ELDER: Enhancing Lifelong Model Editing with Mixture-of-LoRA

Jiaang Li[1], **Quan Wang**[2], **Zhongnan Wang**[1], **Yongdong Zhang**[1], **Zhendong Mao**[1*]

[1] University of Science and Technology of China
[2]Beijing University of Posts and Telecommunications
jali@mail.ustc.edu.cn, wangquan@bupt.edu.cn {zhyd73, zdmao}@ustc.edu.cn

Abstract

Large language models (LLMs) require model editing to efficiently update specific knowledge within them and avoid factual errors. Most model editing methods are solely designed for single-time use and result in a significant forgetting effect in lifelong editing scenarios, where sequential edits are conducted over time. Previous approaches manage sequential edits by freezing original parameters and discretely allocating new parameters for each knowledge update. However, these methods lack robustness to minor input variations due to the discrete mapping between data and parameters. To overcome this challenge, we propose ELDER, a novel approach to create a continuous association between data and adapters. ELDER integrates multiple LoRAs through a router network and is trained to establish a smooth data-adapter association, thereby enhancing the edit robustness and generalization of semantically equivalent inputs. To ensure inputs containing the same knowledge will be processed by the same LoRAs, we design a novel loss to guide the model link LoRA allocations with edit knowledge. Furthermore, we propose a deferral mechanism to retain the original LLM capabilities post-edit. Extensive experiments on GPT-2 XL and LLaMA2-7B demonstrate that ELDER effectively edits models in the lifelong setting, outperforming eight baselines while exhibiting strong scalability and preserving LLMs' general abilities on downstream tasks.

Introduction

Large language models (LLMs) are renowned for their text understanding and generation capabilities (Brown et al. 2020; Achiam et al. 2023; Touvron et al. 2023; Radford et al. 2019). Despite their widespread use, LLMs often produce factual errors, including hallucinations and outdated information (Ji et al. 2023; Wang et al. 2023; Tam et al. 2023). Retraining or fine-tuning to update the model is expensive and time-consuming. Therefore, model editing techniques, which modify specific knowledge within LLMs with low resources, are gaining increasing attention (Yao et al. 2023; Mitchell et al. 2022; Meng et al. 2022). In practice, evolving world knowledge necessitates repeated model edits over time, which is known as lifelong model editing (Hartvigsen et al. 2024; Yu et al. 2024).

* Corresponding author: Zhendong Mao

The most intuitive way to implement lifelong editing is to perform model editing methods successively for multiple times. However, most model editing methods are designed for one-time use (Meng et al. 2022; Mitchell et al. 2022). Repeated using them causes LLMs to forget previous edits and pre-training data, significantly reducing their edit reliability and general ability on downstream tasks (Yao et al. 2023; Gu et al. 2024; Yang et al. 2024b; Gupta, Rao, and Anumanchipalli 2024; Lin et al. 2024; Huang et al. 2022). Recently, a class of representative methods has been developed specifically for lifelong model editing (Hartvigsen et al. 2024; Yu et al. 2024). These methods freeze the original LLM parameters and incorporate additional adapters to modify the model. They cluster the input data and assign a specific adapter to each cluster, maintaining discrete data-adapter mappings. This approach enables the model to manage different knowledge with independent parameters, preventing interference between different edits. As a result, it ensures high reliability after sequential edits and outperforms other techniques.

However, these key-value mapping methods exhibit poor robustness and struggle with semantically equivalent inputs (Tian et al. 2024; Lin et al. 2024). Due to the inherent discreteness of their data-adapter mappings, data points on opposite sides of a cluster boundary will map to entirely different adapters. Unfortunately, their clustering relies on manually set distance metrics and hyperparameters, resulting in inaccurate cluster boundaries. Semantically equivalent data with slight variations (e.g., rephrased sentences) could fall outside the appropriate cluster and are assigned incorrect adapters. Consequently, these methods are not robust and prone to errors with rephrased edit data.

To address the robustness issues in previous discrete mapping methods, we propose a novel approach to create an adaptive and continuous association between factual knowledge and adapter parameters, named ELDER (**E**nhancing **L**ifelong mo**D**el **E**diting with mixtu**R**e-of-LoRA). ELDER successively updates model knowledge by utilizing a router network to integrate multiple LoRAs (Hu et al. 2021), akin to the Mixture-of-Experts (MoE) structure (Jacobs et al. 1991; Jordan and Jacobs 1994). Unlike previous lifelong model editing methods, it adaptively generates LoRA weights through end-to-end learning instead of manually-set distance metrics, and produces a weighted combination

of top-k LoRA outputs. For different edits, ELDER dynamically adjusts LoRA weights to produce varied adapter allocations based on edit semantics, ensuring that semantically equivalent inputs are assigned similar allocations to generate consistent model responses. Thus, it ensures robust performance. Another advantage of ELDER lies in its scalability. Unlike discrete mapping methods, which require new parameters for each modification, ELDER manages knowledge modifications seamlessly by various LoRA combinations rather than independent adapters. Therefore, this approach avoids the need for additional parameters with each successive edit, allowing for scalability to longer editing sequences. To the best of our knowledge, this is the first work to employ a mixture-of-LoRA structure for model editing.

