Learn to Memorize: Scalable Continual Learning in Semiparametric Models with Mixture-of-Neighbors Induction Memory

Guangyue Peng[♡*], **Tao Ge**[♠†], **Wen Luo**[♡], **Wei Li**[♡], **Houfeng Wang**[♡†]
[♡] State Key Laboratory for Multimedia Information Processing,
School of Computer Science, Peking University
[♠] Microsoft
{agy,wanghf}@pku.edu.cn,taoge@microsoft.com
llvvvv22222@gmail.com,weili22@stu.pku.edu.cn

Abstract

Semiparametric language models (LMs) have shown promise in various Natural Language Processing (NLP) tasks. However, they utilize non-parametric memory as static storage, which lacks learning capability and remains disconnected from the internal information flow of the parametric models, limiting scalability and efficiency. Based on recent interpretability theories of LMs, we reconceptualize the non-parametric memory represented by kNN-LM as a learnable Mixture-of-Neighbors Induction Memory (MoNIM), which synergizes the induction capabilities of attention heads with the memorization strength of feed-forward networks (FFN). By integrating into the model's information flow, MoNIM functions as an FFN-like bypass layer within the Transformer architecture, enabling effective learning of new knowledge. Extensive experiments demonstrate that MoNIM is a retentive and scalable continual learner in both data- and model-wise, enhancing the scalability and continual learning performance of semiparametric LMs.[1]

1 Introduction

Semiparametric language models (LMs) have drawn increasing attention (Guu et al., 2020; Yogatama et al., 2021; Ram et al., 2023) for their photographic memorization capabilities and in-domain accuracy. Combining a parameterized neural model with an extensible non-parametric memory, they are skillful at aiding prediction with memorization.

However, current semiparametric LMs are unsuitable for our fast-changing world because of inefficient memory management strategies (He et al., 2021a). They typically record all training data in the static memory, and the search for useful information then relies on additional modules or tunable hyperparameters. A lack of learning ability hinders memory compression, a process crucial for efficient learning in LMs (Delétang et al., 2024), and separates the memory component from the information flow of the model. As a result, these models experience linear growth of memory usage and search time as data or model dimensions increase. This inefficiency becomes especially impractical for large language models (LLMs), which deal with huge volumes of training data.

In this paper, we deal with two questions about semiparametric LMs: *Why they are powerful but inefficient* and *can we build an efficient memory strategy*? Enlightened by research interpreting the learning abilities of LLMs (Elhage et al., 2021; Geva et al., 2021; Olsson et al., 2022), we propose that the non-parametric memory, specifically kNN-LM (Khandelwal et al., 2020) inherently possesses abilities akin to the induction heads in Multi-Headed Self-Attention (MHSA) layers (Olsson et al., 2022). Perfect memorization brings perfect local next-token induction and good performance, but deficiencies in global visions prevent the memory from efficient reasoning.

To promote local advantages while avoiding global weaknesses, we build a learnable Mixture-of-Neighbor Induction Memory (MoNIM) based on the components of concept promotion in Feed-Forward Networks (FFN) (Geva et al., 2022). As an FFN-like bypass layer, MoNIM can select and absorb knowledge with a gradually smaller memory footprint, demonstrating its consistency with gradient descent in parametric models, where the impact of new information is lessened with more data or larger model sizes (Kaplan et al., 2020). Consequently, MoNIM's memory grows sub-linearly with the enhancement of the model's capabilities, resulting in a scalable continual learner free of training. In continual learning scenarios, MoNIM performs comparably to vanilla models while consuming only half the memory space.

[*] This work was done during the author's internship at Microsoft Research Asia.
[†] Corresponding author
[1] Code is publicly available at https://github.com/viniferagy/MoNIM.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28517–28531

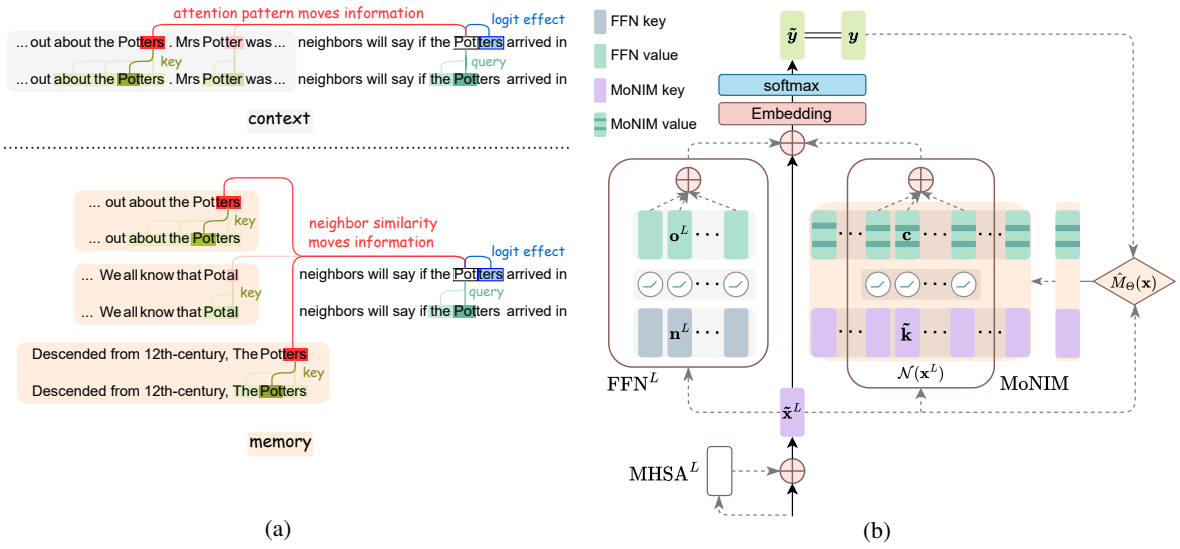

Figure 1: The learning mechanism of MoNIM. **(a)** The analogy between induction heads (up) and MoNIM (down). The shades of lines indicate allocated attention scores. While induction heads assimilate related information in context, MoNIM gathers similar samples memorized in learning history. **(b)** MoNIM as a learnable module. MoNIM shares the same input, working flow, and output with the final FFN layer. While FFN utilizes parametric keys $\mathbf{n}^L$ to match the query $\tilde{\mathbf{x}}^L$ for the promotion of learned concepts encoded in parametric values $\mathbf{o}^L$ (green), MoNIM promotes concept mixtures $\mathbf{c}$ (green with stripes) that are embedded in neighboring memorized values. The learning process of MoNIM is controlled by a compressor $\hat{M}_\Theta(\mathbf{x})$ optimized for the model's loss function.

