CMT: A Memory Compression Method for Continual Knowledge Learning of Large Language Models

Dongfang Li, Zetian Sun, Xinshuo Hu, Baotian Hu, Min Zhang

Harbin Institute of Technology (Shenzhen)
crazyofapple@gmail.com

Abstract

Large Language Models (LLMs) need to adapt to the continuous changes in data, tasks, and user preferences. Due to their massive size and the high costs associated with training, LLMs are not suitable for frequent retraining. However, updates are necessary to keep them in sync with rapidly evolving human knowledge. To address these challenges, this paper proposes the **C**ompression **M**emory **T**raining (**CMT**) method, an efficient and effective online adaptation framework for LLMs that features robust knowledge retention capabilities. Inspired by human memory mechanisms, CMT compresses and extracts information from new documents to be stored in a memory bank. When answering to queries related to these new documents, the model aggregates these document memories from the memory bank to better answer user questions. The parameters of the LLM itself do not change during training and inference, reducing the risk of catastrophic forgetting. To enhance the encoding, retrieval, and aggregation of memory, we further propose three new general and flexible techniques, including memory-aware objective, self-matching and top-k aggregation. Extensive experiments conducted on three continual learning datasets (i.e., StreamingQA, SQuAD and ArchivalQA) demonstrate that the proposed method improves model adaptability and robustness across multiple base LLMs (e.g., +4.07 EM & +4.19 F1 in StreamingQA with `Llama-2-7b`).

Introduction

Large language models (LLMs) have become the core of natural language processing (NLP) (Touvron et al. 2023a; OpenAI 2023). The current challenge is how these LLMs adapt to rapidly changing world knowledge, especially in the context of increasing new data and growing model complexity (Shi et al. 2024). Typically, LLMs are trained on static and pre-defined datasets. For example, the Llama-3.1 model is an open-source large language model by Meta, with a training dataset over 15 trillion tokens (Llama Team 2024). However, in practical applications, language usage habits, information content, and user needs are all dynamically changing (Wu et al. 2024). On the other hand, once training is complete, the model becomes fixed, and the cost and computational demands of retraining or incremental pre-training is extremely high. For example, the GPT-3 model has 174.6 billion parameters, and retraining it once requires approximately 3640 PF-days of computing power (i.e., performing 10 quadrillion calculations per second for 3640 days) (Brown et al. 2020). Therefore, how to adapt downstream tasks *effectively* and *efficiently* with updating the model with the new knowledge while retaining the existing knowledge has become an important and urgent topic.

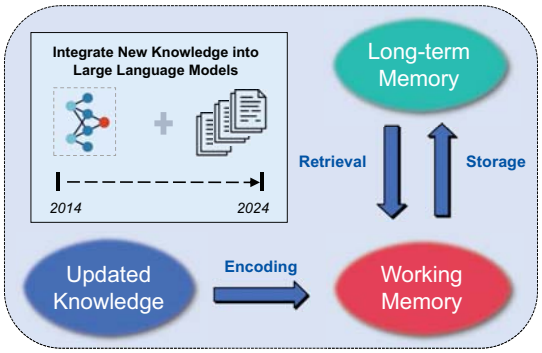

Figure 1: In general, memory can be divided into three stages: (1) *encoding* involves reorganizing and transforming external information; (2) *storage* entails hierarchically categorizing and preserving information in long-term memory; (3) *retrieval* extracts information from long-term memory.

To address these challenges, existing continual learning methods dynamically update new incremental knowledge through techniques such as data replay and incremental task parameters while balancing the generalization of new and old knowledge (Schwarz et al. 2018; Riemer et al. 2018; Shi and Wang 2024). However, data replay and incrementally adding task parameters bring non-negligible computational overhead. Additionally, methods like model editing only provide patches to the model, resulting in poor generalization ability and even causing collapse (Yao et al. 2023). While it is possible to perform thousands of edits simultaneously, scalability is low when updating a large amount of knowledge in large models, and catastrophic forgetting can occur. On the other hand, memory is the foundation of human intelligence while humans use memory to achieve continual learning. As shown in Figure 1, memory impact on intellectual activities such as learning, abstraction, association, and reasoning in the human brain spans three stages: (1) Encoding: It in-

volves reorganizing and transforming external information. The efficiency of learning depends on the strategies used for memory encoding (Pyc and Rawson 2010). Strategies such as multi-channel encoding and contextual association can significantly enhance learning outcomes. (2) Storage: Information is stored hierarchically and categorized in long-term memory. Retaining learned content makes subsequent learning more efficient (Bjork 1994). (3) Retrieval: It involves extracting and aggregating information from long-term memory. It consolidates memory storage, stimulates meta-cognitive abilities, and promotes reasoning, abstraction, and association (Karpicke and Roediger III 2008). Hence, how to draw on human memory mechanisms to the continual learning process has become an interesting and possible direction for adapting to changing world knowledge of LLMs.

To this end, we introduce the **C**ompression **M**emory **T**raining (**CMT**) method that encodes knowledge extracted from new documents into a dynamic memory bank within its latent space, serving as long-term memory for subsequent retrieval and aggregation. The core idea is to freeze the parameters of the LLM itself and construct a memory-based module that learns to automatically encode and collect relevant information. Specifically, we first utilize an instantiable compressor to compress information from new documents into compact representations, which are cached to maximize the performance of the LLM on unseen tasks. Different from Tack et al. (2024), this representation is generated through memory tokens with decoder-only model representing compressed knowledge, resulting in a memory bank that is less redundant than traditional knowledge bases in retrieval-based methods or contexts in prompting compression methods. Thus, during online adaptation, each document stream instance is stored in the memory bank. It allows contexts to be pre-computed offline once and reducing the LLM's computational costs at inference. Next, we learn to aggregate representations (i.e., memory) in the feature space into a single representation based on the given query, which is then mapped into cached key-value pairs within each transformer layer of the LLM. To ensure the effectiveness and scalability of CMT, we further propose three training and inference techniques corresponding to the encoding, retrieval and aggregation stages of memory respectively: (1) memory-aware objective; (2) self-matching; and (3) top-k aggregation. The evaluation of CMT focuses on several key aspects: (1) Integration of new knowledge. The model's performance is assessed with downstream QA tasks, where CMT demonstrates substantial improvements over existing methods, indicating the superiority of supplementing LLMs with CMT; (2) Knowledge retention. CMT is evaluated on knowledge retention experiments under scenarios with different numbers of adapted documents, showcasing its ability to recall knowledge; (3) Robustness. We use the proportion of unrelated documents as a measure to test the model's performance in the presence of irrelevant interference. The results show that CMT outperforms competitive baselines, demonstrating superior robustness.