Beyond employing Mixture-of-LoRA, we also introduce two novel techniques designed to address specific challenges in the lifelong editing task. Specifically, we propose a guided loss function to align adapter allocation with edit knowledge and a deferral mechanism to retain general capabilities in post-edit models. Firstly, in the training data, semantically equivalent edits are preset with the same LoRA allocation. We design a novel training loss to guide the model in learning these allocations and establishing the association between LoRA allocation and data semantics, thus promoting the model to assign similar LoRA allocations to similar inputs during inference. Moreover, our deferral mechanism identifies whether an input requires editing based on its LoRA allocation. In this way, it concentrates on edit-related features while ignoring irrelevant details like input format to ensure accurate discrimination. For test inputs that differ significantly from edited samples, this mechanism deactivates the mixture-of-LoRA, retaining the model's original performance on downstream tasks while leveraging LoRAs for specific edits.

Through extensive experiments, we have demonstrated the effectiveness of ELDER. Our experiments are conducted on two popular LLMs, i.e., GPT2-XL(Radford et al. 2019) and LLaMA2-7B (Touvron et al. 2023), with two widely used model editing datasets, ZsRE (Levy et al. 2017) and COUNTERFACT (Meng et al. 2022). Results indicate that ELDER achieves better editing performance and enhances the robustness of rephrased edits by improving the editing generalization by over 10% higher than eight baselines. It is also superior in reliably maintaining previous edits. Furthermore, we show that ELDER retains most of post-editing LLM's abilities on downstream general tasks, significantly surpassing most existing methods.

Related Works

Model Editing Model editing aims to accurately and efficiently modify knowledge in deep neural networks(Sinitsin et al. 2019), especially language models(Yao et al. 2023). Meta-learning-based methods MEND(Mitchell et al. 2021) and KE(De Cao, Aziz, and Titov 2021) train an extra hypernetwork to learn changes in the base model. KN(Dai et al. 2022a) attributes the neuron that embodies the knowledge and updates these neurons. ROME(Meng et al. 2022) locates the edit area by casual analysis and modifies the entire weight matrix. SERAC(Mitchell et al. 2022) trains a scope classifier to identify inputs that need to be edited, then uses a counterfactual model to process these inputs separately from the original model.

However, the above methods only consider static edits, i.e., modifying the model a single time. Huang et al. (2022) proposed a sequential editing approach by adding one neuron for each edit sample. Nevertheless, this method relies on large sets of unrelated inputs and is slow due to the need to train neurons for each edit (Yao et al. 2023). Most similar works to ours are GRACE (Hartvigsen et al. 2024) and MELO(Yu et al. 2024), which discretely maps different adapters to successive edits.

Mixture-of-Experts (MoE) Jacobs et al. (1991); Jordan and Jacobs (1994) introduced MoE models to compute different examples with independent expert modules. Shazeer et al. (2016) extended this concept to large-scale language models with LSTMs. Advances such as GShard (Fedus, Zoph, and Shazeer 2021), Switch Transformer (Fedus, Zoph, and Shazeer 2021), and BASE Layer Lewis, Yao, and Ruder (2021) have refined routing input tokens to experts. Hash Layer (Roller et al. 2021) uses a pre-defined token-level hash table for token-to-expert assignment. StableMoE (Dai et al. 2022b) addresses routing fluctuation by training the router network first and then freezing it for further model training. In contrast, our approach guides the router network to learn a pre-set sample-to-adapter assignment, effectively handling unseen but equivalent inputs.

Mixture-of-LoRAs LoRA (Hu et al. 2021) uses low-rank matrices to update the LLMs, and has become a popular parameter-efficient fine-tuning method. Recently, researchers have been using combinations of multiple LoRAs for further benefits. Huang et al. (2023) proposed composing existing LoRA modules for generalization on new tasks. Zadouri et al. (2023) combined LoRA with token-level mixture-of-experts (Jacobs et al. 1991) and designed a new PEFT module. Dou et al. (2023) split LoRA experts into two separate groups to maintain world knowledge of LLMs during instruction tuning. MoRAL (Yang et al. 2024a) augments LoRA with MoE to achieve both multi-task and fine-tuning abilities.

Although sharing a similar base structure with previous mixture-of-LoRA research, our approach distinguishes itself from these works in following aspects. First, we focus on the task of editing factual knowledge within the model and demonstrating strong effectiveness, which previous works have not explored. Second, our routing scheme is tailored for model editing. We route the whole sequence altogether to ensure equal treatment of knowledge, unlike previous works that apply different experts to individual tokens. Moreover, we introduce novel techniques to enhance the adaptability of mixture-of-LoRA to the lifelong editing task, including the guided loss function and deferral mechanism.

Proposed Methods

Our proposed ELDER establishes a continuous and smooth connection between data and adapters. It contains multiple mixture-of-LoRA modules, which adaptively allocate LoRAs to each successive edit. Moreover, an auxiliary training

