WISE: Rethinking the Knowledge Memory for Lifelong Model Editing of Large Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) need knowledge updates to meet the ever-growing world facts and correct the hallucinated responses, facilitating the methods of lifelong model editing. Where the updated knowledge resides in memories is a fundamental question for model editing. In this paper, we find that editing either long-term memory (direct model parameters) or working memory (non-parametric knowledge of neural network activations/representations) will result in an impossible triangle—reliability, generalization, and locality can not be realized together in the lifelong editing settings. For long-term memory, directly editing the parameters will cause conflicts with irrelevant pretrained knowledge or previous edits (poor reliability and locality). For working memory, retrieval-based activations can hardly make the model understand the edits and generalize (poor generalization). Therefore, we propose WISE to bridge the gap between memories. In WISE, we design a dual parametric memory scheme, which consists of the main memory for the pretrained knowledge and a side memory for the edited knowledge. We only edit the knowledge in the side memory and train a router to decide which memory to go through when given a query. For continual editing, we devise a knowledge-sharding mechanism where different sets of edits reside in distinct subspaces, and are subsequently merged into a shared memory without conflicts. Extensive experiments show that WISE can outperform previous model editing methods and overcome the impossible triangle of question answering, hallucination, and out-of-distribution settings across trending LLM architectures, e.g., GPT, LLaMA, and Mistral.

1 Introduction

Large language models (LLMs) show emergent intelligence when scaling the number of parameters, computes, and data (Kaplan et al., 2020; Sorscher et al., 2022; Alabdulmohsin et al., 2022; Zhao et al., 2023), which reveals the sparks of artificial general intelligence (Bubeck et al., 2023). However, when deployed, LLMs still make mistakes (Balachandran et al., 2022), generating responses with hallucinations (Ji et al., 2023), bias (Ferrara, 2023), and factual decays (De Cao et al., 2021). On the other hand, the world's knowledge is ever-growing, so the up-to-date knowledge is usually different from the one during LLMs' pretraining (Hartvigsen et al., 2023). Many such errors and emerging facts will arise sequentially in deployment, some of which have to be addressed timely and efficiently without waiting for retraining or finetuning (Huang et al., 2023; Lazaridou et al., 2021). Also, retraining or finetuning is often too computationally expensive (Touvron et al., 2023; Hartvigsen et al., 2023), which is not sustainable for lifelong growing knowledge. Therefore, *lifelong model editing* (Hartvigsen et al., 2023) was proposed to remedy the continual knowledge updates and injections for LLMs in a cheap and timely manner.

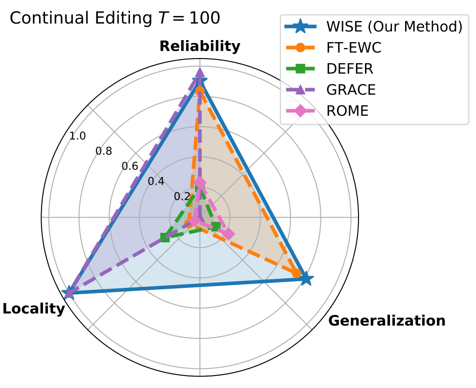


Figure 1: **Metric triangle among reliability, generalization, and locality.** ZsRE dataset, number of continual edits $T = 100$, LLaMA-2-7B.

An effective lifelong model editing approach should satisfy the following properties (Yao et al., 2023; Tan et al., 2023; Huang et al., 2023): **i) reliability**, the model can remember both current and previous edits after sequential editing; **ii) locality**, model editing will not influence inherent pretrained knowledge which is irrelevant to the edited knowledge; **iii) generalization**, the model is not just merely memorizing the query-target pairs; instead, it should generalize when given other forms of queries with the same knowledge. We compare existing model editing and continual learning methods on the three metrics in Figure 1 and find that *it seems to be an impossible triangle—reliability, generalization, and locality* can not be realized at the same time in the continual editing settings.