Our contributions are summarized as follows:

- We introduce CMT that incorporates an integrated memory bank within the latent space to address the challenges of continual learning of LLMs.
- To utilize encoded memory more efficiently, we further propose three effective training and inference strategies.
- CMT demonstrates competitive performance across three benchmarks and knowledge retention settings, showcasing its versatility, effectiveness, and robustness.

Related Work

Memory-Augmented Models Memory-augmented models are not a new concept. Early memory networks introduced computational methods to store contextual information in limited space, thereby enhancing inference efficiency (Weston, Chopra, and Bordes 2015; Sukhbaatar et al. 2015; Ba et al. 2016). Following this, Memory Transformer (Burtsev and Sapunov 2020) and RMT (Bulatov, Kuratov, and Burtsev 2022) proposed adding memory tokens when reading contexts. However, expanding memory and incorporating information without disrupting the model's original capabilities remains a long-term challenge (Khandelwal et al. 2019; Zhong, Lei, and Chen 2022; Modarressi et al. 2023; Moro et al. 2023; Zhong et al. 2023; Wang et al. 2023; Yang et al. 2024). Recent research has also focused on compressing prompts to enhance LLM inference efficiency (Wingate, Shoeybi, and Sorensen 2022; Snell, Klein, and Zhong 2022; Phang et al. 2023). For instance, AutoCompressor (Chevalier et al. 2023) and ICAE (Ge et al. 2024) propose auto-encoding methods for compressing contexts into soft embeddings. Gisting (Mu, Li, and Goodman 2024) introduces learnable tokens to compress context information within attention hidden states. Moreover, several improvements to transformers have demonstrated the benefits of equipping LLMs with external, controllable memory (e.g., MemoryLLM) (Kim et al. 2023; He et al. 2024; Wang et al. 2024b). However, these methods have not yet been applied to continual knowledge learning for existing LLMs as they typically require training from scratch and rely on inflexible and non-reusable implementations.

Continual Learning of LLMs Continual learning aims to integrate LLMs into dynamic data distributions, task structures, and user preferences without significantly degrading performance in learned domains (Zheng et al. 2024; Shi et al. 2024). This involves sequentially training models on a series of tasks with the goal of maintaining performance across all tasks (Kirkpatrick et al. 2017; Li and Hoiem 2017; Riemer et al. 2018; Buzzega et al. 2020). During training, models often have limited or no access to previous data, making it challenging to retain past knowledge since optimization constraints from previous data are absent during current-task learning (Li and Hoiem 2017; Buzzega et al. 2020; Smith et al. 2023; Shi and Wang 2024). This challenge, known as *catastrophic forgetting* (McCloskey and Cohen 1989), has been a central focus since its inception. Over the years, researchers have explored various techniques to mitigate forgetting in models. These include replay-based methods (Schwarz et al. 2018; Riemer et al. 2018; Shi and Wang 2024), parameter regularization (Kirkpatrick et al. 2017; Ritter, Botev, and Barber 2018; Aljundi et al. 2018; Sprechmann et al. 2018), and model architecture expansion (Wang et al. 2022). Recently, in the context of continual learning for LLMs, the

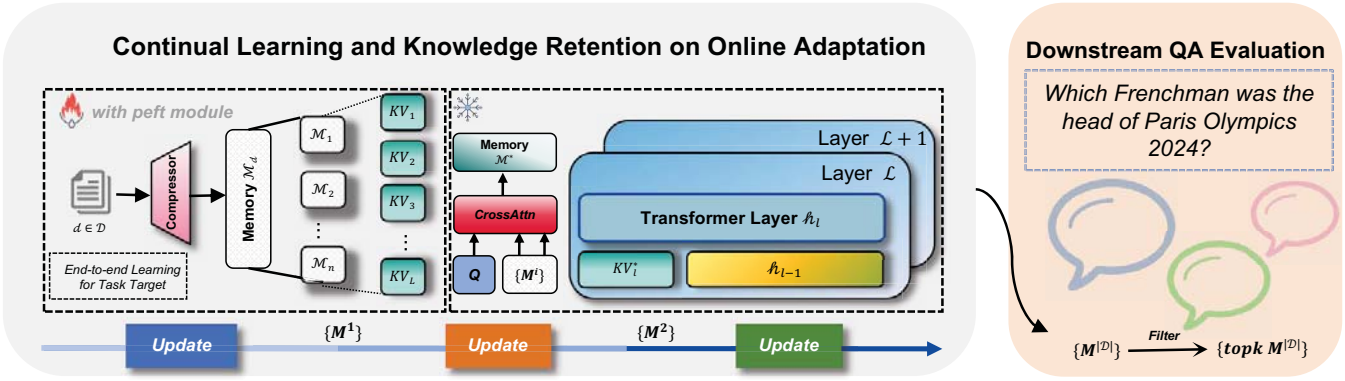

Figure 2: Illustration of compression memory training method. During the adaptation process, each document to be learned will be compressed into the dense vector by a compressor, and these vectors $\{M^{|D|}\}$ will be aggregated through the cross attention mechanism and sent to LLMs together with the question for answer output. The training goal is the accuracy of the downstream task answer. In the online adaptation stage, all documents will be compressed into vectors and then filtered and aggregated.

challenge has shifted from storage efficiency to computational efficiency (Song et al. 2023; Wang et al. 2024a). In this paper, we focus on integrating memory systems into continual learning, enabling encoding, retrieval, and aggregation of knowledge without the need for expensive retraining.