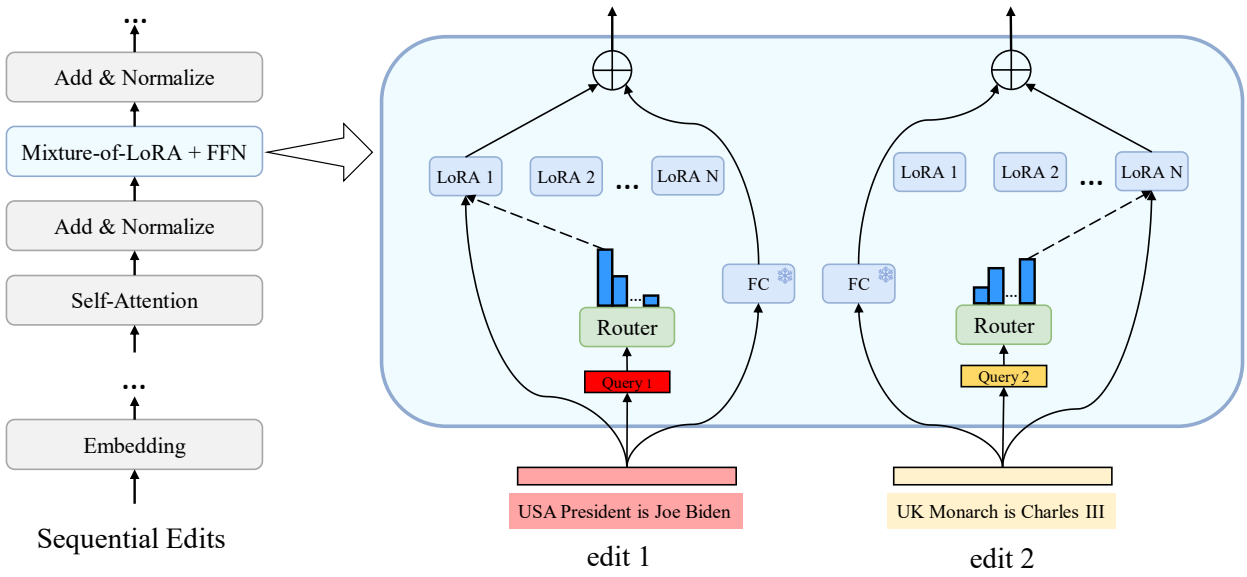

Figure 1: An illustration of processing two different edits with the mixture-of-LoRA module in ELDER. A mixture-of-LoRA module is applied to the FC layer at the FFN of the Transformer block. Each edit is routed to top-k LoRAs with the highest scores based on its query vector. This figure takes $k = 1$ as an example. The final results are summations of LoRA outputs and outputs of the original FC. Dotted lines denote multiplying LoRA outputs with corresponding weights. Training loss and deferral mechanism are omitted in this figure for simplicity.

loss is specially designed for lifelong model editing to assist the model in aligning LoRA allocations with the input knowledge. During inference, these LoRA allocations help to identify task inputs that do not require editing via a novel deferral mechanism tailored for the lifelong editing task. Such inputs are processed using the original LLM, thereby preserving the model's performance on general tasks.

Problem Formulation

The lifelong model editing task, as described by Hartvigsen et al. (2024), involves continuously editing an initial base LLM, f_{base}, without harming its overall abilities or negating the corrections made by prior edits. Let $D_{edit} = \{e_1, \ldots, e_n\}$ denote n successive edits applied to the model, where each edit $e_i = (x_i, y_i)$. The primary objective is to obtain a post-edit model f_n that correctly maps an input x to its corresponding prediction y, i.e., $f_n(x_i) = y_i$. Furthermore, f_n should be capable of predicting the equivalent neighbor $N(e_i)$ (Yao et al. 2023), such as rephrased sentences, ensuring $f_n(x_i') = y_i'$ for all $(x_i', y_i') \in N(e_i)$. Additionally, f_n must retain the performance on its original test data for practical usability(Hartvigsen et al. 2024). Given k example tasks $t_1, \ldots, t_k$ and their respective metrics $m_1, \ldots, m_k$, f_n should preserve f_{base}'s abilities across them, i.e., $m_j(t_j, f_n) = m_j(t_j, f_{base}), \forall j \in \{1, \ldots, k\}$.

Mixture-of-LoRA Structure

ELDER employs mixture-of-LoRA modules as the base structure. Each mixture-of-LoRA module comprises N LoRAs and a router network, as shown in Figure 1. The router network, implemented as a fully connected (FC) layer, takes a text sequence's query vector as input and routes the entire sequence to the top-k LoRAs selected from a pool of N LoRAs. Notably, we route all tokens in the same instance to one LoRA allocation based on the query vector, ensuring equal treatment of the entire knowledge. Following prior works (Hartvigsen et al. 2024; Yu et al. 2024), we define the query vector as the hidden representation of the last token in the input sequence, denoted as $\mathbf{x} \in \mathbb{R}^d$. The router network generates scores $\mathbf{s}(\mathbf{x})$ for each LoRA by normalizing the projection of $\mathbf{x}$ using a softmax distribution over the available N LoRAs at that layer, which are given by:

$$\mathbf{s}(\mathbf{x}) = \text{softmax}(\mathbf{W}_r \cdot \mathbf{x}),$$

where $\mathbf{W}_r \in \mathbb{R}^{N \times d}$ is a projection matrix. This process adaptively captures the semantic association between edits and their rephrases, assigning them similar LoRA scores.

We further apply top-k gating (Shazeer et al. 2016), selecting k LoRAs with the highest scores for routing. Each selected LoRA module generates a product matrix $\Delta\mathbf{W}_i = \mathbf{B}_i\mathbf{A}_i$, where $\mathbf{B}_i \in \mathbb{R}^{d \times r}$ and $\mathbf{A}_i \in \mathbb{R}^{r \times k}$ are two low-rank matrices in the i-th LoRA module. The updated matrix $\Delta\mathbf{W}$ is obtained by computing the linearly weighted combination of the selected LoRA matrices based on the score values,

$$\Delta\mathbf{W} = \sum_{i \in \mathcal{T}} s_i(\mathbf{x}) \cdot \Delta\mathbf{W}_i,$$

where s_i is the i-th element of $\mathbf{s}$, and $\mathcal{T}$ is the set of selected top-k indices.