Our contributions can be summarized as follows:

- We introduce Mixture-of-Neighbor Induction Memory (MoNIM), a learnable memory in semiparametric continual learning settings that functions as an FFN-like bypass layer.
- MoNIM achieves its prowess by integrating the inductive capabilities of Multi-Head Self-Attention (MHSA) with the memorization functions of Feed-Forward Networks (FFN).
- Extensive experiments in language modeling and downstream tasks show that MoNIM effectively compresses seen information and is both data- and model-wise scalable, thus suitable for continual learning over streaming data with semiparametric LMs.

2 MoNIM: Mixture-of-Neighbor Induction Memory

2.1 Preliminaries: kNN-LM

Formally, we use $\Theta = (\theta, \mathcal{M})$ to denote a semiparametric LM, where θ stands for the parametric LM and $\mathcal{M}$ for the non-parametric memory.

As a representative, kNN-LM (Khandelwal et al., 2020) enhances the prediction of θ by leveraging the information of k-nearest neighbors in $\mathcal{M}$. Given a leftward context $\mathbf{x} = (x_1, \ldots, x_t)$, kNN-LM uses the hidden states in the final position before an FFN layer $\ell \in \{1, .., L\}$ as the contextualized representation $\widetilde{\mathbf{x}}^\ell \in \mathbb{R}^d$, and computes its next word y's probability as follows:

$$P(y|\mathbf{x};\Theta) = f(\underbrace{P(y|\mathbf{x};\theta)}_{\text{Model}}, \underbrace{P(y|\widetilde{\mathbf{x}}^\ell;\mathcal{M})}_{\text{Memory}}, \lambda) \quad (1)$$

where f represents the interpolation function to weigh the predictions of the model and memory by λ. kNN-LMs construct $\mathcal{M}$ by the training set $\mathcal{D}$ as a key-value lookup, with an entry for each token in the training set x_t (as value) and the representation of its context $\widetilde{\mathbf{x}_{<\mathbf{t}}}^\ell$ (as key):

$$\mathcal{M} = \{(\widetilde{\mathbf{x}_{<\mathbf{t}}}^\ell \rightarrow x_t) | \mathrm{x} \in \mathcal{D}\}$$

During inference, we first use $\tilde{\mathbf{x}}^\ell$ as a query to retrieve k nearest neighbors from the memory $\mathcal{M}$:

$$\mathcal{N}(\tilde{\mathbf{x}}^\ell) = \{(\tilde{\mathbf{k}} \rightarrow \tilde{y})_i | i = 1, 2, \ldots, k\} \subseteq \mathcal{M}$$

Then, we obtain the prediction from $\mathcal{M}$ by computing the weighted sum of retrieved targets:

$$P(y|\tilde{\mathbf{x}}^\ell;\mathcal{M}) \propto \sum_{\mathcal{N}(\tilde{\mathbf{x}}^\ell)} \mathbb{1}_{\tilde{y}=y} \exp(-d(\tilde{\mathbf{k}}, \tilde{\mathbf{x}}^\ell))$$

here $d(.,.)$ denotes the semantic similarity. Khandelwal et al. (2021) proved that $\ell = L$ produces the best retrieval quality.

2.2 Induction Memory

Recent interpretability studies (Olsson et al., 2022; Wang et al., 2023) have shown that induction heads, namely attention heads that implement a simple algorithm to complete sequences in the form of [A][B] ... [A] -> [B], might constitute the fundamental abilities for in-context learning in LLMs. Attention heads exhibit two typical properties: prefix matching, to attend to the tokens with similar context; and copying, to increase the logit of the output corresponding to the attended-to tokens (Bansal et al., 2023).

Figure 1(a) demonstrates the reasoning mechanisms between induction heads and the kNN memory. It is clear to observe their closeness: as induction heads assimilate related information scattered in the contexts, kNN memory gathers similar information from $\mathcal{N}(\tilde{\mathbf{x}})$ memorized in history. The memory encodes the prefix into the key vector, uses it to match the query, and copies the memorized value to the position to be predicted as the function of induction heads. From this perspective, kNN memory is an induction buffer, considering all training data as neighbors, thus the name Mixture-of-Neighbor Induction Memory (MoNIM).

While induction heads serve as basic components in the early layers, transferring context information in the model to implement more complex global reasoning pathways, MoNIM, as a highly localized source of information, only provides the memorized labels. The capability to induce the most similar tokens also suggests its inability to perform efficient prediction. To make the best use of MoNIM's local precision, we move on to develop a new conceptual architecture of MoNIM.

2.3 MoNIM for Local Concept Mixture Promotion

According to previous work (Sukhbaatar et al., 2015; Geva et al., 2021), FFN layers function as key-value memory, and the value vectors can be projected to vocabulary space to represent comprehensible concepts such as food or movie characters (Geva et al., 2022). An FFN update $\mathcal{F}(\tilde{\mathbf{x}})$ thus can be interpreted as successive concept promotions towards the connotation of the target token:

$$\mathcal{F}^\ell(\tilde{\mathbf{x}}^\ell) = \sum_{i=1}^{d_\mathcal{F}} f(\tilde{\mathbf{x}}^\ell \cdot \mathbf{n}_i^\ell)\mathbf{o}_i^\ell = \sum_{i=1}^{d_\mathcal{F}} m_i^\ell \mathbf{o}_i^\ell$$

$\mathbf{n}_i^\ell, \mathbf{o}_i^\ell$ is the i-th column of $W_{in}^{\mathrm{T}}, W_{out} \in \mathbb{R}^{d_\mathcal{F} \times d}$ in $\mathcal{F}$, m_i^ℓ represents the weight of $\mathbf{o}_i^\ell$, where the model stores its understanding of concepts. Similarly, for the hidden state before an MHSA layer $\hat{\mathbf{x}}$, given the attention pattern $\mathbf{a}^\ell \in \mathbb{R}^T$ for a context of length T and corresponding $\mathbf{v}_j^\ell$, the j-th column of $W_\mathrm{V} \in \mathbb{R}^{T \times d_\mathcal{A}}$, an MHSA update $\mathcal{A}(\hat{\mathbf{x}})$ is the linear combination of vectors of the output matrix $W_\mathrm{O} \in \mathbb{R}^{d_\mathcal{A} \times d}$.

$$\mathcal{A}^\ell(\hat{\mathbf{x}}^\ell) = \sum_{j=1}^{d_\mathcal{A}} (\mathbf{a}^\ell \cdot \mathbf{v}_j^\ell)\mathbf{o}_j^\ell = \sum_{i=j}^{d_\mathcal{A}} m_j^\ell \mathbf{o}_j^\ell$$