Method

Task Formulation

We consider the scenario where an outdated model, f_θ, is updated using an online stream of recent documents, $D_{\text{test}} = \{x_i\}$. This process produces an updated model, $f_{\theta+\triangle\theta}$, which is then evaluated against a set of queries, $Q_{\text{test}} = \{q_i\}$, with corresponding labels, $Y_{\text{test}} = \{y_i\}$. Each query q_i and its label y_i are derived from a distribution related to the corresponding document x_i: $q_i, y_i \sim p(q_i, y_i \mid x_i)$. For instance, q_i could be a question about some information in document x_i, with y_i being the answer provided by the document. A key constraint is that during the update process using D_{test}, we do not have access to Q_{test}. Thus, our methodology for updating f_θ must be general rather than query-specific. To make this problem tractable, we assume the availability of an additional corpus of documents D_{train} and corresponding query samples Q_{train} and labels Y_{train} generated by a similar process to $Q_{\text{test}}, Y_{\text{test}}$. This training set enables learning the types of queries that may be of interest, guiding us on how to update our model to optimize performance on test queries while minimizing disruption to its prior knowledge and behaviors. We define the adaptation process involving D_{train}, Q_{train}, and Y_{train} as the *learning* phase, while the process involving D_{test} is referred to as the *online adaptation* phase. We next describe the method that adjusts the learning phase to more efficiently update our base model on the test stream of documents D_{test}.

CMT: Compression Memory Training

Our goal is to efficiently adapt given LLMs to unseen knowledge while retaining previously learned knowledge, whether from the original pre-training stage or updates from documents in a stream of new data. To achieve this, we designed a learning method called **C**ompression **M**emory **T**raining shown in Figure 2. During the online adaptation phase, it only requires a single forward pass to compress the documents knowledge into memory, which avoids the cost of gradient computation. Here, we first introduce the general process of the method.

First, each document d in the document set D is compressed into condensed vectors $\boldsymbol{M}$ through the compressor Θ. Then, these dense vectors corresponding to each document are further aggregated. The aggregated vectors are further mapped and input into the LLM Φ to be adapted in the form of cached key-value pairs. The LLM Φ to be adapted freeze their parameters during both the training and online adaptation phases, reducing the risk of catastrophic forgetting during continuous updating. The parameters to be learned include memory encoding, storage, and mapping parts. The entire network is trained during the learning phase, with the learning objective being to better answer Q_{train}.

Compression Memory We introduce the method of compressing documents D into condensed vectors. It aims to transform lengthy documents into concise, compact representations while striving to maintain the core semantics and integrity of the original knowledge. We define a document $d \in D$ as $w = (w_1, w_2, \ldots, w_n, c_1, c_2, \ldots, c_k)$, where w_i means the i-th token of document d, c_j means the j-th soft virtual token adhere to this document, and n is the number of actual tokens in document d. Let $\boldsymbol{e}(\cdot)$ represent the word embedding lookup in the LLM and $\boldsymbol{m}(\cdot)$ represent the learnable embeddings of soft tokens $c_1, c_2, \ldots, c_k$. A document compressor model Θ utilizes the document embeddings $\boldsymbol{e}(w) = (\boldsymbol{e}(w_1), \boldsymbol{e}(w_2), \ldots, \boldsymbol{e}(w_n))$ and the soft token embeddings $\boldsymbol{e}_{soft}(c) = (\boldsymbol{e}_{soft}(c_1), \boldsymbol{e}_{soft}(c_2), \ldots, \boldsymbol{e}_{soft}(c_k))$ to produce compact representations $\boldsymbol{M} = (\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_k) \in \mathbb{R}^{k \times d}$ of the document d, where k is the length of the compressed document and $k \ll n$. The condensed vectors $\boldsymbol{M}$ can replace the original context and be combined with other prompt embeddings $\boldsymbol{e}(p) = (\boldsymbol{e}(p_1), \ldots, \boldsymbol{e}(p_l))$ for input to an LLM Φ. The output $y = (y_1, \ldots, y_m)$ remains faithful to the content

of the original context w. As illustrated in Figure 2, inspired by Ge et al. (2024), the compressor can be instantiated as a series of cross-attention layers, pre-trained decoder models, and encoder-decoder models with a set of learnable soft tokens, termed condensed tokens. Here, the compressor utilizes document tokens and condensed tokens as inputs, leveraging a causal Transformer decoder to compress the document information into condensed vectors. We leave the application of these vector across different LLMs for future work.

Memory Aggregation Given the memory bank of compressed documents D represented as $\{\boldsymbol{M}^i\}_{i=1}^{|D|}$, we aim to learn how to select most relevant information in the form of a transformation $\boldsymbol{M}^* \in \mathbb{R}^{k\times d}$ for a given input q_i. There are two feasible methods: (1) Retrieve one or multiple memory units and map them into the LLM space. For example, xRAG (Cheng et al. 2024) addresses context aggregation from a multi-modal fusion perspective. It introduces a modality projector trained to directly project retrieved dense vectors into the LLM representation space. However, this approach has the risk of selecting incorrect memory units and requires a pre-training phase to learn how to resolve relationships among different memories. (2) Linearly interpolate multiple memory units, aggregate them by weights into a single memory unit, and map it into the LLM space. For example, Sukhbaatar et al. (2015) computes the weighted sum of the memory bank as the representative vector of the memory. The advantage of this method is that it can leverage ideas like attention mechanisms, model soups (Wortsman et al. 2022) and the mixture-of-experts method (Shazeer et al. 2017) to filter and aggregate different memory units, maintaining permutation invariance of the memory units. However, such methods do not consider the relative position information of soft tokens within memory units during aggregation. Moreover, as we discussed in the task definition, the source of accurate information for answering question Q_i is mostly related to the i-th document. Therefore, similar to Tack et al. (2024), we select $\boldsymbol{M}^*$ using cross-attention blocks (Vaswani et al. 2017; Kim et al. 2019; Xu et al. 2020), with the set aggregation network ψ:

$$\boldsymbol{M}^* = \psi\big(\Theta(q_i), \{\boldsymbol{M}^i\}_{i=1}^{|D|}\big) \tag{1}$$

