) technique. The backbone LLM holds a $L = 26, H = 32, d = 100$ architecture. Unless specified otherwise, we use the 13-th layer as *the memory layer* and the [14,18,22,26] layers as the *retrieval-augment* layers. The training for retrieval-augmented adaptation iterates only on 0.5B tokens with 1024 sequence length. MemLong's trainable parameters are from 14 to 26 layers. We utilized the slimpajama dataset sampled by (Fu et al., 2024) as our training corpus.

Position Remapping. There are several chunk-level K-V in the $\mathcal{M}$ retrieved for generation. Due to the uncertainty of retrieval at each step, we need to remap position embeddings to the retrieved chunks. Same as the previous work (Tworkowski et al., 2024), The local context (up to 2048 tokens) receives the standard rotary positional encoding, whereas memory keys are encoded as if they had position 0 in the local context window.

4.2 Long-Context Language Modeling

We first evaluate MemLong on long-context language modeling benchmarks to assess basic language modeling abilities. Due to the K-V cache providing sinificant background and contextual information, MemLong can retrieve relevant K-V cache quickly and make full use of it, thereby enhancing the model's in long-context modeling tasks.

Datasets. We conducted an evaluation of our model across four extensive text benchmark datasets: English-language books PG-19 (Rae et al., 2019) and BookCorpus (Zhu et al., 2015), Wikipedia articles Wikitext-103 (Merity et al., 2016), and mathematical papers Proof-Pile (Azerbayev et al., 2023). The experimental results indicate a significant perplexity improvement across all datasets. Our model was tested over various lengths ranging from 1024 to 32768 tokens. Across all datasets, our model demonstrated substantial performance gains with minimal memory overhead by leveraging an external retriever and memory.

Setup. Following (Yen et al., 2024), we calculate the perplexity on the last 2048 tokens of each sequence. This experimental setup was designed to validate the influence of different retriever sizes on the overall performance of our model. For the implementation of the efficient fine-grained retrieval, we use the *faiss* (Johnson et al., 2019) toolkit to construct an exact-search index on GPU to store the *Representation Embeddings* of text chunks and perform efficient retrieval. For MemLong, we split and put the tokens over finetune-length $= 1024$ into the $\mathcal{M}$ used for further retrieval.

Baselines. For our experiments, we employ the OpenLLaMA-3B model as our baseline. To ensure a fair comparison, we utilize an identical LoRA configuration and finetuned the models on the same amount of data from the slimpajama dataset. Additional, we compare LongLLaMA-3B (Tworkowski et al., 2024), which finetuned with the Focused

Model	PG19				Proof-pile				BookCorpus				Wikitext-103			
	1k	2k	4k	16k	1k	2k	4k	16k	1k	2k	4k	16k	1k	2k	4k	16k
7B Model																
LLaMA-2-7B	10.82	10.06	8.92	-	3.24	3.40	2.72	-	8.73	7.91	6.99	-	10.82	6.49	5.66	-
LongLoRA-7B-32k	9.76	9.71	10.37	7.62	3.68	3.35	3.23	2.60	14.99	12.66	11.66	6.93	7.99	7.83	8.39	5.47
YARN-128k-7b	7.22	7.47	7.17	-	3.03	3.29	2.98	-	7.02	7.54	7.06	-	5.71	6.11	5.71	-
3B Model																
OpenLLaMA-3B	11.60	9.77	$> 10^3$	-	**2.96**	**2.70**	$> 10^3$	-	**8.97**	8.77	$> 10^3$	-	10.57	8.08	$> 10^3$	-
LongLLaMA-3B*	10.59	10.02	$> 10^3$	-	3.55	3.15	$> 10^3$	-	10.70	9.83	$> 10^3$	-	8.88	8.07	$> 10^3$	-
LongLLaMA-3B†	10.59	10.25	9.87	-	3.55	3.22	**2.94**	-	10.14	9.62	9.57	-	10.69	8.33	7.84	-
Phi3-128k	11.31	9.90	**9.66**	- / *9.65*	4.25	3.11	2.77	- / *3.08*	11.01	**9.22**	**8.98**	- / *9.27*	**7.54**	**7.22**	7.01	- / *7.20*
MemLong-3B*	10.66	10.09	$> 10^3$	-	3.58	3.18	$> 10^3$	-	10.37	9.55	$> 10^3$	-	8.72	7.93	$> 10^3$	-
w/ 4K Memory	10.54	9.95	9.89	**9.64**	3.53	3.16	3.15	2.99	10.18	9.50	9.57	9.61	8.53	7.92	7.87	7.99
w/ 32K Memory	**10.53**	**9.85**	**9.83**	9.73	3.51	3.15	3.11	**2.99**	9.64	9.56	**9.51**	**9.54**	8.02	7.58	**6.89**	**7.09**

