CAMELoT: Towards Large Language Models with Training-Free Consolidated Associative Memory

Anonymous ACL submission

Abstract

Large Language Models (LLMs) struggle to model long input sequences due to high memory and runtime costs. Memory-augmented models have emerged as a promising solution to this problem, but current methods are hindered by limited memory capacity and require costly re-training to integrate with a new LLM. In this work, we introduce an associative memory module which can be coupled to any pre-trained (frozen) attention-based LLM without re-training, enabling effective long language modeling. Unlike previous methods, our associative memory module consolidates representations of individual tokens into a non-parametric distribution model, dynamically managed by properly balancing the novelty and recency of the incoming data. By retrieving information from this consolidated associative memory, the base LLM can achieve significant (up to 29.7% on Arxiv) perplexity reduction in long-context language modeling compared to other baselines on various standard benchmarks. This architecture, which we call CAMELoT (**C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer[1]), demonstrates superior performance even with a tiny context window of 128 tokens.

1 Introduction

Humans are exposed to a myriad of events through their lives. The human brain effectively processes and consolidates events to form memories that exemplify related events and form the basis for future actions, by retaining essential information and discarding inessential details (Sara, 2000). Associative Memory (AM) is a key type of such human-like memory systems to store information, with a core computation to link (*associate*) a query with representations stored in the memory banks (Willshaw et al., 1969; Hopfield, 1982). Specifically, for any given query, AM selects the consolidated memory slot that best matches the query. The representations in AM concisely summarize past experiences and provide valuable cues for future actions. Recently, there has been growing interest in designing associative memory networks (Krotov and Hopfield, 2016; Ramsauer et al., 2021). Other works investigate memory consolidation in neural networks with various local learning rules (Dudai, 2004), which are computationally cheaper than the traditional end-to-end back-propagation for neural networks (Tyulmankov et al., 2021).

[1]We will release our codes upon publication to facilitate future research.

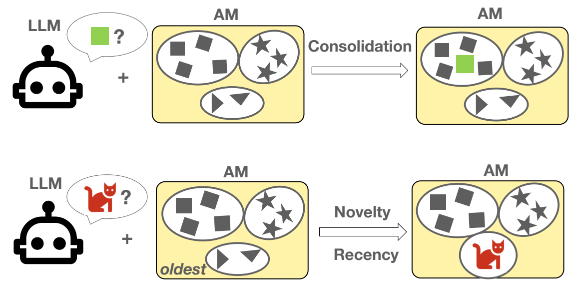


Figure 1: **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer (CAMELoT). Top: Consolidation of representations in the associative memory (AM) – related concepts are grouped together and averaged. Bottom: Recency-dependent incorporation of novel concepts – when a new concept is introduced with no close matches, the oldest slot (since its last update) is replaced with the new concept.

Concurrently, large language models (LLMs) have demonstrated their potential in various practical NLP applications such as chatbots (OpenAI, 2024), text summarization (Radford et al., 2019), and question answering (Chung et al., 2022), etc. A key parameter for LLMs is the input context length L that the models are trained with. Supporting longer context makes it possible to incorporate richer information and increase LLM performance at inference time (Press et al., 2022). However, the attention mechanism of a pre-trained LLM usually

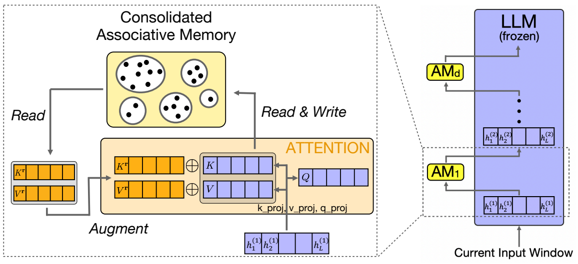


Figure 2: The general pipeline of our method. Every layer of the backbone LLM is augmented with an AM module (we draw AM in the first attention layer here, just as an example). Keys and values are calculated for every token, keys are used to search for relevant memorized tokens in the memory bank and return them (Read). The retrieved memory keys and values are prepended to the original token keys and values as prefixes. Finally, the attention operation is applied on the concatenation of the retrieved and native keys and values (Augment). After retrieval, the memory state is modified according to the Write operation, see Figure 3.

scales quadratically (L^2) with an increasing number of tokens, which makes increasing the context length computationally challenging due to substantial requirements for resources.

These constraints raise a question: *can we develop a plug-and-play module for pre-trained (frozen) LLMs to handle longer contexts beyond L after the training*? Importantly, this module should be computationally efficient and should *not* require any retraining or fine-tuning of the backbone LLM.