We find that where the updated knowledge resides in memories affects editing performances, and previous methods can be generally divided into editing either long-term memory, e.g., ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and FT-EWC (Finetuning with Elastic Weight Consolidation (Kirkpatrick et al., 2017), a continual learning method), or working memory, e.g., GRACE (Hartvigsen et al., 2023). Model editing of long-term memory refers to directly editing the model parameters, which contain generalizable parametric knowledge (Wang et al., 2024; Li et al., 2023a). However, editing long-term memory will cause conflicts with previous pretrained knowledge, resulting in poor locality (e.g., ROME and FT-EWC in Figure 1). Working memory refers to the non-parametric knowledge of neural network activations/representations by retrieval, and it does not change the network parameters (Li et al., 2023a); instead, it replaces the representations by retrieval at working (inference) time, like GRACE. GRACE's working memory shows promising results in reliability and locality, but in our experiments, it shows poor generalization since retrieval-based representations can hardly make the model understand the edits and generalize to different queries.

It reveals that long-term memory and working memory both have drawbacks for lifelong model editing, though there were some special memory designs for LLM architectures, like MemoryLLM (Wang et al., 2024), SPALM (Yogatama et al., 2021), and Memoria (Park and Bak, 2023), they change the architectures and cannot be directly applied for different LLMs. Intuitively, there is a gap between editing working and long-term memories, thus, in this paper, we study: *What is the better memory mechanism for lifelong model editing to break the impossible triangle?*

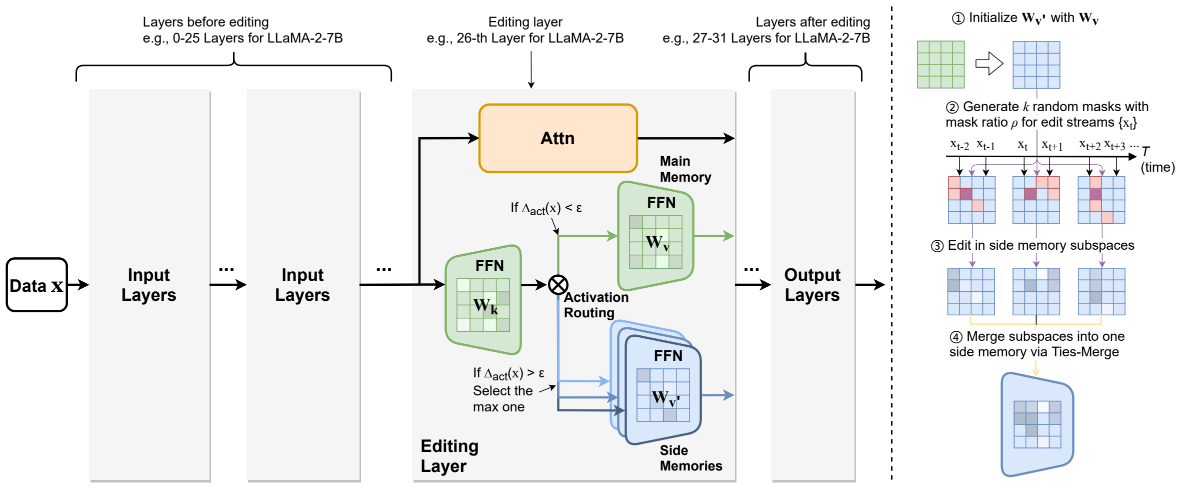
Human brains contain the left and right hemispheres, which have different divisions as studied in recognition science (Hellige, 2001; Ivry and Robertson, 1998), e.g., the left brain is typically associated with logical tasks while the right brain is more involved in intuitive processes. This inspires us to design **WISE**, which makes model editor *WISER* in *memories*. WISE contains a dual parametric memory mechanism for LLMs' editing: the main memory for the pretrained knowledge and a side memory for the edited knowledge, realizing both long-term memory's generalization and retrieval-based working memory's reliability and locality. The side memory is a form of mid-term memory. Our contributions are as follows:

- We identify the pitfalls of current model editing methods in lifelong settings, that is, the impossible triangle among—reliability, generalization, and locality. Behind the impossible triangle, we find there is a gap between editing long-term memory and working memory.
- We propose WISE, with a side parametric memory as the mid-term memory, realizing the advantages of both parametric long-term memory and retrieval-based working memory. We design memory routing, sharding, and merging modules in WISE, making WISE lead in continual knowledge editing, reaching the three metrics better simultaneously.
- Extensive experiments on GPT, LLaMA, and Mistral across QA, Hallucination, and out-of-distribution datasets validate the effectiveness of WISE for lifelong model editing.