Table 1: Sliding window perplexity of different context window extension models on PG19, Proof-pile, BookCorpus, Wikitext-103. All experiments are conducted on one 3090 24GB GPU. LongLLaMA-3B and MemLong-3B marked with * means evaluating without Memory, and LongLLaMA-3B marked with † means evaluting with infinite memory. We also evaluate MemLong with 4K/32K Memory scenarios. "- / 6.95" indicates that the model results in an Out of Memory (OOM) error on a single GPU, while on dual GPUs it yields the corresponding result.

Transformer (FoT) method and 5B tokens. To perform a further comprehensive comparison, we additionally test two 7B models: LLaMA-2-7B and LongLoRA-7B-32K (Chen et al., 2023b) and two positional encoding models: Yarn-7b-128k (Peng et al., 2023) and Phi3-128k (Abdin et al., 2024).

Results. The results are shown in Table 1. We employ Perplexity (PPL) as the evaluation metric for the language model. Lower PPL indicates stronger language modeling capabilities. Compared to the two fully fine-tuned models, OpenLLaMA-3B and LLaMA-2-7B, our model demonstrates comparable performance across multiple datasets when test lengths are within their pre-trained limits (2048 for OpenLLaMA-3B and 4096 for LLaMA-2-7B). However, once the test lengths exceed these pre-trained limits, our model continues to reduce perplexity even beyond the fine-tuning length of 1024 and the pre-trained length of 2048, showcasing its superior generalizability. In contrast, the OpenLLaMA-3B and LLaMA-2-7B models fail to generalize to inputs beyond their pre-trained lengths and exhibit significantly increased memory overhead due to the quadratic complexity of attention. We also compare our model with LongLoRA. Although the proposed Shifted Sparse Attention in LongLoRA significantly reduces memory usage, it also diminishes the model's performance on short texts. In contrast, LongLLaMA, which `K-V` pairs can also be stored, suffers from OOM issues when test lengths become excessively long due to its infinitely growing memory usage. **Compared to their methods, MemLong leverages an external retriever to handle longer input tokens and achieve better perplexity improvements.At the same time, because of the high storage efficiency, MemLong can effectively control the use of GPU to avoid OOM problems.**

4.3 In Context Learning

Traditional in-context learning (ICL; Brown et al., 2020) inputs few-shot non-parameterized demonstration examples along with the query into the model. However, these methods are typically constrained by the model's input length. In this experiment, since MemLong can store examples in a parameterized form within its memory, we primarily investigate whether MemLong can effectively utilize the knowledge stored in its memory to enhance its emergent abilities. The results are shown in Table 2. Compared to OpenLLaMA,which rely solely on non-parametric knowledge , given the same number of in-context demonstrations, MemLong can utilize additional demonstrations stored in its memory. The performance further increases or remains consistent with more demonstrations in the memory. In our comparative analysis against LongLLaMA, it was observed that our model outperforms LongLLaMA across the majority of datasets under the same conditions of preserving In-Memory Demonstrations. **It is important to highlight that our model operates with signif-**