We inject a mixture-of-LoRA module into the feed-forward network (FFN) of the Transformer block to edit the original FC layer, as shown in Figure 1. The forward pass of

a standard FC layer is formulated as

$$\mathbf{y} = \mathbf{W}_0\mathbf{v} + \mathbf{b},$$

where $\mathbf{b} \in \mathcal{R}^k$ is the bias, $\mathbf{W}_0 \in \mathbb{R}^{d\times k}$ is the weight matrix, and $\mathbf{y}$ and $\mathbf{v}$ are the output and input vectors, respectively. The modified FC computation involves multiplying both $\mathbf{W}_0$ and $\Delta\mathbf{W}$ with the input $\mathbf{v}$ and summing the result coordinate-wise. The modified forward pass is:

$$\mathbf{y} = \mathbf{W}_0\mathbf{v} + \Delta\mathbf{W}\mathbf{v} + \mathbf{b}.$$

Here, $\mathbf{W}_0$ is the FC layer's original weight matrix, which remains frozen during training. The modified FC computation is applied to all tokens in the given input sequence. Furthermore, as illustrated in the figure, ELDER can handle more edits through adapter combinations without requiring additional learnable parameters for each new edit. In contrast, previous discrete mapping methods increase the parameter count linearly for each new edit, implying scalability issues.

Guided Loss

We propose a novel training loss, guiding the model to allocate LoRAs based on knowledge embedded in each input. This training objective promotes the model to associate input semantics with various adapter allocations, thereby cultivating the adaptability to assign similar allocations to semantically equivalent inputs during inference. Specifically, we first pre-assign a LoRA allocation for each edit in the training data via random generation, ensuring that identically labeled edits receive the same allocation distinct from others. Subsequently, our proposed training loss guides the router network in learning to assign these allocations.

Formally, we guide the model with a new loss, L_{guide}, by maximizing the probability of selecting the pre-assigned LoRAs. Suppose the model employs mixture-of-LoRA to L layers. Each produced allocation $\mathcal{A}$ should contain $L \times k$ indices, indicating selected adapters from a total $L \times N$ available LoRAs. Therefore, the loss function takes the form of:

$$\mathcal{L}_{guide} = \sum_{i,j\in\mathcal{A}} -\log(s_{i,j}),$$
$$\mathcal{L}_{total} = \mathcal{L}_{model} + \lambda\mathcal{L}_{guide},$$

where $s_{i,j}$ is the score of the chosen j-th LoRA at the i-th layer. L_{total} is the total loss to be optimized, L_{model} is the original model loss, and λ is a hyperparameter. Intuitively, this guiding process encourages the model to generate a unique allocation for each piece of knowledge, thereby avoiding interference between different knowledge and improving parameter utilization.

Although previous MoE methods (Fedus, Zoph, and Shazeer 2021; Dai et al. 2022b) improve the token-routing scheme by balancing the usage load of parallel modules to achieve uniform distribution, they present a challenge in lifelong model editing. In this context, sequential edits occur sparsely, and all tokens in a single edit apply the same routing scheme rather than being routed separately, as they represent the same knowledge. Thus, each training batch has fewer samples, making it difficult to calculate the average usage load accurately. This disparity means that batch measurements do not accurately reflect real usage loads, rendering the optimization for a uniform distribution inapplicable to our task.

Algorithm 1: Inference with deferral mechanism.

Input: Test input x. Threshold ϵ
Input: Model M with layers $\{M_i\}_{i=1}^{t}$. First mixture-of-LoRA layer is M_{l_0}
Input: Allocation codes of all edits $\{\mathbf{c}_e^i\}_{i=1}^{n}$
Output: Model response R

$layer\_in \leftarrow x$
$flag \leftarrow 0$
for l in 1 to t **do**
  if $l = l_0 - 1$ **then**
    $layer\_out \leftarrow M_l(layer\_in, flag)$
    $\mathbf{c} \leftarrow GetAlloc(M, layer\_out)$ ▷ Get allocation code with all router networks.
    $dist \leftarrow \min_i HamDist(\mathbf{c}, \mathbf{c}_e^i)$ ▷ Find the nearest distance.
    if $dist < \epsilon$ **then**
      $flag \leftarrow 1$ ▷ Following layers will use mixture-of-LoRAs if available.
    else
      $flag \leftarrow 0$ ▷ Use original model parameters.
    end if
  else
    $layer\_out \leftarrow M_l(layer\_in, flag)$
  end if
end for
$R \leftarrow layer\_out$

Deferral Mechanism

During inference, we aim to preserve the original capabilities of LLMs on general tasks to ensure their practical application. To this end, we design a novel deferral mechanism to identify inputs that do not involve edited knowledge, referred to as task inputs. The proposed mechanism is based on the LoRA allocations, since they contain edit-related features within the input, excluding the impact of irrelevant details like input format as described in Section . After distinguishing these non-edited inputs, we deactivate the mixture-of-LoRA module and process them directly with the original model. This mechanism ensures that task inputs yield the same output as they would from the initial model.

The detailed steps of this deferral mechanism are outlined in Algorithm 1 and described next. First, test input is compared to all edited samples via the allocation code, a $L \times N$ boolean vector with $L \times k$ nonzero elements indicating the top-k LoRAs at each layer. Formally, for a test input with allocation $\mathcal{A}$, its allocation code $\mathbf{c}$ is defined as:

$$c_{i\times N+j} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{A}, \\ 0 & \text{if} \quad (i,j) \notin \mathcal{A}, \end{cases}$$

where (i, j) denotes the indices of the j-th LoRA at the i-th mixture-of-LoRA layer. $\mathbf{c}$ is represented by boolean values, and computed using all router networks before the first

mixture-of-LoRA (line 6) to discriminate model inputs in advance for efficient editing. The allocation codes $\{\mathbf{c}_e^i\}_{i=1}^n$ for all edits are precomputed and stored during the editing stage The nearest distance between $\mathbf{c}$ and all items in $\{\mathbf{c}_e^i\}_{i=1}^n$ is then calculated using the Hamming distance (line 7). This distance computation is highly efficient because all allocation codes are boolean, allowing the operations between them to be performed using simple bitwise operations. Inputs whose nearest distance exceeds a threshold ϵ are identified as task inputs and are processed with the preserved original parameters (line 9).