Here, for reasons of efficiency and consistency in the representation space, we use a document compressor Θ to compress the input q_i (e.g., user query). Using an additional question encoder is left for future work. As the vanilla cross-attention mechanism suffers from capturing the relative positional relationships among soft tokens within the document d. It implies that swapping any two tokens in the memory results in an identical condensed vector. Hence, in the aggregation process, we apply RoPE (Su et al. 2024) to represent the relative positional relations within the soft tokens. We only perform position embedding operations on query and key. And we allocate positional embeddings as if placing the soft tokens subsequent to the context tokens. The RoPE embeddings R_i and R_j manifests the relative positional relationships through the inner product between $Q_{\mathrm{pos}} = \{\boldsymbol{q}_i\} := \Theta(q_i)$ and $K_{\mathrm{pos}} = \{\boldsymbol{k}_i\} := \{\boldsymbol{M}^k\}_{k=1}^{K}$:

$$(\mathrm{R}_i\boldsymbol{q})^T(\mathrm{R}_j\boldsymbol{k}) = \boldsymbol{q}^T\mathrm{R}_i^T\mathrm{R}_j\boldsymbol{k} = \boldsymbol{q}^T\mathrm{R}_{j-i}\boldsymbol{k} \tag{2}$$

In this way, each soft token can recognize the relative positions relations of both intra- and inter-document soft tokens.

Alignment for LLM After obtaining $\boldsymbol{M}^* \in \mathbb{R}^{k\times d}$, we do not intend to use the memory as an embedding layer input to the LLM Φ, as this does not fully leverage the memory to promote the association. Therefore, we design a network π to map the original memory representation into cached key-value pairs (with the number being the actual tokens count). The purpose of π is to perform self-attention on the memory tokens and use multiple multi-layer perceptrons to transform the memory tokens' features into actual tokens' features. Specifically, the module handles actual tokens by repeating memory tokens and recombines the processed features into the final output. Here, the actual tokens are defined as the new virtual tokens in each layer multiplied by the number of layers, then multiplied by 2 (i.e., key and value).

Training Objective To train the memory embeddings $\boldsymbol{e}_{soft}$, the compressor Θ, the aggregation networks ψ and alignment module π, we optimize both networks end-to-end using the loss function $\mathcal{L}$, which is the negative log-likelihood of the given label y:

$$\mathcal{L} = \min_{D_{\text{train}}, Q_{\text{train}}, Y_{\text{train}}} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\big(\mathrm{LM}_\theta(q_i; \pi(\boldsymbol{M}^*)), y_i\big) \tag{3}$$

where N is the number of the training queries and labels in Q_{train}. Note that we do not update the static LLM θ to avoid the risk of catastrophic forgetting by overwriting important parameters. It is important to train the model using the cross-entropy loss of the final QA task. We experimented with document auto-encoding pretraining tasks (Ge et al. 2024), with dividing it into two stages or using multi-task learning, but neither approach resulted in significant improvements.

Online Adaptation After training the entire network on a given training corpus D_{train}, Q_{train}, and Y_{train}, we introduce an online adaptation phase. Formally, the CMT processes a stream of test documents D_{test} that are sequentially fed to the LLM. Considering that the task query $q^i_{\text{test}} \in Q_{\text{test}}$ is unavailable during the adaptation process, we first store the condensed representation of the document in the memory $\boldsymbol{M} = \{\Theta(d^i_{\text{test}})\}_{i=0}^{|D_{\text{test}}|}$ and later use the aggregation network to predict the modulation to adapt the LLM:

$$\hat{y}^i_{\text{test}} = \mathrm{LLM}_\theta(q^i_{\text{test}}; \pi(\boldsymbol{M}^*)) \tag{4}$$

where $\boldsymbol{M}^* = \psi\big(\Theta(q^i_{\text{test}}), \boldsymbol{M}\big)$.

Effective Learning Strategy

Memory-Aware Conditional Objective To enhance the model's utilization of memory, we propose a training objective of contrastive ensemble between the logits. It enables the model to account for external knowledge which may not be aligned with the model's training data. Specifically, we adopt the vanilla logits from LLM as l_θ for the prior knowledge $l_\theta(y_i \mid \boldsymbol{q}_i)$. We demote this knowledge from the model's original output distribution via $\frac{l_\theta(y_i|\boldsymbol{M}^*,\boldsymbol{q}_i)}{l_\theta(y_i|\boldsymbol{q}_i)}$. We choose this formulation because it also represents the pointwise mutual

information between the external knowledge from the document set $\boldsymbol{M}^*$ conditioned on $\boldsymbol{q}_i$. Optionally, one can adjust the original distribution by $\frac{l_\theta(y_i|\boldsymbol{M}^*,\boldsymbol{q}_i)}{l_\theta(y_i|\boldsymbol{M}^-,\boldsymbol{q}_i)}$, where $\boldsymbol{M}^-$ represents an explicit knowledge one wants to demote from (e.g., a set of other unrelated documents). We interpolate this demotion $\frac{l_\theta(y_i|\boldsymbol{M}^*,\boldsymbol{q}_i)}{l_\theta(y_i|\boldsymbol{q}_i)}$ and the original output logits $l_\theta(y_i \mid \boldsymbol{M}^*, \boldsymbol{q})$ via a product-of-experts weighted by α. We sample y_i from the reweighted distribution:

$$y_i \propto l_\theta(y_i \mid \boldsymbol{M}^*, \boldsymbol{q}_i) \left(\frac{l_\theta(y_i \mid \boldsymbol{M}^*, \boldsymbol{q})}{l_\theta(y_i \mid \boldsymbol{q}_i)} \right)^\alpha \tag{5}$$

Further, we normalize it across all possible y_i:

$$y_i \sim (1+\alpha) l_\theta(y_i \mid \boldsymbol{M}^*, \boldsymbol{q}_i) - \alpha l_\theta(y_i \mid \boldsymbol{q}_i) \tag{6}$$

where larger α means more weight on our adjustment (we set it to 0.5) and $\alpha = 0$ reduces to standard negative log-likelihood. If we identify an external knowledge $\boldsymbol{M}^*$ conditionally independent to the generation, $l_\theta(y_i \mid \boldsymbol{M}^*, \boldsymbol{q}_i) = l_\theta(y_i \mid \boldsymbol{q}_i)$, even a non-zero α would not have an impact on the original output distribution.