2 Methodology

2.1 Preliminaries: Lifelong Model Editing

We focus on lifelong model editing problem (Hartvigsen et al., 2023; Huang et al., 2023), which can ensure hundreds or even thousands of sequential edits on LLMs to make the outputs of target queries align with human expectations while maintaining LLMs' previous knowledge and capability. Let $f_\Theta : \mathbb{X} \mapsto \mathbb{Y}$, parameterized by Θ, denote a model function mapping an input $\mathbf{x}$ to the prediction $f_\Theta(\mathbf{x})$. The initial model before editing is Θ_0, which is trained on a large corpus $\mathcal{D}_{\text{train}}$. When the LLM makes mistakes or requires injections of new knowledge, it needs model editing with a time-evolving editing dataset as $\mathcal{D}_{\text{edit}} = \{(\mathcal{X}_e, \mathcal{Y}_e)|(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_T, \mathbf{y}_T)\}$. At the time step T, a model editor (ME) takes the T-th edit and the LLM of the $T-1$ time step $f_{\Theta_{T-1}}$ as


Layers before editing
e.g., 0-25 Layers for LLaMA-2-7B
Editing layer
e.g., 26-th Layer for LLaMA-2-7B
Layers after editing
e.g., 27-31 Layers for LLaMA-2-7B
Data x
Input Layers
Input Layers
Attn
Main Memory
FFN
W_v
If $\Delta_{act}(x) < \epsilon$
FFN
W_k
Activation Routing
If $\Delta_{act}(x) > \epsilon$
Select the max one
FFN
$W_{v'}$
Side Memories
Editing Layer
Output Layers
① Initialize $W_{v'}$ with W_v
② Generate k random masks with mask ratio ρ for edit streams $\{x_t\}$
x_{t-2} x_{t-1} x_t x_{t+1} x_{t+2} x_{t+3} ... T (time)
③ Edit in side memory subspaces
④ Merge subspaces into one side memory via Ties-Merge

(a) Workflow Overview with Knowledge Routing   (b) Knowledge Sharding and Merging

Figure 2: **Overview of WISE.** Side memory (in blue) and main memory (in green) store edited and pretrained knowledge, respectively. Note: during inference, if WISE-Retrieve, the activation routing will retrieve and select one side memory with maximal activation score.

inputs and produce the revised LLM model f_{Θ_T} following the equation below:

$$f_{\Theta_T} = \mathrm{ME}(f_{\Theta_{T-1}}, \mathbf{x}_T, \mathbf{y}_T),$$

$$\text{s.t. } f_{\Theta_T}(\mathbf{x}) = \begin{cases} \mathbf{y}_e & \text{if } \mathbf{x} \in \mathcal{X}_e, \\ f_{\Theta_0}(\mathbf{x}) & \text{if } \mathbf{x} \notin \mathcal{X}_e. \end{cases} \quad (1)$$

Equation 1 describes that after model editing, the LLM should make the correct prediction on the current edit as $f_{\Theta_T}(\mathbf{x}_T) = \mathbf{y}_T$, while also preserving knowledge from past editing instances $(\mathbf{x}_{<T}, \mathbf{y}_{<T}) \in \mathcal{D}_{\text{edit}}$ as well as maintaining capability of f_{Θ_0} on the irrelevant data when $x \notin \mathcal{X}_e$, especially for general training corpus $\mathcal{D}_{\text{train}}$.

2.2 WISE: Side Memory with Knowledge Sharding, Merging, and Routing

As illustrated in Figure 2, WISE comprises two key components: 1) **Side Memory Design**: i) *side memory*: side memory is a memory container that is initialized as a copy of LLM's certain FFN layer, storing the stream of edits; ii) *memory routing mechanism*: similar to retrieval, a routing activation component is adopted to identify the scope of edits, routing the main (original) or side memories during inference; 2) **Knowledge Sharding and Merging**: i) *knowledge in random memory subspaces*: to make the edits in appropriate knowledge density and avoid forgetting, we shard the side memory into several random subspaces for editing; ii) *knowledge merging*: we leverage model merging techniques to merge different memory shards into one side memory without loss of knowledge.