Model	In-Context Demons.	In-Memory Demons.	SST-2 ACC↑	MR ACC↑	Subj ACC↑	SST-5 ACC↑	MPQA ACC↑	Avg.
OpenLLaMA	4	N/A	90.7	84.0	58.2	41.0	70.5	68.9
w./ Rag	4	4	90.9	90.5	61.6	39.2	63.2	69.1
LongLLaMA	4	4	90.4	83.9	64.3	40.0	64.2	68.6
MemLong	4	4	91.5	84.5	61.5	41.4	70.2	**69.8**
LongLLaMA	4	18	91.4	87.1	59.1	41.0	64.5	68.7
MemLong	4	18	91.0	89.6	61.7	43.5	69.4	**71.0**
OpenLLaMA	20	N/A	93.6	91.2	55.4	38.2	66.4	69.0
w./ Rag	20	18	92.2	91.3	75.8	39.8	57.6	71.3
LongLLaMA	20	18	**94.1**	90.8	64.2	41.4	**72.1**	72.7
MemLong	20	18	93.5	**93.8**	**65.8**	**43.3**	70.6	**73.4**

Table 2: Accuracy [%] of 4-shot and 20-shot ICL on 5 NLU tasks(SST-2,MR,Subj,SST-5,MPQA).We compare the MemLong with both vanilla model (OpenLLaMA) and memory-augment model (LongLLaMA). Across a diverse range of experimental settings, our method consistently show competitive performance.

icantly lower training parameters (200M V. S. 0.3B) and fine-tuning data volume (0.5B V. S. 5B) compared to LongLLaMA. This underscores our model's efficiency in leveraging an external retriever for information acquisition, demonstrating a superior ability to synthesize and utilize knowledge effectively with substantially fewer resources.

5 Ablation Study

5.1 Training Setting

During the training phase, we explore the effects of varying retrieval layers on the model and examine whether the distribution shift problem, as discussed in MemTrm (Wu et al., 2022), could be adequately resolved by our approach. As mentioned before, Our method proposes a low-cost solution for distribution shifts. As shown in Figure 4, the brown line (the line at the top of the picture; the training method is similar to MemTrm fine-tuning all parameters of the model and all layers after the *memory layer* are involved in the retrieval) is significantly worse than all other ours methods (even the most unreasonable settings) in terms of performance and fitting speed. We will analyze the performance of the reasoning stage later.

5.2 Inference Performance

Q1: Does the memory length affect the performance of the model ? As depicted in Figure 5, our examination of the same model's performance across various memory sizes demonstrates a clear correlation between memory capacity and model efficiency. The trend indicates that incremental increases in memory size yield gradual enhancements in performance. Moreover, a critical threshold is identified at a memory size of 65536, beyond which the model's capabilities undergo a substantial leap. This suggests that while expanding memory offers substantial benefits, there is a practical ceiling to its effectiveness, likely influenced by the nuances of the data's distribution.

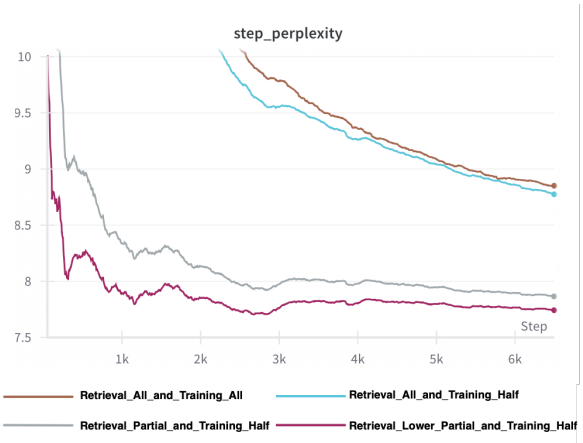

Figure 4: Degree of PPL during the training phase. The indicator for the y-axis is PPL. We mainly focus on training params and retrieval layers. We provide the specific parameter settings of each line in appendix A.