Knowledge Transfer with Self-Matching Recall that during the aggregation process, we calculate the attention weights between the query and the memory bank. These weights represent the contribution of each document to answering the current question to some extent. Therefore, this inspires us to leverage the nature of the task to capture significant features of memory during continual learning. Note that the specific vector $\mathbf{q} \in \mathbb{R}^d$ for each query has the same dimension as the memory unit $\mathbf{m}_i \in \mathbb{R}^d$. Then, we calculate the cosine similarity between the query embedding and the memory vectors for the current i^{th} document, as the document matching score $\alpha[:i] = \cos(\mathbf{q}_i, \boldsymbol{M}[:i])$. The additional training objective for updating the i^{th} document is to maximize the cosine similarity between $\mathbf{m}_i$ and the corresponding query embedding $\mathbf{q}$. To prevent the model from collapsing all documents into similar vectors, we add a uniformity term to penalize excessive similarity between document embeddings, promoting diversity by encouraging larger pairwise distances between different memory vectors.

Inference with Top-k Aggregation Additionally, we employ a filtering mechanism to handle a large memory bank during downstream task inference. Let n and $|D|$ be the number of output tokens for each context and the number of memory units, respectively. Then, the memory usage of t cross-attention layers in the memory aggregation becomes $t \cdot \mathcal{O}(|D|n^2)$. It indicates that the memory cost of the aggregation process scales linearly with the size of the memory. Unlike previous work (Tack et al. 2024) that reduces memory consumption using hierarchical modulation aggregation, here we use a simple but effective top-k filtering method. Specifically, for a given memory bank with k units, we first compute the similarity between them and query, then sort by similarity and filter according to the window size used during training. Hence, the memory bank window size seen by the model in the aggregation process during training and testing is consistent, which requiring no additional training or modifications to the primary training objective. It also reduces the impact of overfitting and noise, and yields better results. Similar observations are reported in retrieval-augmented generation works (Qin et al. 2023; Cuconasu et al. 2024).

Experiments

We conduct extensive experiments on these Question Answering (QA) benchmark datasets to answer the following Research Questions (**RQs**):

- **RQ1**: How does our model contribute to QA accuracy compared with other state-of-the-art methods?
- **RQ2**: How effective are the key components in our model, such as the self-matching of memory vectors?
- **RQ3**: Can our model demonstrate robustness against knowledge interference from irrelevant documents?
- **RQ4**: How does our model perform in terms of the forgetting and plasticity dynamics within the document stream?

StreamingQA The StreamingQA dataset (Liška et al. 2022) features both human-written and language model-generated questions. These questions are sourced from English WMT news articles published between 2007 and 2020. Each question is linked to a complete WMT news article, with an average length of about 500 tokens per article. For learning purposes, question-article pairs from post-2018 publications are used, resulting in 21k training questions, 1.7k validation questions, and 5k test questions.

SQuAD The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016) includes questions created by humans based on Wikipedia articles. The answer to each question is a text span from a specific paragraph within the article. Typically, paragraphs are around 150 tokens. We utilize the validation set of SQuAD as our test set and divide the training set into four additional splits. It results in 39.9k training questions, 5.6k validation questions, and 10.6k test questions. Additionally, we use 8.6k training documents, 1.2k validation documents, and 2.1k test documents.

ArchivalQA The ArchivalQA dataset (Wang, Jatowt, and Yoshikawa 2022) contains questions generated automatically from the New York Times Annotated Corpus (Sandhaus, Evan 2008). Each answer is a text span within an article, with questions paired to paragraphs from NYT articles. We split the validation set of ArchivalQA into five segments for our study. This setup provides us with 21.7k training questions, 5.3k validation questions, and 8.7k test questions. For documents, we utilize 12.8k training documents, 3.0k validation documents, and 5.0k test documents.

Experiment Settings We conducted extensive experiments using 4 LLMs as backbones, including the GPT-2 series (Radford et al. 2018) and the Llama-2 series (Touvron et al. 2023b). The model parameters are 82M, 774M, 1.5B, and 7B, respectively. Larger models, such as the 70B, were not included due to insufficient computational resources for training. For the compressor, unlike previous work, we chose a larger decoder model, Llama-2-7b, and used Parameter-Efficient Fine-Tuning (PEFT) with a rank of 6 and a LoRA alpha set to 32, employing a different encoding method as

Datasets	Method	DistilGPT2		GPT2-Large		GPT2-XL		Llama-2	
		EM	F_1	EM	F_1	EM	F_1	EM	F_1
StreamingQA	Uniform	1.62	3.76	4.74	7.00	5.11	7.48	12.43	13.54
	Salient Spans	1.44	4.67	4.86	8.54	5.40	9.42	13.33	18.97
	CaMeLS	1.62	5.79	5.35	10.60	6.55	11.67	-	-
	MAC	5.59	10.18	7.25	13.31	8.99	15.38	14.29	21.79
	CMT (ours)	**6.43**	**12.32**	**7.32**	**13.43**	**9.61**	**16.48**	**18.36**	**25.98**
SQuAD	Uniform	1.24	2.54	3.64	4.97	6.10	6.78	13.25	17.01
	Salient Spans	1.03	2.47	4.03	6.48	4.55	6.74	13.74	18.66
	CaMeLS	1.47	3.08	4.97	8.63	6.70	10.15	-	-
	MAC	2.01	6.85	6.43	11.42	7.10	12.55	15.07	21.14
	CMT (ours)	**3.12**	**7.59**	**7.15**	**12.45**	**9.81**	**12.85**	**19.54**	**25.50**
ArchivalQA	Uniform	4.86	4.08	7.66	8.71	8.61	10.78	18.53	21.35
	Salient Spans	4.52	3.76	9.75	11.19	11.81	14.11	18.97	22.75
	CaMeLS	4.62	6.19	9.92	12.41	13.87	15.74	-	-
	MAC	7.55	10.58	11.84	15.26	14.01	17.12	20.12	23.90
	CMT (ours)	**8.15**	**11.03**	**12.28**	**16.12**	**14.55**	**18.01**	**21.73**	**25.40**