Experiments

Experimental Setup

Baselines We compare our proposed method, ELDER, with eight baselines. We first select two methods tailored for the lifelong editing setup. They preserve the original model and modify it with discrete data-adapter mappings. **GRACE** (Hartvigsen et al. 2024) uses representation vectors as adapters, writing them into the pre-trained model's latent space based on specific data samples. **MELO** (Yu et al. 2024) builds on the same foundation framework but uses LoRAs as adapters. Other baselines include **SERAC** (Mitchell et al. 2022), which additionally trains a scope classifier and a counterfactual model to alter the model behavior, and **T-Patcher** (Huang et al. 2022), which adjusts one neuron for each new edit. We also include **ROME** (Meng et al. 2022), a state-of-the-art one-time model editing method. It locates the edit area in GPTs via casual tracing and updates relevant weights. We further include **WilKE**(Hu et al. 2024), which modifies different layers for different edits. Beyond these model editing techniques, we compare ELDER with **FT-L** (Meng et al. 2022), which fine-tunes the layers identified by ROME, and with the standard **LoRA** (Hu et al. 2021).

Metrics We apply three metrics to evaluate both the editing performance and the preservation of post-edit model capability. These metrics align with those reported in previous works (Meng et al. 2022; Gu et al. 2024; Yao et al. 2023).

Reliability An edit $e_i = (x_i, y_i)$ is reliable if the post-edit model f_n generates the target answer correctly. We check how well f_n retains previous edits by reliability, which is measured as the average accuracy of the edit data:

$$\mathbb{E}_{e_i \in D_{edit}} \mathbb{I}\{\operatorname*{argmax}_y f_n(y|x_i) = y_i\}.$$

Generalization We check how well f_n generalizes to semantic equivalent data $N(e_i)$, e.g. rephrased sentences, by the average accuracy on these data:

$$\mathbb{E}_{(x_i', y_i') \in N(e_i), e_i \in D_{edit}} \mathbb{I}\{\operatorname*{argmax}_y f_n(y|x_i') = y_i'\}.$$

Test Retention on General Tasks We aim to evaluate the side effects of model editing on the capabilities of LLMs. Although a previous metric, *Locality*, is designed for this purpose, it is limited in scope, focusing on a narrow data range and simple QA task, which is insufficient to assess the full functionality of LLMs, which can lead to inflated results and fails to assess the full functionality of LLMs (Yang et al. 2024b; Gupta, Rao, and Anumanchipalli 2024). To achieve a more comprehensive evaluation, we extend the assessment to a broader range of tasks, following the approach of Gu et al. (2024). Specifically, we measure how well f_n retains its original performance during inference across k representative general tasks. The average result is then used to determine its retention on these tasks:

$$\frac{1}{k} \sum_{j=1}^{k} m_j(t_j, f_n).$$

Datasets We evaluate the reliability and generalization of ELDER on lifelong model editing with two widely used model editing datasets for training and evaluation. The first one, ZsRE (Levy et al. 2017), is a zero-shot relation extraction dataset that uses question rephrasing generated through back-translation as the rephrased edit data. The second dataset, COUNTERFACT (Meng et al. 2022), is a more challenging dataset. It comprises counterfactual statements initially receiving low factuality scores. We adopt both datasets to the lifelong model editing setting by extracting a sequence of 1000 editing samples with their rephrasings for our main experiments, following the methodologies outlined in (Hartvigsen et al. 2024) and (Yu et al. 2024). Further details are in the technical appendix.

To evaluate the test retention of post-edit LLMs on general tasks, we expand the evaluation task data used in previous studies (Hartvigsen et al. 2024; Yu et al. 2024) for a more comprehensive assessment of the post-edit LLMs' general abilities. This expansion is crucial because recent studies show that current model editing methods can degrade LLM performance on downstream tasks (Gu et al. 2024; Gupta, Rao, and Anumanchipalli 2024; Yang et al. 2024b), underscoring the importance of assessing their side effects with diverse datasets. Specifically, we employ a benchmark from (Gu et al. 2024), including eight diverse tasks: **Reasoning** on GSM8K (Cobbe et al. 2021), **Natural Language Inference** on RTE(Dagan, Glickman, and Magnini 2005), **Open-domain QA** on Natural Question(Kwiatkowski et al. 2019), **Closed-domain QA** on BoolQ(Clark et al. 2019), **Dialogue** on MuTual(Cui et al. 2020), **Summarization** on SAMSum (Cui et al. 2020), **Named Entity Recognition** on CoNLL03(Sang and De Meulder 2003), and **Sentiment Analysis** on SST2(Socher et al. 2013). Their respective metrics are in the technical appendix.

Implement Details We use two LLMs as base models: LLaMA2 (7B) (Touvron et al. 2023) and GPT2-XL (1.5B)(Radford et al. 2019). To evaluate our baselines, we use their original implementations and adapt them to our datasets and base models. For our proposed ELDER across all settings, the rank of LoRAs is set to 8, and the number of layers that apply mixture-of-LoRA is set to 6. The number of LoRAs per layer is set to 4, k is set to 2, and ϵ is set to 12. λ is set to $1e-2$. More details of training and hyperparameter tuning are available in the technical appendix.