Since Geva et al. (2022) has proved that the layer normalization (LN) is almost linear and does not affect the linear properties of MHSA and FFN outputs, we assert that the final prediction before the unembedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ can be decomposed to the reweighted sum of information gained in MHSA and FFN layers, that

$$\tilde{\mathbf{x}}^\ell = \sum_{i=1}^{\ell \cdot (d_\mathcal{F} + d_\mathcal{A})} \tilde{m}_i^{<\ell} \mathbf{o}_i^{<\ell}$$

$$\tilde{\boldsymbol{y}} = \mathrm{softmax}(E\tilde{\mathbf{x}}^L)$$

The prediction of $\tilde{\boldsymbol{y}}$ is determined by the mixture of concepts $\boldsymbol{o}^{<L}$ in FFN. We define the best **local mixture of concepts** $\mathbf{c}$ which outputs the golden prediction $\boldsymbol{y}^*$:

$$\mathbf{c} = \sum_{i=1}^{L \cdot (d_\mathcal{F} + d_\mathcal{A})} \tilde{m}_i^{*<L} \mathbf{o}_i^{<L}$$

$$\boldsymbol{y}^* = \mathrm{softmax}(E\mathbf{c})$$

We can infer that in MoNIM $\mathcal{M}$, the functioning form of memory entries is $(\tilde{\mathbf{k}} \rightarrow \mathbf{c})$, while the actual memory entries $(\tilde{\mathbf{k}} \rightarrow \tilde{y})$ can be explained as economical and practicable appearance. MoNIM update can be viewed as a collection of sub-updates, each corresponding to a local mixture of concepts in the MoNIM output:

$$\mathcal{M}(\tilde{\mathbf{x}}^L) = \sum_{\mathcal{N}(\tilde{\mathbf{x}}^L)} f_\mathcal{M}(\tilde{\mathbf{x}}^L \cdot \tilde{\mathbf{k}}) \cdot \mathbf{c} = \sum_{\mathcal{N}(\tilde{\mathbf{x}}^L)} \tilde{m}\mathbf{c}$$

$(\tilde{\mathbf{k}} \rightarrow \tilde{y})$ integrates into the information stream of Transformer for sake of E to transform to $(\tilde{\mathbf{k}} \rightarrow \mathbf{c})$ to operate. Through E, MoNIM transforms into the general form as in Eq 1.

$$\begin{aligned}\log P(y|\mathbf{x};\Theta) &= \log \frac{\exp((1-\lambda)\boldsymbol{e}_y\mathbf{o}^L + \lambda \boldsymbol{e}_y \mathcal{M}(\tilde{\mathbf{x}}^L))}{Z(E((1-\lambda)\mathbf{o}^L + \lambda\mathcal{M}(\tilde{\mathbf{x}}^L)))} \\ &\propto \log \frac{\exp((1-\lambda)\boldsymbol{e}_y\mathbf{o}^L)}{Z(E\mathbf{o}^L)} \frac{\exp(\lambda \boldsymbol{e}_y \mathcal{M}(\tilde{\mathbf{x}}^L))}{Z(E\mathcal{M}(\tilde{\mathbf{x}}^L))} \\ &= (1-\lambda)\log P(y|\mathbf{x};\theta) + \lambda \log P(y|\tilde{\mathbf{x}};\mathcal{M})\end{aligned}$$

where $\boldsymbol{e}_y$ is the embedding of y, and $Z(\cdot)$ is the constant softmax normalization factor. Figure 1(b) demonstrates the equivalent working flows between $(\tilde{\mathbf{k}} \rightarrow \mathbf{c})$ in MoNIM and $(\mathbf{n}^\ell, \mathbf{o}^\ell)$ in $\mathcal{F}^L$, formalizing MoNIM as an FFN-like bypass layer. From this perspective, MoNIM focuses on promoting local mixtures of concepts induced by the memorized neighbors, augmenting the induction abilities of LMs in the final layers. The complicated reasoning tasks are left to the parametric model which handles them better.

3 MoNIM is a Scalable Continual Learner

MoNIM's blend of memorization and induction suggests its potential to adapt to new knowledge, namely it can *learn* as induction heads and compress worthless data for its induction task (Jiang et al., 2024; Delétang et al., 2024). When the model confidently relies on global reasoning to tackle problems, MoNIM should step back to avoid impacting the model's performance. However, when the model lacks information for a decision, MoNIM should step in, promoting memorized local concept mixtures to help the model generate a more probable prediction.

3.1 Learning Strategies of MoNIM

We propose learning strategies for MoNIM that adopt cross-entropy, the optimization objectives of gradient descent.[2]

In gradient descent, the greater the cross entropy, the greater the gradient and the impact of data on parameter updates. For MoNIM learning, we indicate the same effect of data on memory capacity. The memory effect of a sample $M_\Theta(\mathbf{x})$ can be expressed by its loss on model Θ:

$$M_\Theta(\mathbf{x}) \propto \log P(x_t|\mathbf{x_{<t}};\Theta)$$

To compress the data, rather than assign weights to indicate the importance of data points, we transform the weighted update into a "full-or-none" compressor $\hat{M}_\Theta(\mathbf{x})$, namely only updates that weigh above a threshold δ will be saved into memory. Through this approximate method, we compress the unimportant part of data to take up no space and prove that the compressed part of data has very little effect on results.

$$\hat{M}_\Theta(\mathbf{x}) = \begin{cases} 1 & \text{if } \log P(x_t|\mathbf{x_{<t}};\Theta) < \delta \\ 0 & \text{else} \end{cases} \quad (2)$$

$$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathbb{1}_{\hat{M}_\Theta(\mathbf{x})}(\widetilde{\mathbf{x_{<t}}} \rightarrow x_t)\} \quad (3)$$

3.2 Adaptive MoNIM weight

Instead of using a fixed threshold (δ in Eq 2), we propose to use an adaptive memorization threshold (AMT) to enhance the effect of MoNIM:[3]

$$\delta_{\text{ada}} = \frac{\delta}{\max_y \log \frac{P(y|\mathbf{x_{<t}};\Theta)}{P(x_t|\mathbf{x_{<t}};\Theta)} + 0.5}$$

The best form of threshold is not the focus of this paper, however, we found that AMT-like types of threshold boost the experimental results. The intuition of AMT is straightforward: if $\max_y \log P(y|\mathbf{x_{<t}};\Theta) == \log P(x_t|\mathbf{x_{<t}};\Theta)$, then the memorization margin $\delta_{\text{ada}} \leftarrow 2\delta$ (δ is the base threshold), meaning we can relax the threshold to 2δ since x_t is already the top-1 prediction and thus not urgent to be memorized. On the contrary, if $\max_y \log P(y|\mathbf{x_{<t}};\Theta) >> \log P(x_t|\mathbf{x_{<t}};\Theta)$, then $\delta_{\text{ada}} < \delta$, indicating we should aggressively memorize this sample because of the large gap between it and the top-1 prediction.

AMT is simple yet effective in practice. It allows us to skip many samples with the top rank, substantially reducing the memory size with marginal generation quality loss; moreover, it alleviates overfitting top-ranked samples, playing a similar role as label smoothing to avoid overconfident predictions.