Table 1: Performance comparisons of online adaptation with CMT are presented. We report the Exact Match (EM) and F_1 scores after adapting the LLMs on a stream of documents and subsequently conducting downstream QA for test. We use the average of 3 random seeds and baseline results are from the corresponding papers. The **boldfaced** means the best results for this dataset.

well. We performed 1,000 steps of pre-training using English Wikipedia with auto-encoder tasks, as additional steps did not yield consistent improvements. The number of soft tokens used is 24. Additionally, we found that while increasing the number of tokens beyond 24 initially led to marginal gains, the performance plateaued or slightly decreased due to overfitting and increased computational overhead. We evaluated online adaptation performance on a test dataset composed of documents and QA pairs. Following Tack et al. (2024), we adapted the LLMs using 1,665 documents and then assessed its performance post-adaptation. To test the model's general understanding, QA pairs were sampled from these documents. Each document contains up to 1,024 tokens. Cross-attention involves 4 blocks. The batch sizes for updates and validation are 8 and 16 respectively, with gradient accumulation over 4 steps. The optimizer is AdamW, and training will run for 50 epochs with validation every 250 steps. The learning rate is set to $1e^{-6}$ with a warmup ratio of 0.01 and a constant-with-warmup schedule. During training, we sample the document memory in the same batch and at inference k is equal to training batch size. We run on the NVIDIA A100 80G GPUs.

Baseline We include the online fine-tuning baselines introduced in Tack et al. (2024), including Uniform, Salient Spans, CaMeLS and MAC. The uniform baseline uses uniform token weighting kearning documents and involves additional fine-tuning for question answering after adaptation. Salient Spans assigns uniform weights to tokens in salient spans (Guu et al. 2020) and no weights to other tokens. CaMeLS leverages the output of a token-weighting language model (i.e., meta-learned to predict important tokens to maximize the performance of the adapted LLM). Memory of Amortized Contexts (MAC) is an efficient online learning framework that uses the modulation to integrate new document knowledge.

Model Comparison (RQ1)

Table 1 illustrates the performance of CMT in online adaptation compared to other baselines. CMT consistently outperforms these baseline methods across all datasets and models, demonstrating its superior capabilities in continual learning and knowledge retention on online adaption. These advantages are particularly evident in larger models, indicating that CMT effectively scales with model size and complexity. For instance, in the StreamingQA dataset, CMT consistently surpasses other methods for all model variants. Specifically, with DistilGPT2, CMT achieves an EM score of 6.43 and an F_1 score of 12.32, outperforming the next best method, MAC, which scores 5.59 (EM) and 10.18 (F_1). This performance gap widens with larger models, with CMT achieving the highest scores on Llama-2 (EM: 18.36, F_1: 25.98). This demonstrates CMT's superior ability to incorporate and retain new knowledge in real-time. Furthermore, CMT is efficient in terms of memory usage and adaptation time. Unlike CaMeLS, CMT does not require gradient computation for updates. Furthermore, CMT reduces the proportion of trainable parameters by 5.4 times and inference time due to top-k aggregation compared to MAC, facilitating easier scalability with larger document corpora and model sizes.

Ablation Study (RQ2)

This section presents an ablation study to assess the impact of various components of the CMT on its performance. Table 2 summarizes the results across three datasets. We evaluate the full CMT model (1) and three variants: CMT without the Memory-Aware Objective (2), CMT without Self-Matching (3), and CMT without Top-k Aggregation (4). For example, in the StreamingQA dataset, the full CMT model achieves an EM score of 18.36 and an F_1 score of 25.98, outperforming

#	Method	StreamingQA		SQuAD		ArchivalQA	
		EM	F_1	EM	F_1	EM	F_1
(1)	**CMT**	18.36	**25.98**	**19.54**	**25.50**	21.73	25.40
(2)	w/o Memory-Aware Objective	**18.54**	23.71	15.38	22.77	20.89	24.18
(3)	w/o Self-Matching	17.87	22.54	17.97	23.40	**22.43**	**25.68**
(4)	w/o Top-k Aggregation	16.43	20.13	18.35	24.12	21.09	23.99

Table 2: Results of ablation study where the best results are **boldfaced** and the second-best results are underlined.

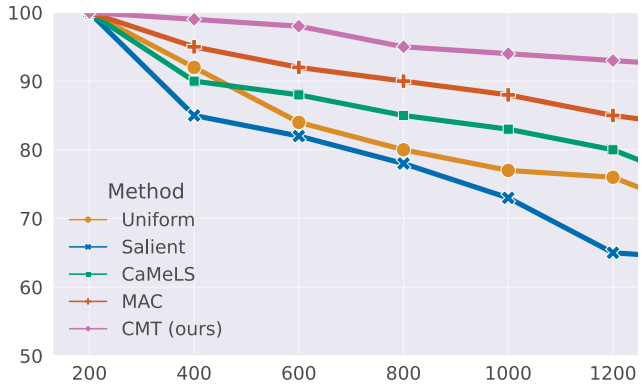

Figure 3: Knowledge retention analysis under `Llama-2-7b` trained on StreamingQA dataset.

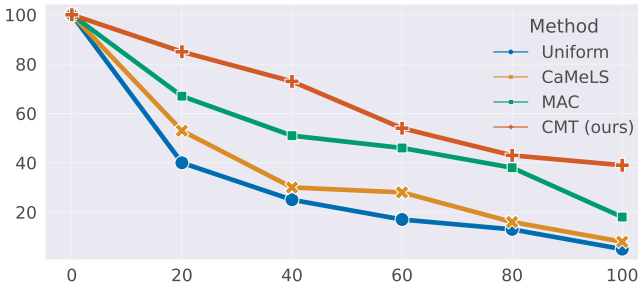

Figure 4: Performance of robustness analysis experiments. We use synthetic unrelated documents to test the impact of irrelevant interference brought by memory integration.

all other variants. The removal of the Memory-Aware Objective (variant 2) slightly increases the EM score to 18.54 but decreases the F_1 score to 23.71. The absence of Self-Matching (variant 3) results in lower scores (EM: 17.87, F_1: 22.54), indicating the importance of this component. The variant without Top-k Aggregation (4) shows the lowest performance (EM: 16.43, F_1: 20.13), highlighting its critical role in CMT. The ablation study reveals the relative importance of each CMT component. The Memory-Aware Objective and Top-k Aggregation are crucial for maximizing performance across most datasets. The Top-k aggregation also helps by focusing on relevant memory units, reducing inference latency by 18%. Self-Matching, while generally beneficial, can sometimes be omitted without severe performance degradation, as seen in the ArchivalQA results. However, the full CMT model consistently provides the best or near-best performance, validating the integrated approach.