Q2: How many layers do we need to introduce extra memory information? As shown in Figure 4, (the pink line) and Table 3 (RPL+TH), the model performs best when the number of retrieval layers is set to [13,17,21,25]. It is empirically believed that if retrieval information is introduced into all upper layers of the model, it leads to a decrease in the model's attention to local context. Therefore, selecting retrieval layers at appropriate intervals can actually enhance the model's capabilities.

Method	PG19			Proof-pile		
	2k	4k	8k	2k	4k	8k
MemLong*	10.09	$> 10^3$	-	3.18	$> 10^3$	-
w. RA + TA	11.43	11.40	10.65	3.51	3.26	3.14
w. RA + TH	10.57	10.48	10.36	3.30	3.26	3.15
w. RP + TH	10.28	10.15	10.12	3.21	3.13	3.08
w. RLP + TH	**9.85**	9.83	**9.80**	**3.15**	**3.11**	**3.04**

Table 3: Different retrieval layers can affect MemLong's performance. MemLong marked with * means evaluating without Memory. The size of all methods using Memory is set to 32768. RA means retrieval across all upper layers; TA means training all params without freeze; RP means retrieval across fewer upper layers; RPL means retrieval acorss much fewer upper layers.

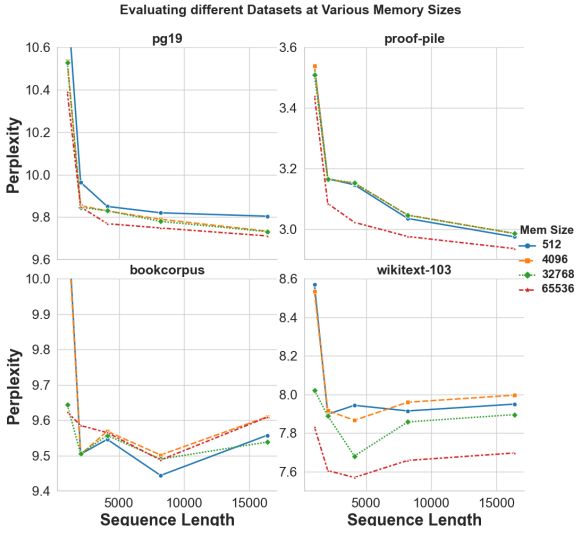

Figure 5: Evaluating different datasets at various memory sizes.In each subplot, all parameters are the same except for the memory size.

6 Related Work

6.1 Long Context Language Modeling

Long context Language Modeling mainly concentrate on length extension and context window expansion. Length Extension studies typically target the popular RoPE encoding, aiming to scale unseen PE into the space of positions seen during pre-training. These works (Su et al., 2024; Press et al., 2021; Chen et al., 2023a; Peng et al., 2023) enable the model to generalize to unseen positional encodings during inference, thereby achieving extrapolation beyond the lengths encountered during training. In contrast, our method does not require modifying the PE, and only use one addition module to extend the context. Context Window Extension focuses on how to extend the context window that LLMs can handle the input at one time. Due to the quadratic time and space complexity of computing attention, extending the input length of language models is quite challenging. Sparse attention (Kitaev et al., 2020; Chen et al., 2023b; Tworkowski et al., 2024; Bertsch et al., 2024; Beltagy et al., 2020) techniques have made significant strides, but our focus is on improving long-range language modeling by enabling LLMs to access relevant information at shorter input lengths via a retrieval-enhanced method.

6.2 Retrieval-Augmented Language Modeling

Much effort has been made to enhance Retrieval-Augmented Language Modeling (Lewis et al., 2020; Izacard and Grave, 2020; Ram et al., 2023; Yu et al., 2022; Asai et al., 2023). While some approaches use external retrievers, non-parametric information fusion often falls short compared to parametric methods within the model. We concentrate on integrating retrieval concepts directly into the model. REALM (Guu et al., 2020) suggests that relying solely on internal model knowledge is inefficient and advocates for the model to learn to retrieve and comprehend. kNN-LM (Khandelwal et al., 2019) enhances language modeling by blending the LLM's next-word predictions with those from a retrieval-based mechanism. MemTrm (Wu et al., 2022) introduces a memory bank but risks shifting memory distributions due to parameter adjustments. LongMEM (Wang et al., 2024b) mitigates this by training a sub-network, though this adds significant overhead. In contrast, our approach involves a fixed pre-trained model, enhancing it with a frozen retriever that aligns with the model's internal retrieval processes, thus avoiding distribution shifts and architectural changes.