Since MoNIM updates, unlike FFN layers (Geva et al., 2022), always promote concepts rather than eliminate or run shortcuts, if extracted neighbors are so thin and scattered that there is no reliable concept to promote, its weight λ should be pushed down accordingly. Inspired by previous studies (He et al., 2021a; Drozdov et al., 2022), we train a simple calibrator to inform the reliability of MoNIM with three categories of features: distribution information of the parametric LM, lexical information of the training data, and density information of MoNIM. Following He et al. (2021a), we use a 4-layer MLP network, optimized on a small subset of the validation set.[4]

3.3 Scalability of MoNIM

Unlike the space-inefficient kNN-LM, MoNIM's learning capability allows it to compress and reduce memory demand throughout the learning process.

[2]Other possible strategies are discussed in Appendix C.1.

[3]The ablation of AMT is placed in Appendix C.4.

[4]The detailed implementation of the calibrator is placed in Appendix A.3, and the ablation study in Appendix C.3.

We designed experiments in continual learning settings showing that MoNIM can keep compressing when updating. Further, we reveal that the features of compression are consistent with those of updates of model parameters and lead MoNIM to scalability: **(i) data scalability**: In parametric models, as learning progresses, the impact of data on parameter updates diminishes; similarly, in MoNIM, as it continues to learn, the influence of data on memory capacity diminishes, meaning less new information needs to be memorized. **(ii) model scalability**: As parametric models grow in size, the impact of data on parameter updates decreases; likewise, in MoNIM, the impact of data on memory capacity also diminishes with model growth.

4 Experiments

4.1 Experimental Setting

We use the news from December 2019 in the News Crawl corpus (NC-19Dec) as pilot data, and apply **Newscrawl-20H1** (NC-20H1), the articles during the first half of 2020 in the News Crawl, as our streaming data for continual learning (CL). We randomly select 1% data per day as the validation and test set, and the rest 98% articles as the training set. Table 1 shows the statistics of NC-20H1. We continually learn the streaming data in chronological order and update the search index[5] every day.

In addition, we construct **WikiEvent-20H1** (WE-20H1), a Wikipedia event dataset[6] describing real-world events during 20H1, for testing our approach in domains other than news. WE-20H1 contains, on average 10 Wikipedia articles per month with ∼100k tokens in total.

We use GPT-2 (Radford et al., 2019) as the backbone LM to study CL over 20H1's streaming data. We experiment with the GPT-2 small (S, 123M), medium (M, 355M), and large (L, 774M) variants[7], and GPT-2 small is assumed to be the default size unless otherwise specified. All the experiments are implemented using the Fairseq (Ott et al., 2019) toolkit and run on 1 NVIDIA V100 GPU.

We define the memorization rate (MemRate) as the percentage of key-value pairs stored in memory compared to the *FullMem* baseline. MemRate is utilized to measure the memory efficiency and scalability of our method. It comes that for NC-19Dec, when $\delta = -1.5$, MoNIM can achieve comparable performance to *FullMem* with ∼60% MemRate. Thus, we set $\delta = -1.5$[8] throughout our following experiments.

After CL, we conducted extensive experiments in both language modeling and downstream tasks to estimate MoNIM's performance and scalability. MoNIM is compared with the following baselines:

- Full memorization (*FullMem*): Conventional memorization policy that memorizes every token in the training set.
- Random memorization (*RandMem*): Randomly memorize data equal to MoNIM's initial MemRate (60%). We conduct three runs with random seeds and choose the best as the baseline.

	Daily		Monthly		Total	
	#Train	#Dev/Test	#Train	#Dev/Test	#Train	#Dev/Test
Articles	4.4K	46	133K	1.3K	796K	8.2K
Tokens	2.4M	24.7K	73.2M	741K	439M	4.5M

Table 1: Statistics of NC-20H1.

4.2 Results of language modeling

4.2.1 Data scalability

Table 2 compares between MoNIM and *FullMem*, *RandMem* for CL with NC-20H1, showing substantial improvements for the bare GPT-2 small. Among them, MoNIM achieves comparable (even slightly better) language modeling performance to *FullMem* but with only 50% MemRate, and largely outperforms *RandMem* (60% MemRate), demonstrating that MoNIM is a cost-effective memorization policy for CL.

Methods	PPL (↓)	MemRate (↓)
Bare GPT-2 small	24.1	0%
+*FullMem*	9.0	100%
+*RandMem*	15.0	60%
+MoNIM	**8.6**	**50%**

Table 2: Perplexity (PPL) on the test set of NC-20H1 by different memorization methods for CL over NC-20H1.

We look into the results by tracking MoNIM's monthly memorization throughout CL. The sublinear growth trend of MoNIM observed in Figure 2(a) indicates that the compression rate of MoNIM gradually increases as it learns over time because

[5]Implementation of search is included in Appendix A.2.
[6]An example event article is 2020 Caribbean earthquake.
[7]The detailed configurations are in Appendix A.1.
[8]We explored the effect of different choices of δ on performance and memorization in Appendix C.2.

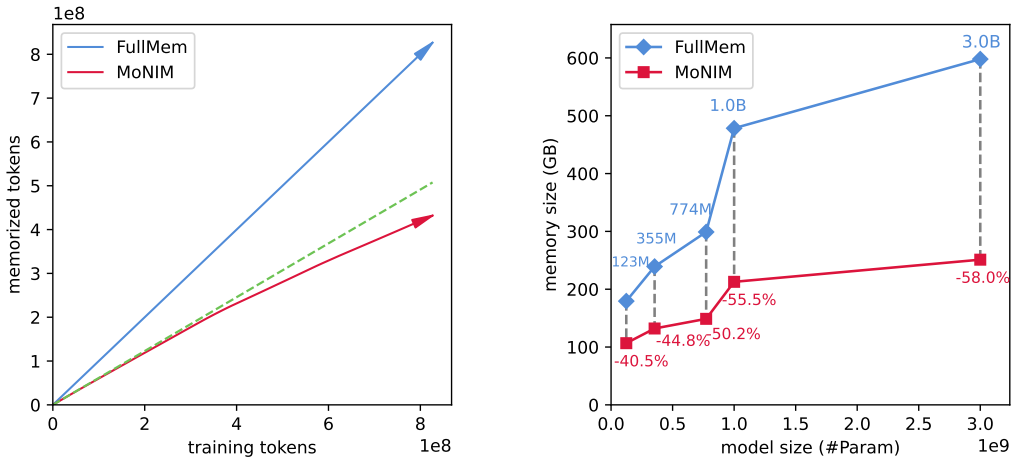

Figure 2: **(a)** The memory growth of the conventional full memorization (*FullMem*) policy and our proposed MoNIM policy, whose performance is controlled to be comparable with that of *FullMem*. Not only does MoNIM significantly reduce its memory but also the growth of its memory will become slower than when it starts (i.e., the green dashed line), as it continuously learns; **(b)** The growth trend of memory size (GB) with the increase of the model size (from 123M to 3B). The red numbers below the MoNIM's line indicate the memory consumption reduced by MoNIM, showing that MoNIM's effect is more remarkable in larger LMs.