Knowledge Retention (RQ3)

Following Tack et al. (2024), we evaluate retention ratio determined by the decline in the F_1 score of the initially adapted 200 documents during online adaption. As shown in Figure 3, CMT and CaMeLS lead in performance, followed by MAC, Salient, and finally the Uniform method. The plot indicates that all methods reduce from an increased number of online adaptation documents, as shown by the downward trends in both F_1 score and retention rate. However, the rate of improvement varies among the methods. Methods like CMT and CaMeLS show a higher scalability factor, indicating that they are better suited for environments where the volume of adaptation data is substantial. The gap between the highest (CMT) and the lowest performing method (Uniform) widens as the number of documents increases, highlighting the importance of choosing a more efficient method for large-scale online adaptation tasks.

Robustness Analysis (RQ4)

We make use of irrelevant synthetic data, which is obtained by generating text that is irrelevant to the current document using `gpt-4o`. As shown in Figure 4, the performance of the Uniform shows a significant decline as the proportion of irrelevant documents increases. Initially maintaining a high relative F_1 score, the performance deteriorates sharply beyond the 20%, indicating a high sensitivity to irrelevant data. CaMeLS exhibits a more robust performance compared to the Uniform method. However, a noticeable performance drop is still observed beyond the 60% threshold. The MAC method demonstrates a relatively stable performance across different proportions of irrelevant documents. While there is a gradual decline in the F_1 score, it is less pronounced compared to the Uniform and CaMeLS methods, highlighting MAC's effectiveness in handling irrelevant data. Among the four methods, CMT shows the best performance stability. The relative change in F_1 score remains minimal even as the proportion of irrelevant documents approaches 100%.

Conclusion

We propose a continual learning method for LLMs named CMT including the memory bank in a latent space as the model's updatable knowledge parameters. CMT can update the memory with new knowledge, enabling effective knowledge integration and slow forgetting of prior knowledge.

Acknowledgements

We thank reviewers and ACs for their comments. This work is supported by the Natural Science Foundation of China (Grant No. 62406088) and the Guangdong Basic and Applied Basic Research Foundation (2023A1515110078) and the Natural Science Foundation of China (Grant No. 62376067). Baotian Hu is the corresponding author.

References

Aljundi, R.; Babiloni, F.; Elhoseiny, M.; Rohrbach, M.; and Tuytelaars, T. 2018. Memory aware synapses: Learning what (not) to forget. In *ECCV*.

Ba, J.; Hinton, G.; Mnih, V.; Leibo, J. Z.; and Ionescu, C. 2016. Using fast weights to attend to the recent past. In *ICONIP*.

Bjork, R. A. 1994. Memory and Metamemory Considerations in the. *Metacognition: Knowing about Knowing*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

Bulatov, A.; Kuratov, Y.; and Burtsev, M. S. 2022. Recurrent Memory Transformer. In *NeurIPS*.

Burtsev, M. S.; and Sapunov, G. V. 2020. Memory Transformer. *CoRR*.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; and Calderara, S. 2020. Dark experience for general continual learning: a strong, simple baseline. *NeurIPS*.

Cheng, X.; Wang, X.; Zhang, X.; Ge, T.; Chen, S.; Wei, F.; Zhang, H.; and Zhao, D. 2024. xRAG: Extreme Context Compression for Retrieval-augmented Generation with One Token. *CoRR*.

Chevalier, A.; Wettig, A.; Ajith, A.; and Chen, D. 2023. Adapting Language Models to Compress Contexts. In *EMNLP*.

Cuconasu, F.; Trappolini, G.; Siciliano, F.; Filice, S.; Campagnano, C.; Maarek, Y.; Tonellotto, N.; and Silvestri, F. 2024. The Power of Noise: Redefining Retrieval for RAG Systems. In *SIGIR*.

Ge, T.; Jing, H.; Wang, L.; Wang, X.; Chen, S.-Q.; and Wei, F. 2024. In-context Autoencoder for Context Compression in a Large Language Model. In *ICLR*.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M. 2020. Retrieval Augmented Language Model Pre-Training. In *ICML*.

He, Z.; Qin, Z.; Prakriya, N.; Sun, Y.; and Cong, J. 2024. HMT: Hierarchical Memory Transformer for Long Context Language Processing. *CoRR*.

Karpicke, J. D.; and Roediger III, H. L. 2008. The critical importance of retrieval for learning. *science*.

Khandelwal, U.; Levy, O.; Jurafsky, D.; Zettlemoyer, L.; and Lewis, M. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Kim, H.; Mnih, A.; Schwarz, J.; Garnelo, M.; Eslami, S. M. A.; Rosenbaum, D.; Vinyals, O.; and Teh, Y. W. 2019. Attentive Neural Processes. In *ICLR*.

Kim, J.; Yeom, J.; Yun, S.; and Song, H. O. 2023. Compressed Context Memory For Online Language Model Interaction. *CoRR*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*.

Liška, A.; Kočiský, T.; Gribovskaya, E.; Terzi, T.; Sezener, E.; Agrawal, D.; de Masson d'Autume, C.; Scholtes, T.; Zaheer, M.; Young, S.; Gilsenan-McMahon, E.; Austin, S.; Blunsom, P.; and Lazaridou, A. 2022. StreamingQA: A Benchmark for Adaptation to New Knowledge over Time in Question Answering Models. arXiv:2205.11388.

Llama Team, A. . M. 2024. The Llama 3 Herd of Models.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem.