2.2.1 Side Memory Design

Side memory in FFN's value matrix. Each layer in a Transformer contains a multi-head self-attention (MHA) mechanism and a feed-forward network (FFN), where the FFN constitutes two-thirds of the model parameters (Geva et al., 2021). The question of how Transformers retrieve and utilize stored knowledge remains unresolved (Meng et al., 2022; Niu et al., 2024), yet past works (Mitchell et al., 2022a; Geva et al., 2021) have demonstrated that editing the weights of the FFN is consistently more effective for LLMs. The FFN typically consists of key-value linear matrices: $\mathbf{W}_k, \mathbf{W}_v$, i.e., two multi-layer perceptron (MLP) layers. For the output of attention feature $\mathbf{f}$, the computation of the feed-forward network, omitting the bias terms, can be represented as:

$$\mathrm{FFN}(\mathbf{f}) = \mathbf{a} \cdot \mathbf{W}_v = \sigma(\mathbf{f}^\top \cdot \mathbf{W}_k) \cdot \mathbf{W}_v, \quad (2)$$

where σ is a nonlinear activation function (e.g. SwiGLU, GeLU), and $\mathbf{a}$ represents the activation values of the first MLP layer. Following previous works (Meng et al., 2022; Geva et al., 2021), we edit the value matrix $\mathbf{W}_v$ of the chosen FFN layer.

However, directly editing the value matrix may cause forgetting and side effects in a lifelong setting. Thus, we **copy a value matrix as side memory and edit the side memory instead of the original matrix (main memory)**. Specifically, the side memory is initialized with the copy of main memory as $\mathbf{W}_{v'} \leftarrow \mathbf{W}_v$. Given the side memory, the new output is expressed as $\mathrm{FFN}_s(\mathbf{f}) = \mathbf{a} \cdot \mathbf{W}_{v'}$. We will introduce how to update the side memory in Section 2.2.2.

Locating side memory's FFN layer. Transformer LLMs have been widely demonstrated to encode "lower-level" information (e.g., parts of speech) in earlier layers while processing more advanced linguistic phenomena like anaphora and coreference in later layers (Jawahar et al., 2019; Otmakhova et al., 2022; Tenney et al., 2019). Representations in later hidden layers propagate through residual connections without drastic changes (Chuang et al., 2024; Meng et al., 2022), enabling effective early exit in LLMs (Men et al., 2024; Schuster et al., 2022). Therefore, to minimize the side effects of editing and adjust advanced linguistic phenomena, we target mid-to-late layers (e.g. 27) for side memory. Further analysis of layer selection is provided in Section 4.

Routing between side memories and main memory. Similar to the retrieval-based methods (Hartvigsen et al., 2023; Mitchell et al., 2022b), during inference, it is needed to decide whether the main memory or the side memory is used. If a given query is within the scope of previous edits, the side memory is used; otherwise, the main memory. Inspired by (Huang et al., 2023), we introduce a routing activation indicator, given an input $\mathbf{x}$, it is formulated:

$$\Delta_{\text{act}}(\mathbf{x}) = \|\mathcal{A}(\mathbf{x}) \cdot (\mathbf{W}_{v'} - \mathbf{W}_v)\|_2, \qquad (3)$$

where $\mathcal{A}(\cdot) = \mathbf{a}$ is the activation of the side memory's corresponding FFN layer in Equation 2. We want the activation indicators of editing queries to be larger than the ones of irrelevant queries by a large margin, which is: $\min\{\Delta_{\text{act}}(\mathbf{x}_e) | \mathbf{x}_e \in \mathcal{D}_{\text{edit}}\} \gg \max\{\Delta_{\text{act}}(\mathbf{x}_i) | \mathbf{x}_i \in \mathcal{D}_{\text{irr}}\}$, where $\mathcal{D}_{\text{irr}}$ is the irrelevant dataset which includes $\mathcal{D}_{\text{train}}$. To achieve the above objective, we design a margin-based loss function during editing training, similar to contrastive (Chen et al., 2020) or triplet loss (Schroff et al., 2015). The margin-based loss function for routing activation is:

$$\begin{aligned} L_a = \min_{\mathbf{W}_{v'}} \{&\max(0, \Delta_{\text{act}}(\mathbf{x}_i) - \alpha) + \max(0, \beta - \\ &\Delta_{\text{act}}(\mathbf{x}_e)) + \max(0, \gamma - (\Delta_{\text{act}}(\mathbf{x}_e) - \Delta_{\text{act}}(\mathbf{x}_i)))\}, \\ &\text{s.t. } \mathbf{x}_e \in \mathcal{D}_{\text{edit}}, \mathbf{x}_i \in \mathcal{D}_{\text{irr}}. \end{aligned} \qquad (4)$$