Model	NC-20July (same domain)		WE-20H1 (different domain)	
	MemRate	PPL	MemRate	PPL
GPT-2 small	60%	8.8	62%	29.0
+CL w/ NC-20H1	**44%**	**8.2**	**57%**	**27.5**

Table 3: Comparison of memorization rate and perplexity for new CL data – NC-20July and WE-20H1 before and after CL with NC-20H1.

Model	Strategy	PPL (↓)	MemRate (↓)
S(123M)	*FullMem*	9.0	100%
	MoNIM	8.6	50%
M(355M)	*FullMem*	7.0	100%
	MoNIM	7.2	46%
L(774M)	*FullMem*	6.2	100%
	MoNIM	6.3	40%

Table 4: MoNIM's model-wise scalability for GPT-2 models of different sizes.

the model becomes increasingly knowledgeable and skips more training cases that it already knows.

We further confirm MoNIM's data-wise scalability by comparing the MemRates of two additional CL datasets — news data from July 2020 in the News Crawl (NC-20July) and WE-20H1 — before and after CL with NC-20H1. Following CL on NC-20H1, we utilize MoNIM (equipped with the memory acquired from NC-20H1) to continually process NC-20July and WE-20H1. As shown in Table 3, MoNIM's MemRates for both datasets are significantly lower than those of the models without CL, while improving performance. This reduction in MemRates can be attributed to the prior CL, which enables the model to avoid memorizing numerous instances learned previously, without compromising performance.

4.2.2 Model scalability

Table 4 shows the model size's effect on MoNIM. In the premise of comparable results to *FullMem*, the MoNIM's effect becomes more significant as the parametric model becomes larger: its overall MemRate drops from 50% (GPT-2 small) to 40% (GPT-2 large). As we assumed, a larger LM tends to skip more training cases than a smaller LM. The reduced MemRate demonstrates the model-wise scalability of MoNIM.

To test the generalizability of scalable memory on larger LMs, we choose Meta's Llama-3.2-1B and 3B versions (Dubey et al., 2024) for a brief evaluation of MoNIM's performance[9]. As shown in Figure 2(b), the total MemRate reduces from GPT2-small's 59.5% (123M) to Llama-3.2-3B's 42.0% (3B). We assume that larger LMs have the potential to achieve even more negligible memory consumption, as long as MoNIM maintains both data- and model-scalability.

[9]Since Llama-3.2 was released on September 25, 2024, we extract news from the first week of October 2024 in the News Crawl corpus to implement this experiment.

Methods	Wiki-103	1 (Jan)	2 (Feb)	3 (Mar)	4 (Apr)	5 (May)	6 (Jun)
Bare GPT-2	29.1	24.3	24.0	24.1	24.0	24.7	23.8
+Fine-tune	33.4 (+2.9)	20.4 (+2.4)	18.7 (+2.3)	17.3 (+1.3)	17.3 (+0.2)	16.2 (+0.8)	15.4 (+0.0)
(best)	30.6	18.2	16.4	16.0	17.1	15.4	15.4
+RecAdam	34.5 (+2.7)	19.6 (+0.7)	18.3 (+0.7)	17.3 (+0.2)	17.1 (+0.3)	16.8 (+0.3)	16.9 (+0.0)
(best)	31.8	18.9	17.6	17.1	16.8	16.5	16.9
+MixReview	33.6 (+3.0)	19.9 (+1.8)	18.5 (+2.1)	17.3 (+1.4)	17.2 (+0.1)	15.8 (+0.4)	15.6 (+0.0)
(best)	30.6	18.1	16.4	15.9	17.1	15.4	15.6
+Greedy Merging	35.2 (+5.9)	15.3 (+6.3)	15.5 (+5.2)	16.0 (+3.1)	15.9 (+2.8)	15.8 (+1.8)	14.6 (+0.0)
(best)	29.3	9.0	10.3	12.9	13.1	14.4	14.6
+MoNIM	**29.9** (+0.5)	**9.4** (+0.1)	**7.6** (+0.2)	**7.8** (+0.5)	**6.9** (+0.1)	**9.5** (+0.0)	**8.8** (+0.0)
(best)	29.4	9.3	7.4	7.3	6.8	9.5	8.8

Table 5: PPL evaluated on the 7 test sets after CL over NC-20H1 between MoNIM and representative CL approaches. The numbers in the second row of each cell denote the best result achieved during the process of CL.

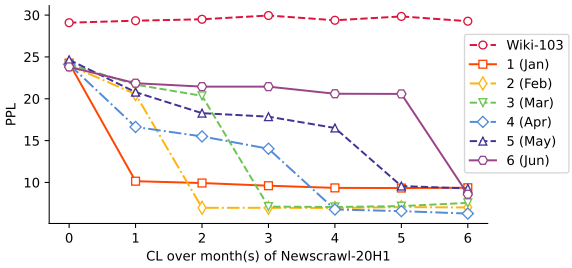

Figure 3: Language modeling performance on 7 test sets (Wiki-103 and 6 subsets of Newcrawl-20H1's test set by month) throughout CL via MoNIM over NC-20H1.

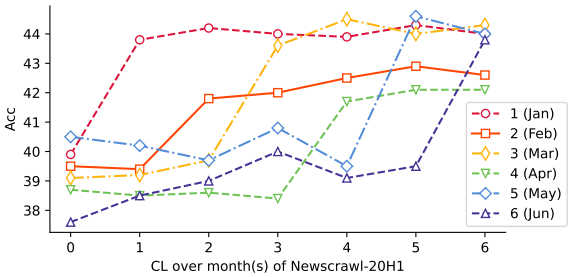

Figure 4: Next-word prediction accuracy on 6 subsets (by month) of WE-20H1 throughout CL over NC-20H1.

Model (MemRate)	Wiki-103	NC-20H1	ACL
Bare GPT-2 (0%)	29.1	24.1	40.5
+*FullMem* (100%)	30.1 → 31.0	9.0 → 10.0	**22.5**
+MoNIM (64%)	29.9 → **30.4**	8.6 → **9.2**	22.7

Table 6: CL performance over ACL papers after learning NC-20H1. The numbers beside each arrow indicate the PPL before/after studying the ACL papers.