In this work, we draw inspiration from the human memorization mechanism and tackle this question using Associative Memory (AM). We propose a plug-and-play AM module that consolidates individual tokens based on the novelty and recency of the input text (as shown in Figure 1). The consolidated text is modeled as non-parametric distributions, with one distribution per key-space for each LLM layer. When processing a long text, the modes of these distributions are dynamically updated as the context window sweeps over time, with new modes created for novel information and outdated ones replaced. As a result, the module consolidates information about the prior context far beyond the current context window (of length L). The module then retrieves the modes closest to the current input and operates the attention computation on them. This module can be integrated with any pre-trained attention-based LLM, extending its context window far beyond L by approximating a full-context attention over all past information.

Our method does not require any re-training, fine-tuning, or learning adaptors between the base LLM and the AM module. We conduct comprehensive experiments on long-context language modeling tasks, demonstrating that this human-like memory design leads to significantly stronger results compared to baselines. For instance, when coupled to a pre-trained LLaMA2, our memory-enhanced network results in significant (up to 29.7% on Arxiv) perplexity reduction in long-context modeling compared to the base LLM.

2 Related Work

Memory networks. There is a large body of literature on memory models, e.g. memory networks (Weston et al., 2014), sparse distributed memory (Kanerva, 1988), and associative memory (Kohonen, 2012). Neuroscience-inspired memory models have also been used for language model augmentation (Park and Bak, 2023). Memory augmentation has shown its effectiveness in reinforcement learning (Graves et al., 2016) and recurrent neural networks (Graves et al., 2014). To the best of our knowledge, none of these works enable memory augmentation of LLMs without additional training, as in our approach.

Long Context Modeling. Several streams of work aim to enhance the long context capability of LLMs. **Long-range self-attention techniques** have been proposed to improve the efficiency of transformer models, including low-rank factorization (Wang et al., 2020), local attention (Ramachandran et al., 2019), dilated attention (Ding et al., 2023), sparsity (Beltagy et al., 2020; Zaheer et al., 2020; Kitaev et al., 2020), and hardware-aware attention mechanisms such as FlashAttention (Dao et al., 2022; Dao, 2023). Despite notable progress, these methods struggle to retrieve information in the middle of the input (Liu et al., 2023). They can also be used in tandem with our proposed approach for longer context modeling.

Another line of work utilizes **state-space models** to handle long-range dependencies in sequential data. Mamba (Gu and Dao, 2023), a seminal work in this area, captures long-term dependencies with a selective state-space model, achieving linear training time without the quadratic scaling of traditional attention mechanisms. Jamba (Lieber et al., 2024) improves on this by combining Transformer layers with Mamba layers to train effectively on long contexts. However, these models require training on long sequences. While combining AM with state-space models is an interesting future direction, our focus in this work is on enhancing the long-context capability of a Transformer-based LLM after its training.

Memory-augmented language models have emerged as a promising approach (Packer et al., 2023; Dai et al., 2019; Wu et al., 2022; Tworkowski et al., 2023; Weston et al., 2014). In particular, Wu et al. (2022) show that a kNN lookup into a memory cache bank containing (key, value) pairs of individual past inputs can improve language modeling. Tworkowski et al. (2023) further improved this approach using contrastive learning. In the same vein, Wang et al. (2023) addressed the memory staleness limitation of these works by training a side network model, while keeping the LLM frozen. Unlike these methods, our approach relies on consolidated representations of past tokens which are dynamically updated, therefore getting rid of the limitation of the number of memory slots. Moreover, different from these approaches, our method is *training-free* (memory updates occur solely at runtime), making it easier to integrate our memory module into any existing LLM architecture.

Prompt compression research (Ge et al., 2023; Mu et al., 2023; Chevalier et al., 2023) has also been explored recently to extend the context length in transformer models. These methods operate at the input level, while our method consolidates the internal representations of the model based on a local associative memory update rule. Rae et al. (2019b) proposed the Compressive Transformer, which compresses past activations of the model for long-range sequence modeling. In contrast, our proposed approach does not require training or additional losses like attention-reconstruction. In addition, we offer a novel way to effectively update our associative memory representations, balancing information about novelty and temporal proximity.

3 Associative Memory Enabled LLM

For long document modeling tasks, it is desirable to have an architecture capable of efficient usage of information that appeared in past contexts. Our proposed method is built on three desiderata. First, redundant information from the past should be compressed and stored in the AM block while reducing repetitions (**consolidation**). When the same concept appears in the past context multiple times, it is wasteful to store each individual instance of that concept in a separate memory slot; instead, all those instances should be consolidated and stored only once. Second, novel concepts not encountered by the model in the past must be detected and stored in a new memory slot at their first encounter (**novelty**). These novel memory slots can be subsequently consolidated with the possible future occurrences of related concepts. Third, in situations when the topic shifts, the model should be able to discard outdated memory slots that are no longer useful, if that is required for the incorporation of additional novel concepts encountered following the topic shift (**recency**).