Dataset	Model	Metric	Method									
			Base	FT-L	LoRA	ROME	WilKE	SERAC	T-Patcher	MELO	GRACE	ELDER
ZsRE	GPT2-XL	Reliability	0.00	48.23	30.22	48.40	53.32	96.09	77.29	70.81	96.80	**97.47**
		Generalization	0.00	47.61	19.39	47.20	47.10	53.03	67.74	66.41	0.00	**96.08**
	LLaMA2-7B	Reliability	0.25	32.21	10.70	77.60	61.84	83.14	62.94	64.57	89.48	**93.96**
		Generalization	0.37	28.96	7.31	74.53	51.29	60.38	48.37	42.93	0.46	**90.21**
CounterFact	GPT2-XL	Reliability	0.00	55.10	37.21	69.00	68.19	**100.00**	91.36	65.80	88.90	94.65
		Generalization	0.00	44.06	34.55	68.74	53.35	79.49	80.31	49.20	76.05	**91.26**
	LLaMA2-7B	Reliability	0.40	66.26	10.16	79.37	74.78	82.67	88.93	51.39	77.70	**95.07**
		Generalization	0.24	44.65	5.21	80.30	55.83	71.83	77.75	35.71	63.35	**90.79**

Table 1: Lifelong model editing performance of ELDER and baselines. 'Base' denotes the pre-editing models. All metrics shown are computed after all sequential edits, and higher is better. The best results are bolded.

	ZsRE		CounterFact		*Avg.*
	GPT2	LLaMA2	GPT2	LLaMA2	
Base	30.2	32.1	30.2	32.1	31.2
FT-L	0.7	25.2	0.7	7.8	8.6
LoRA	0.2	5.9	0.2	0.6	1.7
ROME	0.2	6.7	0.2	0.5	1.9
WilKE	2.1	7.8	1.2	5.3	4.1
SERAC	30.2	32.0	30.2	32.1	31.1
T-Patcher	2.7	1.1	0.1	0.9	1.2
MELO	29.6	29.9	**31.4**	31.9	30.7
GRACE	**30.3**	**32.7**	30.3	**32.2**	**31.4**
ELDER	30.1	32.3	30.5	31.5	31.1

Table 2: Test retention on general tasks of post-edit LLMs after successive edits from ZsRE and CounterFact. 'Base' represents the original model performance. The highest numbers are bolded.

Experimental Results

Main Results We compare the lifelong model editing performance of our proposed ELDER with recently advanced baselines, as shown in Table 1. Initially, the base models show low performance, indicating that the edit knowledge is not inherent to the model. After long sequences of edits, we observe that ELDER consistently outperforms these baselines, across all datasets and base models. FT-L and ROME, which are not designed for sequential editing, tend to forget previous edits. GRACE, which relies on discrete data-adapter mapping, is a strong baseline for reliably remembering previous edits but lacks robustness when handling semantically equivalent inputs, leading to poor generalization scores, especially on ZsRE, as noted in previous works (Tian et al. 2024; Lin et al. 2024). SERAC and T-Patcher also exhibit good lifelong editing reliability, but fall short in generalization. In contrast, ELDER excels in maintaining previous edits and robustly generalizes to their rephrases, demonstrating its superiority.

Table 2 reports the test retention on general tasks after sequential edits by each method. Detailed results for each task are provided in the technical appendix. ELDER effectively retains the LLM general abilities after lifelong editing, preserving the practical value of post-edit models. The results demonstrate that our deferral mechanism successfully identifies task inputs using edit-related information from LoRA allocations. On the other hand, FT-L, LoRA, ROME, WilKE and T-Patcher significantly degrade LLM performance due to repeated modifications of model parameters, consistent with previous studies (Gu et al. 2024; Gupta, Rao, and Anumanchipalli 2024).

	GPT2-XL		LLaMA2-7B	
	Speed	*#Param*	Speed	*#Param*
SERAC	\	181M	\	224M
GRACE	13.56 s/edits	6.4M	7.47 s/edits	4.1M
ELDER	1.82 s/edits	3.2M	2.12 s/edits	1.6M

Table 3: Editing efficiency, including editing speed and number of learnable parameters.

Editing Efficiency and Scalability Efficiency and scalability are critical for lifelong model editing methods, as they enable the model to manage multiple sequential edits with acceptable cost, when the edit sequence length continually increases. We compare the editing efficiency of ELDER against GRACE and SERAC in Table 3. Both methods exhibit strong performance in lifelong model editing and the retention of general abilities. We show the number of extra parameters and average editing time after 1000 sequential edits from the ZsRE dataset. The results clearly indicate that ELDER is more efficient than the baseline method. ELDER's time efficiency stems from its end-to-end design, which avoids the need to search for discrete mappings compared with GRACE. The editing speed of SERAC is not reported, because it requires additional training of the scope classifier and counterfactual model for each new edit. Moreover, we show the inference efficiency of ELDER in the technical appendix.

Furthermore, we extend the sequential edits from 1000 to 4000 to investigate the editing scalability. Figure 2 illustrates

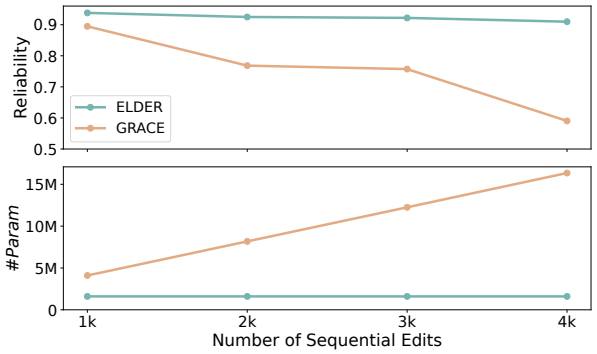

Figure 2: Editing scalability of GRACE and ELDER.