Equation 4 aims that for all queries of irrelevant examples $\mathbf{x}_i$, the activation indicators should be less than threshold α, and for the edit samples $\mathbf{x}_e$, the activations should be larger than threshold β, with a certain distance γ between $\Delta_{\text{act}}(\mathbf{x}_e)$ and $\Delta_{\text{act}}(\mathbf{x}_i)$.

In the continual stream of incoming edits, the smallest activation indicator within the edits is updated and saved: $\epsilon = \min\{\Delta_{\text{act}}(\mathbf{x}_e) | \mathbf{x}_e \in \mathcal{D}_{\text{edit}}\}$. We aim to recognize the local scope of edits in this form. During inference, if the activation indicator of a new input is greater than ϵ, WISE will use the side memory $\mathbf{W}_{v'}$; otherwise, using the main memory $\mathbf{W}_v$. Thus, given the query $\mathbf{x}$, the output of the targeted FFN in Equation 2 is replaced by:

$$\text{FFN}_{\text{out}}(\mathbf{x}) = \begin{cases} \mathcal{A}(\mathbf{x}) \cdot \mathbf{W}_{v'} & \text{if } \Delta_{\text{act}}(\mathbf{x}) > \epsilon, \\ \mathcal{A}(\mathbf{x}) \cdot \mathbf{W}_v & \text{otherwise.} \end{cases} \qquad (5)$$

2.2.2 Knowledge Sharding and Merging

How to effectively and efficiently store continual knowledge in model parameters is important for lifelong editing. We introduce the notion of "*knowledge density*" (similar to knowledge capacity (Allen-Zhu and Li, 2024)) that describes how many pieces of knowledge are stored per parameter on average. There is an editing dilemma w.r.t. knowledge density: i) If only a few edits are made for full fine-tuning or editing the entire memory, the knowledge density is low, which may lead to overfitting. ii) If numerous edits are made within a common and limited parameter space, the knowledge density is high, resulting in conflicts within the edited knowledge and potentially causing catastrophic forgetting. To remedy this dilemma, we propose a knowledge sharding and merging mechanism to divide the edits into several shards, store them in different parameter subspaces, and merge them into a common side memory.

Knowledge in random memory subspaces. We edit the side memory $\mathbf{W}_{v'}$. We divide n edits into k shards, copy the side memory for k times, and generate k random gradient mask with mask ratio ρ for each copy of side memory. A random gradient mask $\mathbf{M}_i \in \{0,1\}^{|\mathbf{W}_{v'}|}, i \in [k]$ is a binary mask whose proportion of 1 is ρ (Li et al., 2024b). For edit shard $i, i \in [k]$, we edit the knowledge into the subspace $\mathbf{M}_i$ as follows:

$$\mathbf{W}_{v'}^i \leftarrow \mathbf{W}_{v'}^i - \eta(\mathbf{M}_i \odot \mathbf{g}_i(\mathbf{W}_{v'}^i)), \qquad (6)$$

where $\mathbf{W}_{v'}^i$ is the i-th copy of the side memory, η is the learning rate, $\mathbf{g}_i(\cdot)$ is the gradient of the i-th shard of edits, and the gradient is the autoregressive loss plus the routing activation loss L_a(Equation 4): $L_{\text{edit}} = -\log P_{W_{v'}}(\mathbf{y}_e|\mathbf{x}_e) + L_a$.

The random mask of gradients freezes the parameters intact when the elements are 0 and updates the weights when the elements are 1. It is superior to pruning because it does not harm the network performance while regularizing optimization in a subspace (Li et al., 2024b). In addition, the ρ subspace will have higher knowledge density when $k \cdot \rho < 1$, resulting in higher generalization (e.g., Figure 5). Also, different shards of edits have different random masks, and due to the (sub)orthogonality of random masks, different shards will not conflict with each other. Therefore, we can non-destructively merge the k copies of side memory into one.