To achieve these desiderata, we design CAMELoT, a **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer, consisting of a base language model and a memory module (overall architecture shown in Figure 2). The memory module is equipped with a **Read** and **Write** operation, supporting information retrieval from the memory bank and the update to the memory bank. With the retrieved information, the current context window of LLM is memory-enhanced via the **Augment** operation. These three desiderata are the foundations of CAMELoT. Our method is agnostic to the specific choice of many popular transformer architectures, in the sense that any attention-based LLM can be enhanced with the AM in CAMELoT.

3.1 Read Operation

When a context window of length L is processed through the LLM, keys and values from every layer (more generally can be an arbitrary subset of layers) are passed to the corresponding AM module (one per memory-augmented layer). AM in each layer consists of M memory slots, enumerated by the index $\mu = 1, ..., M$. Each slot contains two vector variables: memory keys $K_\mu^{\mathbf{mem}}$ and memory values $V_\mu^{\mathbf{mem}}$, and two integer scalar variables: counts c_μ (number of consolidated instances), and age τ_μ (how old the current slot is since its last update).

When a set of keys K_i and values V_i (index $i = 1, ..., L$ enumerates individual tokens from the current context window) is passed to the AM module to retrieve relevant information, a *search function* identifies the memory slots with the strongest association (i.e., highest similarity) between the input token key K_i and AM's memory slot keys $\{K_\mu^{\mathbf{mem}}\}$:

$$\hat{\mu}(i) = \underset{\mu}{argmax}\big[sim(K_\mu^{\mathbf{mem}}, K_i)\big] \qquad (1)$$

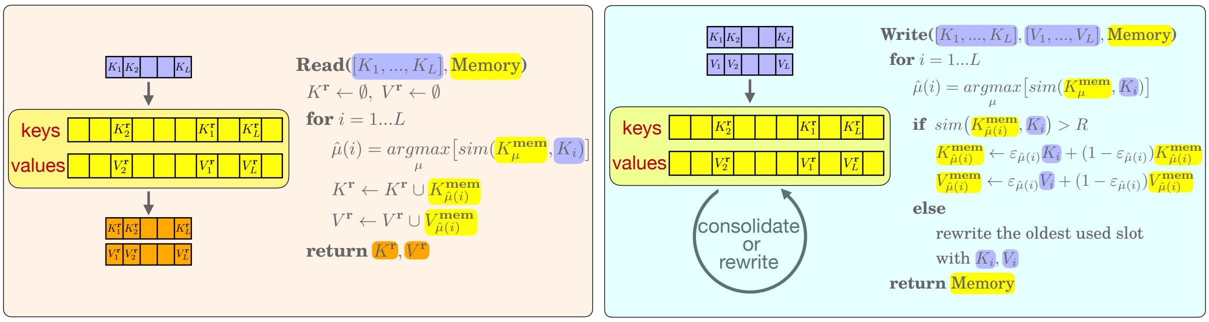


Figure 3: Every AM module performs read and write operations. The read operation retrieves memorized tokens most similar to the native keys. The write operation updates the state of the memory by performing consolidation, which depends on novelty and recency.

The keys and their corresponding values of these L strongest-associated memories ($K^{\mathbf{r}}$ and $V^{\mathbf{r}}$) are returned for the current L native tokens and passed back to the LLM in the form of the key-value cache. See details in Figure 3.

3.2 Augment Operation

The list of retrieved key-value caches ($K^{\mathbf{r}}$ and $V^{\mathbf{r}}$) are passed back to the base LLM and used as the prefix context in each respective memory-augmented layer. They are prepended to the LLM keys and values of current input tokens. Then causal attention is performed on the concatenated list, which after the augmentation contains $2L$ keys and values (the length of current native context + the length of retrieved memories) and L queries (current context only), resulting in the augmented transformer attention output $[a_1, \cdots, a_L]$. The attention output results in augmented hidden states $[h_1, \cdots, h_L]$ which are the inputs to the next layer, as shown in the following equations and Figure 3:

$$[a_1, \cdots, a_L] = Attn(Q, K', V') \tag{2}$$

$$Q = [Q_1, Q_2, \cdots, Q_L] \tag{3}$$

$$K' = K^{\mathbf{r}} \oplus [K_1, \cdot, K_L], \tag{4}$$

$$V' = V^{\mathbf{r}} \oplus [V_1, \cdots, V_L] \tag{5}$$

3.3 Write Operation

The state of AM is updated by the current context window according to the **Write** operation comprised of two parts explained next (see Figure 3 for an illustration).