Methods	Acc (↑)			MemRate (↓)		
	S	M	L	S	M	L
Bare GPT-2	39.5	41.2	44.6	0%	0%	0%
+*FullMem*	43.0	**45.5**	**48.9**	100%	100%	100%
+*RandMem*	40.3	41.3	45.5	60%	60%	60%
+MoNIM	**43.8**	45.4	48.5	**50%**	**46%**	**40%**

Table 7: Next-word prediction accuracy (Acc) on the test set (WE-20H1) and MemRate by different memorization methods and models after CL over NC-20H1.

4.2.3 Mitigation of forgetting

MoNIM's learning against forgetting performance is evaluated by tracking results on 7 test sets throughout CL over the NC-20H1: 6 are each month's held-out data in NC-20H1, and the other is the test set of Wiki-103 benchmark (Merity et al., 2017) which does not overlap with the NC-20H1 training data. According to Figure 3, MoNIM learns from the streaming data well, reflected by the sharp decrease of PPL on a test set after learning its corresponding month's training data. More importantly, it does not suffer much from the catastrophic forgetting issue (French, 1999). PPL scores of all 7 test sets do not significantly degrade throughout CL, since MoNIM will never erase previous memory or update the weights of LM.

The advantage can be better understood by comparing MoNIM with other CL baselines. We select two popular CL methods, RecAdam (Chen et al., 2020) and Mix-Review (He et al., 2021b), with Greedy Merging, the most effective data compression approach in He et al. (2021a).[10] As in Table 5, MoNIM not only achieves better results in learning from the new data but also suffers less from catastrophic forgetting than other CL approaches despite introducing additional memory.

[10]Details of CL baselines are included in Appendix B.

4.2.4 Domain adaptation

After CL over the news streaming data, which is not of great difference from the LM's pretraining dataset (i.e., openwebtext for GPT-2), we test MoNIM's CL performance over data in another domain – the ACL paper dataset (Lo et al., 2020) with 42K ACL papers. We hold off 80 papers (∼200K tokens) to construct the validation and test set, using the rest for training.[11] Table 6 shows that utilizing less memory (64% compared with *FullMem*), MoNIM consistently performs well in new data (40.5 → 22.7 in PPL) with less forgetting degradation. Although *FullMem* is also relatively resilient to catastrophic forgetting, it stores more noise from in-domain samples, which can degrade retrieval performance when applied to out-of-domain inputs.

4.3 Results of downstream tasks

4.3.1 Next-word prediction

For LLM, next-word prediction is the basic and the most straightforward end task, especially important for AI applications (e.g., input methods, Microsoft's text predictions and ChatGPT).

To align this task with our CL setting, we test the next-word prediction on the WE-20H1 to verify if CL over NC-20H1 can help write the current event articles in Wikipedia. As in language modeling, MoNIM consistently shows comparable performance with better scalability than FullMem as the model size increases (Table 7) and desirable results with little forgetting (Figure 4).

4.3.2 Closed-book question answer

We use REALTIME QA (Kasai et al., 2023), a multiple-choice question dataset about real-time events, as our second testbed of downstream tasks. To align with our streaming data, we use the subset of news during 20H1 and evaluate it in the closed-book Multiple Choice setting. As Kasai et al. (2023) suggests, we evaluate GPT-2 large in a zero-shot learning setting, in which GPT-2 small and medium are too weak to perform.

Table 8 shows the results in REALTIME QA. Compared with the bare GPT-2 large, CL through MoNIM over NC-20H1 substantially improves QA performance because it learns the world knowledge during 20H1 from the news stream to answer the questions. MoNIM again performs as well as FullMem with less memory footprint and outperforms RandMem. Moreover, little forgetting is consistently observed, as shown in Table 9.

[11] We split the training data into 4 batches for CL and update the index after finishing each batch.

Methods (MemRate)	Acc (↑)
Bare GPT-2 large (0%)	29.8
+*FullMem* (100%)	**36.3**
+*RandMem* (60%)	30.7
+MoNIM (**40%**)	36.2

Table 8: Accuracy on REALTIME QA of GPT-2 large with CL over NC-20H1 in zero-shot learning setting.

CL over \ Test on	1-2	3-4	5-6
1-2	36.0	29.3	29.5
1-4	37.2	35.8	30.6
1-6	37.0	36.5	35.5

Table 9: Accuracy on every two months of GPT-2 large of REALTIME QA throughout CL over NC-20H1.

5 Related Work

Retrieval-augmented LMs Retrieval components have been found beneficial for language tasks. Unlike the explicit storage methods (Guu et al., 2020; Borgeaud et al., 2022), semiparametric LMs like kNN-LM (Khandelwal et al., 2020) store implicit information as key-value pairs to assist prediction, without the need for retraining. As a powerful method to use the external data, many successive works of kNN-LM have been proposed (Zheng et al., 2021; Jin et al., 2022b; Trotta et al., 2022; Bhardwaj et al., 2023). Among them, He et al. (2021a) focuses on improving the efficiency, which has similar applications as our work, but we focus more on interpretable scalability in CL over streaming data with orthogonal contributions.

Interpretable LMs It remains obscure how Transformer manages to understand and generate natural languages. Among all the struggles to open the black box, mechanistic interpretability (Elhage et al., 2021; Olsson et al., 2022; Gurnee et al., 2023) investigates neurons and their connections in terms of circuits where information flows and transforms. Previous works have found many components that provide learning capabilities (Merullo et al., 2024; Zhang and Nanda, 2024). We transplant these interpretations that work for the parametric models to non-parametric memory, which has not been reasonably explained. Experiments prove the feasi-

bility of our conceptual framework of MoNIM.

Continual learning LMs Continual learning (CL) proposes to address the "new knowledge - catastrophic forgetting" dilemma (French, 1999). According to the taxology of Wang et al. (2024), our method deals with catastrophic forgetting problems based on replay-based methods (Sun et al., 2020; Qin et al., 2022; Scialom et al., 2022), despite that we managed to build a learnable replay memory. CL for LM is gaining traction (Ke et al., 2022; Razdaibiedina et al., 2023), and the closest works to us are Jang et al. (2022) and Jin et al. (2022a), which adapt LMs to emerging corpora across domains and timelines. However, we are the first to explore the memory to deal with non-parametric solutions for CL over streaming data.

6 Conclusion

We introduced Mixture-of-Neighbors Induction Memory (MoNIM), a novel conceptual framework that integrates dynamic induction memory into the Transformer architecture to interpret and enhance semiparametric LMs. Our experiments demonstrate that MoNIM not only offers a fresh perspective on non-parametric memory but also sets a new benchmark for scalable and efficient learning in LLMs, giving insights for the evolution of LLMs without the need for parameter adjustments.

Limitations

We construct the framework of MoNIM and thoroughly investigate its practicality and effectiveness as a representative of semiparametric LMs in continual learning. However, MoNIM only formalizes the interpretation of kNN-LM, while there are diverse models and implementations under semiparametric LMs. Specifically, besides auto-regressive models, auto-encoder models like T5 (Jang et al., 2022) also exhibit their potential for continual LM. Although we have observed that T5s are empirically capable of continual learning, the framework we constructed does not currently include them. In the future, we intend to extend to varied models and architectures and confirm the universal effectiveness of our framework.