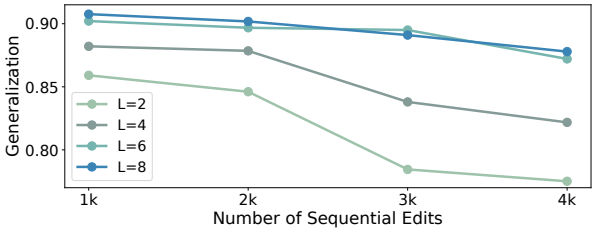

Figure 3: ELDER editing performance after varying numbers of edits with different parameter budgets.

the editing reliability and parameter amounts after different numbers of edits from ZsRE using LLaMA2-7B, averaged over five seeds. Notably, while GRACE requires increasing parameters with more edits, ELDER maintains high performance with a fixed parameter count, effectively accommodating more edits. This strong scalability of ELDER is attributed to its mixture-of-LoRA structure, which combines existing adapters to handle new edits rather than introducing independent parameters for each edit.

We also examine how ELDER scales with different parameter budgets. Figure 3 presents the results of varying L, which represents the number of layers utilizing the mixture-of-LoRA module, initially set to 6. As L increases, ELDER becomes more stable and exhibits better scalability, benefiting from the increased number of learnable parameters.

Ablation Experiments We conduct ablation experiments to evaluate the effect of our proposed guided loss. Two alternatives are considered: removing the guided loss during training or replacing it with the load balancing loss proposed by Fedus, Zoph, and Shazeer (2021). The latter improves the routing scheme by calculating the average LoRA usage load per training batch and optimizes it to be uniformly distributed. The results, shown in Table4, demonstrate that the guided loss achieves superior results and enhances the training of the mixture-of-LoRAs for lifelong model editing. In contrast, directly uniforming usage load per batch degrades the performance, since the small batch size in lifelong editing leads to the disparity between batch measurement and real usage loads, as described in the Method Section.

Model Analysis We further assess the behavior of ELDER by visualizing the LoRA allocation codes generated by the

	ZsRE		CounterFact	
	GPT2	LLaMA2	GPT2	LLaMA2
ELDER	96.08	90.21	91.26	90.79
w/o guide	92.49	87.19	87.57	85.20
w balancing	74.26	71.39	75.49	77.94

Table 4: Editing generalization while altering the auxiliary loss. *w/o guide* denotes removing guided loss from training. *w balancing* means balancing load with the loss proposed by Fedus, Zoph, and Shazeer (2021) instead.

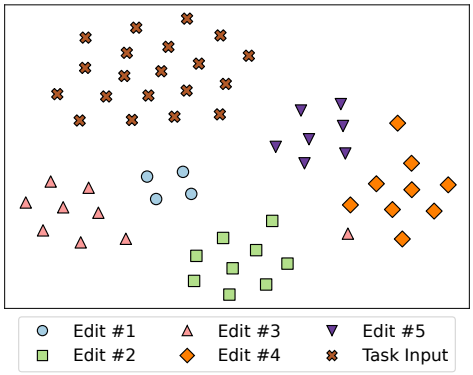

Figure 4: Visualization of LoRA allocation codes. Edit #1 to #5 denote five groups of semantically equivalent inputs, i.e., edits and their rephrases.

mixture-of-LoRA modules. We use ZsRE and LLaMA2-7B for the analysis. We randomly sample five groups of semantically equivalent inputs, including edits and their corresponding rephrases, along with twenty samples from the Natural Question dataset, chosen as task inputs unrelated to the edited knowledge. The allocation codes for these inputs are recorded and visualized using t-SNE, as shown in Figure 4. We observe that semantically equivalent inputs receive similar, though not identical, LoRA allocations. This similarity suggests that ELDER effectively captures the semantic associations between edits and their rephrases, showing robustness to minor input variations. Additionally, task inputs are separated from all edits, supporting the use of allocation codes for discrimination in our deferral mechanism.

Conclusion

In conclusion, this paper presents ELDER, a novel approach for lifelong model editing using a mixture-of-LoRA structure. The training process is guided by a novel loss function to align the LoRA allocation with edit knowledge, and a deferral mechanism is integrated to retain general abilities after editing. ELDER improves editing generalization by effectively assigning similar LoRA allocations to semantically equivalent inputs during inference. Extensive experiments demonstrate that this method significantly enhances editing performance with remarkable efficiency and scalability, while also preserving the original performance of LLMs on general tasks.

Acknowledgements

We would like to thank Rui-chen Zheng for the discussion on optimizing adapter allocation. This research is supported by Artificial Intelligence-National Science and Technology Major Project 2023ZD0121200 and National Natural Science Foundation of China under Grant No.62222212, No.62376033, No.62372006.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; 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.; 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 Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Clark, C.; Lee, K.; Chang, M.-W.; Kwiatkowski, T.; Collins, M.; and Toutanova, K. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Cui, L.; Wu, Y.; Liu, S.; Zhang, Y.; and Zhou, M. 2020. MuTual: A dataset for multi-turn dialogue reasoning. *arXiv preprint arXiv:2004.04494*.

Dagan, I.; Glickman, O.; and Magnini, B. 2005. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, 177–190. Springer.

Dai, D.; Dong, L.; Hao, Y.; Sui, Z.; Chang, B.; and Wei, F. 2022a. Knowledge Neurons in Pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 8493–8502.