Consolidation. If the similarity between the current context token key and the strongest-associated memorized key is large (i.e., $> R$, R is a hyper-parameter), the concept described by that token is declared familiar and, for this reason, its key and value are consolidated with the key and value stored in that memory slot. Specifically, memory slots are updated according to:

$$K_{\hat{\mu}(i)}^{\mathbf{mem}} \leftarrow \frac{K_i + c_{\hat{\mu}(i)} K_{\hat{\mu}(i)}^{\mathbf{mem}}}{c_{\hat{\mu}(i)} + 1} \tag{6}$$

$$V_{\hat{\mu}(i)}^{\mathbf{mem}} \leftarrow \frac{V_i + c_{\hat{\mu}(i)} V_{\hat{\mu}(i)}^{\mathbf{mem}}}{c_{\hat{\mu}(i)} + 1} \tag{7}$$

$$c_{\hat{\mu}(i)} \leftarrow c_{\hat{\mu}(i)} + 1 \tag{8}$$

where c_μ tracks the number of instances consolidated in slot μ. Thus, the consolidated representations stored in each slot μ are always arithmetic averages of individual instances that went into that slot. By introducing an update rate $\varepsilon_\mu = 1/(c_\mu + 1)$, these expressions can be rewritten as incremental modifications to the existing representations stored in the AM, as in Figure 3.

Novelty and Recency. If the similarity with the closest memorized key is weak (i.e., $< R$), the concept is declared novel. In this case, the oldest unused memory slot (the one with maximal age τ_μ) is replaced with K_i, V_i, and its age is set to 0. After each slot $\hat{\mu}(i)$ update, its age statistic $\tau_{\hat{\mu}(i)}$ is set to 0. The ages of all slots that had no matching current context hidden state are incremented by 1. We also provide a probabilistic interpretation for CAMELoT in Section 9.1

4 Experiments

We evaluate CAMELoT on causal language modeling task. We follow the data preprocessing method proposed by Dai et al. (2019) where lengthy documents are segmented into sequential, non-overlapping windows and the LLM processes each window one by one. During this process, we first use the key and value representations of each token to read from the AM and retrieve the relative information, then augment the causal language modeling step by treating the returned memory as the past caches. Subsequently, the keys and values of the current input are integrated into the AM via the Write function. We measure the perplexity for tokens in each window and calculate their average across the entire long context in the end.

Details. We take the officially released LLaMa2-7b from Huggingface Library as the base model in CAMELOT. We put memory banks into a single NVIDIA-A100 GPU for fast parallel computation. We also notice that one can use FAISS (Johnson et al., 2017) approximate search, which is a simple extension of our framework. After hyper-parameter studies (Section 9.2 and Section 9.3), we use cosine similarity in CAMELOT and the similarity threshold R in novelty detection is set to be 0.93. Unless specified otherwise, our experimental results are reported for CAMELOT with 10k memory slots (more experiments on memory size are detailed in Section 9.3.3). For more implementation details, please refer to Section 9.2.

4.1 Evaluation Setups

Datasets We evaluate the long context language modeling capabilities of CAMELOT using three benchmarks:

- **Wiki-103** (Merity et al., 2016)[2], which comprises articles from Wikipedia covering various topics with good language quality;
- **Arxiv** (Gao et al., 2020)[3], a collection of academic papers primarily in the fields of Mathematics, Computer Science, and Physics. This dataset is recognized for its high-quality text and mathematical content, making it a challenging benchmark for long-context language modeling;
- **PG-19** (Rae et al., 2019a)[4] which includes full-length books, offering a standard benchmark widely used in long-range natural language modeling (Wu et al., 2022; Wang et al., 2023; Tworkowski et al., 2023).

We take the test split of each dataset and report its language modeling perplexity.

[2]https://blog.salesforceairesearch.com/the-wikitext-long-term-dependency-language-modeling-dataset/
[3]Taken from the Pile: https://pile.eleuther.ai/
[4]https://github.com/google-deepmind/pg19

Baselines. We compare CAMELOT against two notable memory-augmented transformers that have demonstrated effectiveness in long language modeling tasks:

- **Transformer-XL** (Dai et al., 2019): This model uses a finetuning-based approach, storing a fixed length of previous input in a cache to enhance the current input. Notably, it does not employ similarity-based retrieval.
- **Memorizing Transformer** (Wu et al., 2022): this model saves past caches in a circular manner. Thus older caches are replaced by newer ones as the memory bank fills up (no consolidation occurs) and similar caches are retrieved for input augmentation. The official implementation relied on fine-tuning.