Due to resource constraints, we tested our method on data within half a year (20H1) and on models up to 3B in size. Stretching the time series and increasing the model size is urgent for observing a more prominent and convincing build-up curve for a longer period.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62036001) and National Science and Technology Major Project (No. 2022ZD0116308) . The corresponding author is Houfeng Wang.

References

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. 2023. Rethinking the role of scale for in-context learning: An interpretability-based case study at 66 billion scale. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11833–11856, Toronto, Canada. Association for Computational Linguistics.

Rishabh Bhardwaj, George Polovets, and Monica Sunkara. 2023. Adaptation approaches for nearest neighbor language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1135–1146, Toronto, Canada. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. Language modeling is compression. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Andrew Drozdov, Shufan Wang, Razieh Rahimi, Andrew McCallum, Hamed Zamani, and Mohit Iyyer. 2022. You can't pick your neighbors, or can you?

when and how to rely on retrieval in the kNN-LM. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2997–3007, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021a. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021b. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1121–1133, Online. Association for Computational Linguistics.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2022a. Lifelong pretraining: Continually adapting language models to emerging corpora. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 1–16, virtual+Dublin. Association for Computational Linguistics.

Xuyang Jin, Tao Ge, and Furu Wei. 2022b. Plug and play knowledge distillation for kNN-LM with external logits. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 463–469, Online only. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv preprint*, abs/2001.08361.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A. Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime QA: what's the answer right now? In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. 2022. Continual training of language

models for few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10205–10216, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2024. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. ELLE: Efficient lifelong pre-training for emerging data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2789–2810, Dublin, Ireland. Association for Computational Linguistics.

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

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

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2020. LAMOL: language modeling for lifelong language learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Severino Trotta, Lucie Flek, and Charles Welch. 2022. Nearest neighbor language models for stylistic controllable generation. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 295–305, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.

Fred Zhang and Neel Nanda. 2024. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 368–374, Online. Association for Computational Linguistics.

A Experiment details

A.1 Model configuration

We list the model configurations of the GPT-2 models (as the parametric LMs) in our experiments in Table 10.

Model	Layer	Dim	#Param
GPT-2 small	12	768	123M
GPT-2 medium	24	1024	355M
GPT-2 large	36	1280	774M

Table 10: Model configurations of the GPT-2 models (as the parametric LMs) in our experiments.

A.2 Index building

We use the FAISS toolkit (Johnson et al., 2021) for index building and searching. At each update, we sampled 1M keys randomly from memory to train 4K cluster centroids, and then the whole keys in memory are added to the trained index, all quantized to 64-bytes.

A.3 NN calibrator training

We list the features we use in training the NN calibrator:

- Distribution information of the parametric LM
  - $\tilde{\mathbf{x}}$: contextualized representation of $\mathbf{x}$ by the parameterized LM
  - $conf(\mathbf{x})$: $\max_y P(y|\mathbf{x};\theta)$
  - $ent(\mathbf{x})$: entropy of $P(y|\mathbf{x};\theta)$
- Lexical information of the training data
  - $\log freq(\boldsymbol{x_{-1}})$: log of frequency of the last token in the context
  - $\log distinct(\mathbf{x_{-1}})$: log of the number of distinct values that succeed the last token in the context
- Density information of the external memory
  - $d(\tilde{\mathbf{k}}, \tilde{\mathbf{x}})$: L^2 distance (semantic similarity) between the query and the top-i retrieved neighbor, $i = 1, 2, \ldots, 10$.
  - $\log distinct(\tilde{y})$: log of the number of distinct values of the top-i retrieved values, $i = 1, 2, \ldots, 10$.

On each day during memorization, we extract from the validation set 10 articles and update them into the training set of the calibrator yesterday to obtain the training set of the calibrator today. The validation set of the calibrator is obtained as above, except for it only needs 5 articles each day. Because the training set increases slowly every day, we reduce the number of training epochs from 5 epochs to 1 epoch as time goes on, in case of overfitting.

Each feature is fed into a 1-layer LeakyReLU network to be transformed into hidden states of 128-dimension equally. Then all the hidden states are concatenated to a long vector and fed into a 4-layer MLP network to predict λ. We list the hyperparameters of the NN calibrator in Table 11.

Hyperparameters	Values
Layers	4
Dimension of hidden state	128
Learning rate	3e-4
Optimizer	Adam
Activation function	ReLU
Dropout	0.2

Table 11: Hyperparameters of the NN calibrator

A.4 Inference

During inference, we feed context into the parameterized LM, its contextualized representation into the memory index, and its three types of features into the calibrator. We search for top-1K nearest neighbors from 32 nearest cluster centroids using the memory index. The calibrator reweighs the distributions of the parameterized LM and the memory, and we use this calibrated distribution as the final output of our model.

B CL baselines

RecAdam (Chen et al., 2020) As a regularization-based method, RecAdam recalls previously acquired knowledge by retaining the pretraining object through frozen parameters, and it continually learns new information using a multi-task learning object. As the learning process moves forwards, the regularization is annealed to lessen the restriction.

Mix-Review (He et al., 2021b) Assuming that the pretraining corpus is obtainable, Mix-Review uses an empirical decreasing function to adjust the quantity of the pretraining corpus mixed in the continued training data. As the learning process moves forward, the quantity of the pretraining corpus tapers off to 0, resulting in the remaining training process being equivalent to fine-tuning.

Besides established CL baselines, those methods aimed at data efficiency were also considered to be adapted to CL settings, such as Greedy Merging which performs the best in datastore pruning in He et al. (2021a). However, while Greedy Merging can be generalized for CL by pruning and merging memory greedily every certain number of steps iteratively, this approach presents disastrous distribution shifts. If we merge new memory into the old, the new information distribution will continuously shift towards the old distribution, finally destroying the performance of new data; vice versa, the old distribution will shift towards the new one, causing the catastrophic forgetting problem. It turns out in Table 5 that Greedy Merging undergoes severe catastrophic forgetting in the first 3 months of NC-20H1.

We leave more dedicated and adapted CL approaches to be explored in the future.

C Analysis

C.1 Possible learning strategies for MoNIM

In addition to the intuitive cross-entropy loss, there are reasonable methods to measure and control the learning process. We also propose to rely on the intrinsic information content within the memory to assess the necessity of memorization.