Modarressi, A.; Imani, A.; Fayyaz, M.; and Schütze, H. 2023. RET-LLM: Towards a General Read-Write Memory for Large Language Models. *arXiv preprint arXiv:2305.14322*.

Moro, G.; Ragazzi, L.; Valgimigli, L.; Frisoni, G.; Sartori, C.; and Marfia, G. 2023. Efficient Memory-Enhanced Transformer for Long-Document Summarization in Low-Resource Regimes. *Sensors*.

Mu, J.; Li, X.; and Goodman, N. 2024. Learning to compress prompts with gist tokens. *NeurIPS*.

OpenAI. 2023. GPT-4 Technical Report. *CoRR*.

Phang, J.; Mao, Y.; He, P.; and Chen, W. 2023. Hypertuning: Toward adapting large language models without backpropagation. In *ICML*.

Pyc, M. A.; and Rawson, K. A. 2010. Why testing improves memory: Mediator effectiveness hypothesis. *Science*.

Qin, Z.; Jagerman, R.; Hui, K.; Zhuang, H.; Wu, J.; Shen, J.; Liu, T.; Liu, J.; Metzler, D.; Wang, X.; and Bendersky, M. 2023. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. *CoRR*.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training. In *preprint*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250.

Riemer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2018. Learning to learn without forgetting

by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

Ritter, H.; Botev, A.; and Barber, D. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. *NeurIPS*.

Sandhaus, Evan. 2008. The New York Times Annotated Corpus.

Schwarz, J.; Czarnecki, W.; Luketina, J.; Grabska-Barwinska, A.; Teh, Y. W.; Pascanu, R.; and Hadsell, R. 2018. Progress & compress: A scalable framework for continual learning. In *ICML*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q. V.; Hinton, G. E.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*.

Shi, H.; and Wang, H. 2024. A Unified Approach to Domain Incremental Learning with Memory: Theory and Algorithm. *NeurIPS*.

Shi, H.; Xu, Z.; Wang, H.; Qin, W.; Wang, W.; Wang, Y.; and Wang, H. 2024. Continual Learning of Large Language Models: A Comprehensive Survey. *CoRR*.

Smith, J. S.; Tian, J.; Halbe, S.; Hsu, Y.-C.; and Kira, Z. 2023. A Closer Look at Rehearsal-Free Continual Learning. arXiv:2203.17269.

Snell, C.; Klein, D.; and Zhong, R. 2022. Learning by distilling context. *arXiv preprint arXiv:2209.15189*.

Song, C.; Han, X.; Zeng, Z.; Li, K.; Chen, C.; Liu, Z.; Sun, M.; and Yang, T. 2023. ConPET: Continual Parameter-Efficient Tuning for Large Language Models. *CoRR*.

Sprechmann, P.; Jayakumar, S. M.; Rae, J. W.; Pritzel, A.; Badia, A. P.; Uria, B.; Vinyals, O.; Hassabis, D.; Pascanu, R.; and Blundell, C. 2018. Memory-based Parameter Adaptation. In *ICLR*.

Su, J.; Ahmed, M. H. M.; Lu, Y.; Pan, S.; Bo, W.; and Liu, Y. 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. *NeurIPS*.

Tack, J.; Kim, J.; Mitchell, E.; Shin, J.; Teh, Y. W.; and Schwarz, J. R. 2024. Online Adaptation of Language Models with a Memory of Amortized Contexts. *CoRR*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Canton-Ferrer, C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023a. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*.

Wang, J.; Jatowt, A.; and Yoshikawa, M. 2022. ArchivalQA: A Large-scale Benchmark Dataset for Open Domain Question Answering over Historical News Collections. arXiv:2109.03438.

Wang, L.; Zhang, X.; Li, Q.; Zhu, J.; and Zhong, Y. 2022. CoSCL: Cooperation of Small Continual Learners is Stronger Than a Big One. In *ECCV*.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2024a. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Trans. Pattern Anal. Mach. Intell.*

Wang, W.; Dong, L.; Cheng, H.; Liu, X.; Yan, X.; Gao, J.; and Wei, F. 2023. Augmenting Language Models with Long-Term Memory. *arXiv preprint arXiv:2306.07174*.

Wang, Y.; Chen, X.; Shang, J.; and McAuley, J. J. 2024b. MEMORYLLM: Towards Self-Updatable Large Language Models. *CoRR*.

Weston, J.; Chopra, S.; and Bordes, A. 2015. Memory networks. In *ICLR*.

Wingate, D.; Shoeybi, M.; and Sorensen, T. 2022. Prompt Compression and Contrastive Conditioning for Controllability and Toxicity Reduction in Language Models. In *EMNLP Findings*.

Wortsman, M.; Ilharco, G.; Gadre, S. Y.; Roelofs, R.; Lopes, R. G.; Morcos, A. S.; Namkoong, H.; Farhadi, A.; Carmon, Y.; Kornblith, S.; and Schmidt, L. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*.

Wu, T.; Luo, L.; Li, Y.; Pan, S.; Vu, T.; and Haffari, G. 2024. Continual Learning for Large Language Models: A Survey. *CoRR*.

Xu, J.; Ton, J.; Kim, H.; Kosiorek, A. R.; and Teh, Y. W. 2020. MetaFun: Meta-Learning with Iterative Functional Updates. In *ICML*.

Yang, H.; Lin, Z.; Wang, W.; Wu, H.; Li, Z.; Tang, B.; Wei, W.; Wang, J.; Tang, Z.; Song, S.; et al. 2024. Memory^3: Language Modeling with Explicit Memory. *arXiv preprint arXiv:2407.01178*.

Yao, Y.; Wang, P.; Tian, B.; Cheng, S.; Li, Z.; Deng, S.; Chen, H.; and Zhang, N. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. In *EMNLP*.

Zheng, J.; Qiu, S.; Shi, C.; and Ma, Q. 2024. Towards Lifelong Learning of Large Language Models: A Survey. *CoRR*.

Zhong, W.; Guo, L.; Gao, Q.; and Wang, Y. 2023. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *arXiv preprint arXiv:2305.10250*.

Zhong, Z.; Lei, T.; and Chen, D. 2022. Training Language Models with Memory Augmentation. In *EMNLP*.