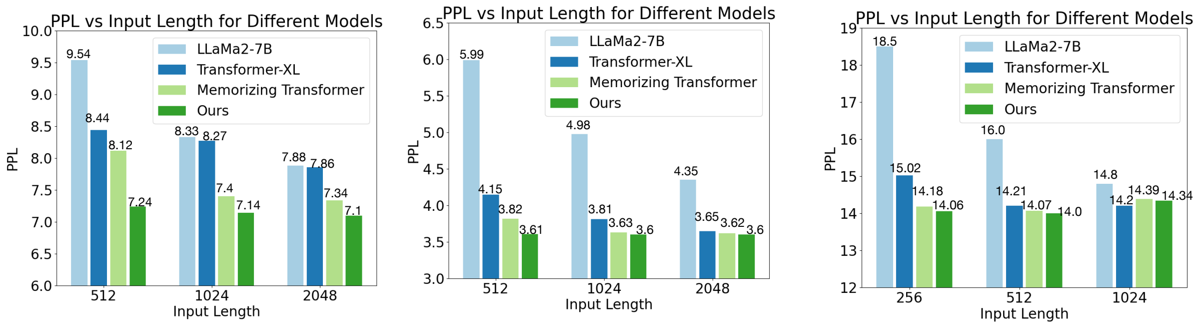
For a fair comparison, in CAMELOT and the baselines experiments, we used the same LLaMa2-7B backbone (original baselines used weaker backbones, such as GPT2), and did not use fine-tuning.

Ablations To assess the impact of each component within CAMELOT, we define the following ablation variants:

- **CAMELOT w/o Read:** Instead of retrieving the closest matching memory concept for each token in the current input, a random memory concept is returned.
- **CAMELOT w/o Recency:** If a token's mode has no close match in memory, it randomly replaces a memory slot rather than the outdated one, ignoring recency.
- **CAMELOT w/o Novelty.** Tokens are consolidated into their closet slot, regardless of if they are from novel modes. R=-1 in cosine similarity retrieval.
- **CAMELOT w/o Consolidating.** Memory gets updated by token representations based on temporal recency, without consolidating, setting R=+1.

4.2 Results

Figure 4 compares CAMELOT with the baseline models. While memory-augmented methods generally improve upon the base model on test perplexity, our analysis uncovers the following observations in their effectiveness. Transformer-XL shows the



((a)) PG-19 ((b)) Arxiv ((c)) Wikitext103

Figure 4: Language Modeling Perplexity (PPL) on wikitext-103, Arxiv, and Pg-19. For wikitext-103, we notice the maximum length of its documents is smaller than 2k. Therefore, we report results using models whose effective input length ≤ 2048. A lower PPL indicates better performance.

Models	**PPL**
LLaMa2-7B	7.30
CAMELOT	**6.85**
CAMELOT w/o Read	> 20
CAMELOT w/o Recency	9.25
CAMELOT w/o Novelty	7.23
CAMELOT w/o Consolidation	7.00

Table 1: Ablation Study on PG-19-sampled. We report the relative performance lost in perplexity (PPL) over the full CAMELOT.

least improvement, hindered by the lack of relevance assessment during memory augmentation. The Memorizing Transformer, with its capability to selectively retrieve relevant information from the past, outperforms Transformer-XL. However, it lacks memory consolidation, meaning it can only hold a finite cache before older memories are overwritten, limiting its long-term utility.

By not only selecting relevant past information but also employing a novel memory consolidation process, CAMELOT significantly enhances model performance (16.6% on PG-19, and 29.7% on Arxiv, and 13.14% on Wikitext-103, relative the base model on average), surpassing other memory-augmented methods. Remarkably, CAMELOT achieves superior performance at shorter input lengths, demonstrating its handling of long-range dependency regardless of input size. For further discussion please see Section 5.1.

4.3 Ablation Studies

We evaluate on PG-19 sampled dataset, a subset of PG-19 comprising 20% of the books in test set. We report test perplexity for each variant with a context length of 2048.

Results shown in Table 1 reveal that CAMELOT w/o Read performs significantly worse compared with full model, emphasizing the crucial role of Read function in ensuring semantic relevance. When a random cache is returned in this variant, it might provide limited or even harmful information for current modeling. CAMELOT w/o Recency also shows a notable performance dip over the full CAMELOT model, confirming the essential role of maintaining the proper recency in the memory. Variations in token consolidation and replacement also impact performance, resulting in different performance drops compared to the full approach. A larger decrement can be expected if the memory size gets smaller or the modeling corpus gets longer. These findings suggest CAMELOT's optimal performance relies on the combination of relevance, recency, novelty, and effective consolidation. Please refer to Section 9.3 to see more discussions on ablation study.

5 Further Discussions

5.1 CAMELOT Reaches SOTA Earlier

This section analyzes CAMELOT's performance with different input lengths on the PG-19 test set, using 10k memory slots. Results are shown in Figure 5.