Dai, D.; Dong, L.; Ma, S.; Zheng, B.; Sui, Z.; Chang, B.; and Wei, F. 2022b. StableMoE: Stable Routing Strategy for Mixture of Experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 7085–7095.

De Cao, N.; Aziz, W.; and Titov, I. 2021. Editing Factual Knowledge in Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6491–6506.

Dou, S.; Zhou, E.; Liu, Y.; Gao, S.; Zhao, J.; Shen, W.; Zhou, Y.; Xi, Z.; Wang, X.; Fan, X.; et al. 2023. Loramoe: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. *arXiv preprint arXiv:2312.09979*.

Fedus, W.; Zoph, B.; and Shazeer, N. 2021. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arXiv preprint arXiv:2101.03961*.

Gu, J.-C.; Xu, H.-X.; Ma, J.-Y.; Lu, P.; Ling, Z.-H.; Chang, K.-W.; and Peng, N. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Gupta, A.; Rao, A.; and Anumanchipalli, G. 2024. Model Editing at Scale leads to Gradual and Catastrophic Forgetting. *arXiv preprint arXiv:2401.07453*.

Hartvigsen, T.; Sankaranarayanan, S.; Palangi, H.; Kim, Y.; and Ghassemi, M. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Hu, C.; Cao, P.; Chen, Y.; Liu, K.; and Zhao, J. 2024. WilKE: Wise-Layer Knowledge Editor for Lifelong Knowledge Editing. *arXiv preprint arXiv:2402.10987*.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Huang, C.; Liu, Q.; Lin, B. Y.; Pang, T.; Du, C.; and Lin, M. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Huang, Z.; Shen, Y.; Zhang, X.; Zhou, J.; Rong, W.; and Xiong, Z. 2022. Transformer-Patcher: One Mistake Worth One Neuron. In *The Eleventh International Conference on Learning Representations*.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87.

Ji, Z.; Yu, T.; Xu, Y.; Lee, N.; Ishii, E.; and Fung, P. 2023. Towards Mitigating LLM Hallucination via Self Reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 1234–1245. Singapore: Association for Computational Linguistics.

Jordan, M. I.; and Jacobs, R. A. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2): 181–214.

Kwiatkowski, T.; Palomaki, J.; Redfield, O.; Collins, M.; Parikh, A.; Alberti, C.; Epstein, D.; Polosukhin, I.; Devlin, J.; Lee, K.; et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7: 453–466.

Levy, O.; Seo, M.; Choi, E.; and Zettlemoyer, L. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Lewis, M.; Yao, Z.; and Ruder, S. 2021. BASE Layers: Simplifying Training of Large, Sparse Models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Lin, Z.; Beigi, M.; Li, H.; Zhou, Y.; Zhang, Y.; Wang, Q.; Yin, W.; and Huang, L. 2024. Navigating the Dual Facets: A Comprehensive Evaluation of Sequential Memory Editing in Large Language Models. *arXiv preprint arXiv:2402.11122*.

Meng, K.; Bau, D.; Andonian, A.; and Belinkov, Y. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35: 17359–17372.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2021. Fast Model Editing at Scale. In *International Conference on Learning Representations*.

Mitchell, E.; Lin, C.; Bosselut, A.; Manning, C. D.; and Finn, C. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, 15817–15831. PMLR.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8): 9.

Roller, S.; Sukhbaatar, S.; Weston, J.; et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34: 17555–17566.

Sang, E. F.; and De Meulder, F. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2016. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*.

Sinitsin, A.; Plokhotnyuk, V.; Pyrkin, D.; Popov, S.; and Babenko, A. 2019. Editable Neural Networks. In *International Conference on Learning Representations*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Tam, D.; Mascarenhas, A.; Zhang, S.; Kwan, S.; Bansal, M.; and Raffel, C. 2023. Evaluating the Factual Consistency of Large Language Models Through News Summarization. In *Findings of the Association for Computational Linguistics: ACL 2023*, 3456–3467. Toronto, Canada: Association for Computational Linguistics.

Tian, B.; Cheng, S.; Liang, X.; Zhang, N.; Hu, Y.; Xue, K.; Gou, Y.; Chen, X.; and Chen, H. 2024. InstructEdit: Instruction-based Knowledge Editing for Large Language Models. *arXiv preprint arXiv:2402.16123*.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Goyal, N.; Balland, M.; Cucurull, G.; Guestrin, C.; Joulin, A.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Wang, X.; Yan, Y.; Huang, L.; Zheng, X.; and Huang, X. 2023. Hallucination Detection for Generative Large Language Models by Bayesian Sequential Estimation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 15361–15371. Singapore: Association for Computational Linguistics.

Yang, S.; Ali, M. A.; Wang, C.-L.; Hu, L.; and Wang, D. 2024a. MoRAL: MoE Augmented LoRA for LLMs' Lifelong Learning. arXiv:2402.11260.

Yang, W.; Sun, F.; Ma, X.; Liu, X.; Yin, D.; and Cheng, X. 2024b. The Butterfly Effect of Model Editing: Few Edits Can Trigger Large Language Models Collapse. *arXiv preprint arXiv:2402.09656*.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10222–10240.

Yu, L.; Chen, Q.; Zhou, J.; and He, L. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19449–19457.

Zadouri, T.; Üstün, A.; Ahmadian, A.; Ermis, B.; Locatelli, A.; and Hooker, S. 2023. Pushing Mixture of Experts to the Limit: Extremely Parameter Efficient MoE for Instruction Tuning. In *The Twelfth International Conference on Learning Representations*.