Internal information based Since memorized keys can be projected to vocabulary space to analyze the information hidden in keys, we can calculate the internal distance from key to value, namely the KL-divergence from key-projected token distribution to the golden token distribution, which represents the amount of new information contained in the sample.

$$\hat{M}_\Theta(\mathbf{x}) = \begin{cases} 1 & \text{if } D_{\mathrm{KL}}(\boldsymbol{y} \,||\, E\mathbf{x}_{<\mathbf{t}}) < \delta \\ 0 & \text{else} \end{cases}$$

Methods	PPL (↓)	MemRate (↓)
Bare GPT-2 small	24.1	0%
+MoNIM(loss)	**8.6**	**50%**
+MoNIM(KL)	9.5	54%

Table 12: Perplexity (PPL) on the test set of NC-20H1 by different learning strategies for MoNIM over NC-20H1.

Methods	PPL (↓)	MemRate (↓)
Bare GPT-2 small	24.1	0%
+*FullMem*	9.0	100%
+MoNIM ($\delta = -1.0$)	**8.2**	54%
+MoNIM ($\delta = -1.5$)	8.6	50%
+MoNIM ($\delta = -2.0$)	9.9	**45%**

Table 13: MoNIM with different memorization threshold δ.

The preliminary results (Table 12) indicate the dominance of cross-entropy loss over KL divergence of internal information. Due to the resource limitation, we stick to the learning strategy using cross-entropy loss in the main experiments throughout the rest of the paper.

C.2 Performance *VS* Memorization rate

We have confirmed that MoNIM can achieve performance comparable to *FullMem* with a substantially reduced memorization rate when $\delta = -1.5$. Intuitively, if δ increases, more cases will be memorized and the performance will likely increase further; on the contrary, if δ decreases, more cases will be skipped, resulting in less memory but weaker performance. Table 13 confirms this intuition, demonstrating that we can obtain a trade-off between scalability and performance through the manipulation of δ.

C.3 NN calibrator

The NN calibrator plays an important role in calibrating the semiparametric LM's prediction by adapting λ at test time. Although such an adaptive method has been proven universally effective in semiparametric LMs by previous work, we re-

Methods	FullMem	RandMem	MoNIM
Constant λ	9.0	15.0	14.3
NN calibrator	**8.3** (-0.7)	12.5 (-2.5)	8.6 (**-5.7**)

Table 14: Perplexity results on the NC-20H1 test data with and without the NN calibrator for FullMem, RandMem, and MoNIM.

Features	PPL (↓)
All	8.6
-Density features	12.0
-Distribution features	9.9
-Lexical features	8.9

Table 15: The ablation study of features in the NN calibrator.

veal in Table 14 that it benefits MoNIM most, significantly more than *RandMem* and conventional *FullMem*, which only introduces marginal improvement as in previous work, demonstrating that calibration is more compatible with MoNIM.

We ablate the features of the NN calibrator to study their effects on the results. According to Table 15, all our proposed features contribute positively to the calibrator, among which the density information, especially the L^2 distance, is the most important one because it can directly reflect if the non-parametric memory has much relevant information given a test case, providing the most straightforward evidence to the calibrator.

C.4 Adaptive memorization threshold (AMT)

We test MoNIM's MemRate with/without AMT, since it serves as a measure of AMT's performance in handling memory overfitting. It reveals in Figure 5 that after adding AMT the downward trend is largely enhanced, while the performance is nearly untouched despite the reduced memory. We claim that the overfitting causes the predictions to favor the neighboring overconfident wrong answers, damaging the generalization of new data. Thus, the alleviation of overfitting is helpful with respect to both performance and scalability.

C.5 In-context learning

We test if the in-context learning capability of a language model is affected by CL through MoNIM. We present the in-context learning result in the REALTIME QA benchmark that the memory can benefit in Table 16, showing that MoNIM is not in

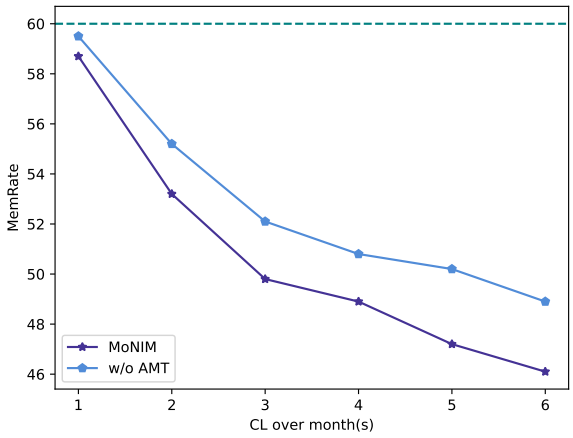

Figure 5: MoNIM's memorization rate with CL over months.

Methods (MemRate)	0-shot	1-shot	2-shot
Bare GPT-2 large (0%)	29.8	31.5	32.5
+*FullMem* (100%)	36.3	**36.9**	37.2
+*RandMem* (60%)	30.7	32.1	33.0
+MoNIM (40%)	**36.3**	36.7	**37.7**

Table 16: In-context learning accuracy of the GPT-2 large with CL over NC-20H1 on the REALTIME QA in 0-, 1- and 2-shot learning.

conflict with in-context learning and that the LM can still benefit from more examples shown in the context.

Also, we evaluate MoNIM's effect on in-context learning for general tasks collected by Shi et al. (2022) that cannot benefit from its memory. According to Table 17, despite no improvement observed, MoNIM does not affect the results in these tasks, demonstrating its robustness.

C.6 Time efficiency

We analyze the time consumption of MoNIM with vanilla kNN-LMs.

- Index Building: In this process, the whole computational overhead is equal to conducting a full forward pass over the training data to extract representations as keys, which is the same as a vanilla kNN-LM.
- Retrieval Process: We have measured that MoNIM's inference time is approximately 0.9–1.1x that of vanilla kNNs. For time-sensitive tasks, we have also explored a simple modification to accelerate inference without significantly hurting performance. We introduce a confidence threshold θ after the cali-

Methods (MemRate)	RTE	CB	RT	SST-2	CR	MR	HYP
Bare GPT-2 large (0%)	53.1	39.3	49.5	51.4	50.5	50.8	60.0
FullMem (100%)	50.3	35.5	49.0	49.8	48.6	47.5	60.0
FullMem (100%) w/ NN calibrator	52.8	41.1	49.5	51.8	50.8	50.9	60.0
MoNIM (40%)	53.1	41.1	49.5	51.5	50.8	50.0	60.0

Table 17: 0-shot learning accuracy of the GPT-2 large with CL over NC-20H1 on general NLP tasks.

brator of λ in Section 3.2, so that if $\lambda < \theta$, we simply skip the retrieval step and rely solely on the LM output. This is intuitive since a low λ indicates that memory contributes little useful information. In our experiments with GPT-2 small, when $\theta = 0.3$, the inference latency is reduced to 0.8x that of vanilla kNNs, while the PPL in Table 2 increases slightly from 8.6 to 8.9, while *FullMem* PPL is 9.0.