Unlike models without memory augmentation, CAMELOT demonstrates a relatively consistent performance across different input lengths. This stability can be attributed to the integration of additional knowledge in the AM saved from previous inputs. As CAMELOT accumulates past information, its visible context range extends beyond the

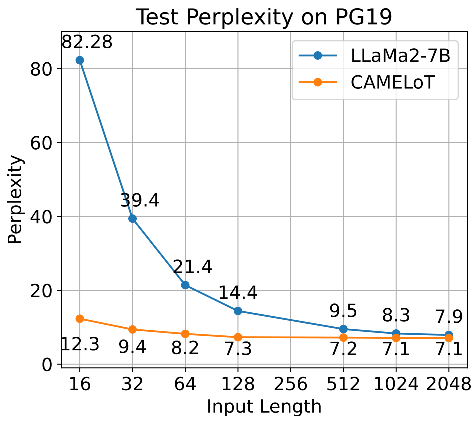


Figure 5: Test perplexity on PG19 with different input lengths.

current input, allowing an effective modeling of long-range dependencies irrespective of the length of the current input. In contrast, the model lacking memory augmentation relies solely on the local context of the current input, leading to performance fluctuations based on input length.

CAMELoT maintains its effectiveness even with tiny input lengths (e.g., 128, 64, 32), reducing the demand on hardware resources such as large GPUs. This enables transformers to operate the attention mechanism with shorter inputs but without compromising the quality of language modeling. Such an advantage lowers the barriers for deploying large language models in environments where computational budget is limited.

5.2 Efficiency Analysis

Here, we provide a detailed comparison of the efficiency of CAMELoT with the base LLM. Consider the results on PG-19 from Figure 5. The base LLM achieves a state-of-the-art (SOTA) perplexity (PPL) of 7.9 at $L = 2048$, while CAMELoT achieves a lower PPL of 7.3 at $L = 128$. With the same number of heads and token dimensions (which is assumed to be the same as the hidden dimension in the product of keys and queries) in transformer, CAMELoT achieves approximately $C_{\text{base}}(L = 2048)/C_{\text{CAMELoT}}(L = 128, M = 10000) \approx 6.2$ reduction in compute cost associated with attention and memory search compared to the base LLaMa (see Section 10 for details). Furthermore, our proposed approach can be made even more efficient by utilizing sublinear similarity search methods (Douze et al., 2024).

	Frequency			
	>10K	1K - 10K	100 - 1K	<100
LLaMa2-7B	2.75	5.08	9.77	25.96
CAMELoT	2.13	4.11	7.54	19.4
Relative Gain Over LLaMa2	22.1%	19.3%	22.8%	25.3%

Table 2: Test perplexity broken down by word frequency buckets.

5.3 CAMELoT Models Infrequent Word Better

In this section, we answer the question: which words benefit from long-term knowledge in CAMELoT? Following (Rae et al., 2019b), we categorize test tokens based on their frequency in the training set. The tokens are grouped into different frequency buckets, and we calculate the average perplexity for each group.

Results are shown in Table 2. All tokens gain at least 19% improvements over the base model. Among them, the high frequency tokens (frequency $>$ 10k), which constitute the majority of the test set, exhibit a 22.1% improvement. The largest improvement (25.3%) is observed in the group of rare tokens. This improvement suggests augmenting language models with mechanisms like CAMELoT can be a viable approach to better address the challenges associated with rare token modeling.

5.4 Visualization: What is Stored in AM?

This section visualizes the contents in the AM's memory slots, to provide insights into memory usage dynamics. Table 3 displays the updates of six slots by processing input tokens over time.

First, we identified two key types of memory slots in CAMELoT: 1. *Functional Slots* that capture lexical, syntactical, or grammatical aspects of tokens, as seen in the top rows of Table 3, related to modeling language structures and rules; and 2. *Semantic Slots* in the bottom rows which capture the semantic essence of inputs. Tokens are assigned to slots based on their functional or semantic relevance, aligning with previous findings that embeddings from different layers or attention heads in Transformer-based models can specialize in different language aspects (Vig et al., 2019). Each slot has consistent modeling rules. For instance, despite the similar functional purposes, prefix and suffix tokens are allocated to separate slots. This indicates that CAMELoT can detect subtle nuances.

Slot97: Pronouns	**Slot110: Prefix**	**Slot103: Suffix**
this, he, she, her, I, our, were had, They, their, they, was that, is, are, those, there, these	pre, Re, alt, al, be, bel, del, comp, Al, per,ple, ab, dis, no, non, de, un, im, bl, bri, Ch, Eng, com, fl, Fr, sal, gen, str, et, es	atives, ful, ate, ere, ish, ible, ily, ry, ly, ling, ent, ence, er, ine, ina, ier, age, ations, ation, ood, inity, itute
Slot60: States to Civilization	**Slot394: Masculine to Feminine**	**Slot7275: Num. to Adv.**
Minn, Miss, states, Tennessee, PA , Minnesota, Lincoln, Pitts, Kingdom, Phil , Si, prep, DEL, Eng, Montreal , British, Franklin, Hill, Rep, Nation , country, county, government, civil	boys, editor, Dr, Jack, men, Judge, him , work, Chair, politics, religion, justice, brave, Scott, business, manager , secret, ary, she, mother , love, house, hand, dress, Virgin	six, four, two, hundred, fifty, thousand, many, few, several, every, another, anything, enough majority, ton, massive great, remarkable, generally

Table 3: Visualization of the memory updating. We take the AM linked to word embedding layer and log the memory assignment of each token on PG-19. As discussed in Section 5.4, the original concept, which is written into memory earlier (we show them in red), can shift slightly to a related new one (colored in yellow) during the consolidation, caused by transition words (colored in orange) or polysemous words (colored in green).