7 Conclusion

We introduce MemLong, an innovative approach that significantly enhances the capability of language models to process long texts by leveraging an external retriever. MemLong utilizes a proficient retriever to swiftly and accurately access text relevant to the distant context with minimal memory overhead. MemLong successfully expands the model's context window from 2k to 80k tokens. We demonstrate that MemLong exhibits considerable competitive advantages in long-distance text modeling and comprehension tasks. MemLong can achieve up to a 10.4 percentage point improvement in performance compared to the full-context model.

Limitations

Our work primarily focuses on OpenLLaMA-3B. We hope that future research will explore and investigate the application of our methods to models of various sizes. At the same time, it has been found that while single-layer K-V Pairs can provide additional semantic information to the upper layers, this information is unstable. We hope that future work can provide a more rational framework to accommodate our methods. At the same time, we employ a retriever with fixed FlagEmbeddings (Xiao et al., 2023b; Zhang et al., 2023a), but studying a greater range of retrievers would be useful.

Ethics Statement

In the pursuit of advancing knowledge and developing innovative solutions, we are committed to upholding the highest ethical standards. Our work is guided by a steadfast dedication to integrity, transparency, and respect for all individuals and communities involved. Since pre-trained models may have some bias due to the unavoidable presence of harmful/offensive corpus during training, MemLong fine-tuning on Slimpajama will face this problem as well. Although solving this problem is out of our current work, we hope that there will be future work that addresses this type of problem well.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

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*.

Zhangir Azerbayev, Edward Ayers, and Bartosz Piotrowski. 2023. Proof-pile: A pre-training dataset of mathematical text.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. 2024. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. 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. 2023b. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

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*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv: Computation and Language,arXiv: Computation and Language*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. 2022. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys*, 55(8):1–35.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation

for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yi Lu, Xin Zhou, Wei He, Jun Zhao, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longheads: Multi-head attention is secretly a long context processor. *arXiv preprint arXiv:2402.10685*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Ohad Rubin and Jonathan Berant. 2023. Long-range language modeling with self-retrieval. *arXiv preprint arXiv:2306.13421*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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*.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2024. Focused transformer: Contrastive training for context scaling. *Advances in Neural Information Processing Systems*, 36.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jian Wang, Chak Tou Leong, Jiashuo Wang, Dongding Lin, Wenjie Li, and Xiao-Yong Wei. 2024a. Instruct once, chat consistently in multiple rounds: An efficient tuning framework for dialogue. *arXiv preprint arXiv:2402.06967*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2024b. Augmenting language models with long-term memory. *Advances in Neural Information Processing Systems*, 36.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023a. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023b. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

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

Haofei Yu, Yue Zhang, Wei Bi, et al. 2023. Trams: Training-free memory selection for long-range language modeling. *arXiv preprint arXiv:2310.15494*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023a. Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*.

Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023b. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Different Training Settings

As shown in 4, we list the variable values corresponding to different setting names in the ablation experiment.

Setting Name	Retreival Layers	Memory Layer	Training Params
Retreival_All_and_Training_All	[14,15,. . .,26]	13	All of Model's Trainable
Retreival_All_and_Training_Half	[14,15,. . .,26]	13	Half of Model's Trainable
Retreival_Partial_and_Training_Half	[14,16,18,. . .,26]	13	Half of Model's Trainable
Retreival_lower_Partial_and_Training_Half	[14,18,22,26]	13	Half of Model's Trainable

Table 4: The specific parameters of different setting names.