Additionally, concept shifts within slots occur during consolidation. Examples in the bottom row of Table 3 show transitioning from *federations* to *civilizations* in slot 60, from *masculine* to *feminine* terms in slot 394, and from specific *numbers* to *quantitative adverbs* in slot 7275. These changes arise from context-dependent updates and the semantic diversity of words, where transitional tokens like "*business, manager, secret, ary*" and polysemous words like "*great*" influence the shifts in slot focus. We view this concept transitions as beneficial, as they facilitate an efficient consolidation while preserving recency. If these cumulative transitions lead to a significant change in the slot's mode, the slot can be replaced by the new token in the future rounds, as part of the novelty mechanism.

6 Conclusion

We introduce CAMELoT, a **C**onsolidated **A**ssociative **M**emory **E**nhanced **Lo**ng **T**ransformer, to handle long dependency modeling without the need for training. CAMELoT has a model-agnostic design, allowing seamless integration into different language models. Experimental results prove its effectiveness, with the long-context language modeling perplexity significantly reduced (by up to 29.7%), and superior performance is consistently obtained even with a tiny input window of 128 tokens or less. Future research directions connecting AM and LLMs involve improving the AM design (e.g., automatically learning a Write function) and tackling other long context modeling tasks (e.g., long document question answering).

7 Ethical Considerations

In this paper, we design a consolidated AM module to store tokens in long contexts (e.g., long documents). However, a malicious user could use this module to store personal information when processing human data, posing a risk to user privacy. We argue that LLMs should be audited rather than used as a 'black box' when handling human data, to protect privacy and prevent harmful usages.

8 Limitations

One limitation of this work is that we only experiment with LLaMa2-7B, a model commonly used in current research projects. However, our method is model-agnostic and can be easily adapted to any attention-based transformers. In future work, we plan to address this limitation by incorporating a broader range of language models with different model architecture such as state space models.

Additionally, our focus is specifically on casual language modeling performance with long contexts, as it is a fundamental task in LLM training and usage. We acknowledge that there are other long-context tasks, such as multi-document question answering and reasoning that could be analyzed. This work aims to provide a novel perspective by enhancing pre-trained LLMs with neuroscience-inspired Associative Memory, offering an initial interdisciplinary exploration of long context modeling. In the future, we plan to test our method on a wider range of downstream tasks.

References

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

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

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*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Yadin Dudai. 2004. The neurobiology of consolidations, or, how stable is the engram? *Annu. Rev. Psychol.*, 55:51–86.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Tao Ge, Jing Hu, 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*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

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

John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *Preprint*, arXiv:1702.08734.

Pentti Kanerva. 1988. *Sparse distributed memory*. MIT press.

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

Teuvo Kohonen. 2012. *Associative memory: A system-theoretical approach*, volume 17. Springer Science & Business Media.

Dmitry Krotov and John J Hopfield. 2016. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

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

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

OpenAI. 2024. Chatgpt. `https://www.openai.com/`. June 15 version.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Sangjun Park and JinYeong Bak. 2023. Memoria: Hebbian memory architecture for human-like sequential processing. *arXiv preprint arXiv:2310.03052*.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation. *Preprint*, arXiv:2108.12409.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019b. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. 2019. Stand-alone self-attention in vision models. *Advances in neural information processing systems*, 32.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Thomas Adler, David Kreil, Michael K Kopp, et al. 2021. Hopfield networks is all you need. In *International Conference on Learning Representations*.

Susan J Sara. 2000. Retrieval and reconsolidation: toward a neurobiology of remembering. *Learning & memory*, 7(2):73–84.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling. *arXiv preprint arXiv:2307.03170*.

Danil Tyulmankov, Ching Fang, Annapurna Vadaparty, and Guangyu Robert Yang. 2021. Biological learning in key-value memory networks. *Advances in Neural Information Processing Systems*, 34:22247–22258.

Jesse Vig, Machine Learning, and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *ACL 2019*, page 63.

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. 2023. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins. 1969. Non-holographic associative memory. *Nature*, 222(5197):960–962.

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

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

9 Appendix

9.1 CAMELoT: Probabilistic Interpretation

The keys and values in AM slots can be viewed as modes of a non-parametric Gaussian mixture distribution estimation approximating the key-value manifold of the past context windows. This mixture accepts new key-value points from the current context via a diagonal kernel of width $\hat{R}$ (distance measure corresponding to similarity R). The means (centers) of the modes of the mixture are updated according to our online average rules while maintaining the needed sufficient statistics (counts) for computing the averages in further updates. Retrieving nearest distribution modes to the current context hidden states effectively approximates the full (long) context attention, at least within the radius $\hat{R}$ from the retrieved mode centers. For the tokens whose keys and values are beyond radius $\hat{R}$ of their closest mode, new modes are created online, while the oldest modes are evicted, maintaining the recency of our distribution estimation and its correspondence with the evolving context.

9.2 Experiment Details

Environments All transformers-based language models are implemented based on the HuggingFace[5] libraries (version 4.34.0) or the officially released Github Repos. All codes are implemented with Python 3.10.12 and PyTorch 2.2.0 with CUDA 12.1.0. We run experiments with 2 NVIDIA A100 GPUs, one for language model inference and one for hosting the memory banks. Each has memory of 80GB.

Hyper-parameters In CLM tasks, we set batch size to be 4. For the similarity hyper-parameter R, we conduct a hyper-parameter study on wikitext-103 and use $R = 0.93$ for all experiments. We show the study in the next section.

9.3 More Ablation Studies

9.3.1 Ablation: Different Choices of R

We conduct hyper-parameter study for similarity threshold R on a subset of Wikitext-103 validation set, in which the examples are randomly sampled. We take LLaMa2 models with input length to be 128. The results are shown in Table 4.

From the results, we notice the best R is 0.93. Therefore we use 0.93 in our experiments.

[5] https://huggingface.co/models

R	Perplexity
R=0.1	PPL=18.72
R=0.2	PPL=18.71
R=0.3	PPL=18.71
R=0.4	PPL=18.69
R=0.5	PPL=18.67
R=0.6	PPL=18.46
R=0.7	PPL=17.35
R=0.8	PPL=15.30
R=0.9	PPL=14.38
R=0.93	PPL=**14.30**
R=0.95	PPL=14.90

Table 4: Hyper-parameter study for R on the validation subset of Wikitext-103

	Perplexity
CAMELoT + Cosine Similarity	**16.96**
CAMELoT + Euclidean Similarity	17.45

Table 5: Analysis of similarity function on a subset of wikitext-103 validation set.

9.3.2 Ablation: Different Choices of Similarity Function in Read Operation

We conduct an ablation study on the similarity function in Read operation. Similarly, we randomly sampled a subset data from the validation set of Wikitext-103. We conduct evaluation experiments with cosine similarity and euclidean similarity. We use input window with 128 tokens. Note in this experiment we use LLaMa1-7B. The results are shown in Table 5. We notice cosine similarity gives the best performance and we use cosine similarity in our other experiments.

9.3.3 Ablation: Different Memory Sizes

Model	Input Context	Memory Size	Perplexity
LLaMa2-7B	512	None	9.84
	2048	None	7.88
CAMELoT	512	4096	7.42
	2048	4096	7.22
	512	10k	7.24
	2048	10k	**7.10**

Table 6: Language Modeling performance on PG19 with different sizes of memory banks and different input lengths.

In this section, we analyze how the size of the memory bank affects CAMELoT. We compare its performance on the PG-19 dataset using two configurations: one with 4,096 memory slots and another with 10k slots. The findings are presented in Table 6.

With each memory slot designed to hold a unique mode of information, increasing the number of slots allows CAMELOT to capture a wider range of knowledge. As a result, the version with 10k slots outperforms, showing a notable improvement in test perplexity – 26.4% for inputs of 512 length and 9.9% for 2048 length relative to the base model.

However, the 4,096-slot configuration also performs strongly, with only slightly lower improvements (24.6% and 8.4%, for the same input lengths) than CAMELOT with 10k slots. This good performance demonstrates that the effectiveness of CAMELOT does not solely rely on the quantity of data modes it can hold in its memory, but also on how it manages and utilizes this data through mechanisms like consolidation and novelty. This balance ensures CAMELOT remains effective across various memory sizes and input lengths, maintaining stability and efficiency.

10 Theoretical Calculation for Computation Cost

Suppose we have memory size M in CAMELOT. We denote the number of heads in the backbone LLM as h, and token dimension per head is d. The computation cost for the multi-head self-attention of the base transformer with input length L is quadratic in the sequence length, linear in token dimension and the number of heads. Additionally, we include the factor of 2, which accounts for the computation of both the attention scores and multiplication of those by the value matrix. We retain only the terms that are dominant for large L. This gives:

$$C_{\text{base}}(L) = 2hL^2d \tag{9}$$

CAMELOT has the same self-attention cost, and the cross attention cost (retrieved tokens effectively double L). Additionally, there is a cost associated with the search through the memory, which is linear in M, L, h, and d.

$$C_{\text{CAMELOT}}(L, M) = 2hL(L + L)d + hdLM \tag{10}$$