Knowledge merging. We merge the k subspaces of side memory into one. Because we randomly generate the subspace masks, different random masks will have some overlapping and disjoint elements, following the theorem below:

Theorem 2.1 ***Subspace Overlap.*** *Generate k memory subspaces $\mathbf{W}_{v'}^{i}, i \in [k]$ by random mask with 1's ratio ρ, so each memory has $\rho \cdot |\mathbf{W}_{v'}|$ active trained parameters. For any two subspaces $\mathbf{W}_{v'}^{i}$ and $\mathbf{W}_{v'}^{j}$ $i \neq j; i, j \in [k]$, there are $\rho^2 \cdot |\mathbf{W}_{v'}|$ active parameters that are overlapped. For all k subspaces, there are $\rho^k \cdot |\mathbf{W}_{v'}|$ overlapped active parameters.*

The theorem shows that larger ρ will cause more overlap of subspace parameters, and the proof is in Appendix D. We find that this overlap is helpful in playing the role of "anchors" for knowledge merging (See Figure 5 and Appendix C.6). However, knowledge conflicts also exist in the overlapped parameters, so we leverage the recent task arithmetic model merging technique Ties-Merge (Yadav et al., 2023) to relieve the conflicts. First, we compute the edit weight shift vectors $\mathrm{T}_e = \{\tau_e^i = \mathbf{W}_{v'}^{i} - \mathbf{W}_v | i \in [k]\}$. Then, we use Ties-Merge to merge the edit vectors into one:

$$\mathbf{W}_{v'} \leftarrow \mathbf{W}_v + \mathrm{Ties}(\mathrm{T}_e; \mathbf{W}_v). \tag{7}$$

Ties-Merge consists of three steps: i) trim: trim the redundant parameters for each task vector; ii) elect the sign: elect the signs of each parameter; ii) disjoint merge: compute the disjoint mean for each parameter which has the same and correct signs (Yadav et al., 2023). By Ties-Merge, different subspaces are integrated into one with fewer conflicts. We study the effects of different merging techniques in Table 8 of Appendix C.3.

Routing and retrieving among several side memories. One single side memory has its limited knowledge capacity (Allen-Zhu and Li, 2024). For the lifelong editing stream, we can produce several side memories and retrieve them via activation score routing. We compute different activation indicator scores of side memories and retrieve the top-1 during inference. This design is named WISE-Retrieve, which enables a more challenging lifelong editing scenario. For WISE with only one side memory, it is notated as WISE-Merge. For most of the experiments, we use WISE-Merge by default, and we compare WISE-Retrieve in Table 11 and Figure 12.

The pseudo-code of our method can be found in Algorithms 1 and 2.

3 Experiments

Table 1: Dataset statistics for main results. *Locality Data* is the irrelevant data of the editing process. T is the number of samples. *Pre-edit* is the unedited model's performance on each dataset.

SETTING	EDITING DATA	T	Pre-edit (LLaMA/Mistral)	LOCALITY DATA
QA	ZsRE=	1,000	0.36/0.39 ACC	NQ (Kwiatkowski et al., 2019)
Halluc.	SelfCheckGPT	600	27.4/19.4 PPL	RedPajama (Computer, 2023)
OOD Gen.	Temporal	100	0.56 δ-ACC (GPT-J)	Pile (Gao et al., 2020)

3.1 Experimental Settings and Metrics

In the experiments, we compare the performance of different baselines and WISE in sequentially editing LLM models hundreds to thousands of times.

Datasets and Models. We choose trending autoregressive LLM models **LLaMA-2-7B** (Touvron et al., 2023), **Mistral-7B** (Jiang et al., 2023), and **GPT-J-6B** (Radford et al.; Wang and Komatsuzaki, 2021) for evaluation. The dataset details are in Table 1. Following (Hartvigsen et al., 2023), we evaluate WISE on the closed-book question-answering (QA) dataset **ZsRE** (Levy et al., 2017), and also evaluate its ability to correct **Hallucination** in SelfCheckGPT (Manakul et al., 2023). The **Temporal** dataset (Hewitt et al., 2024) is employed to test the out-of-distribution (OOD) generalization of editing. Since Temporal comprises emerging entities post-2019, we avoid using the latest LLMs in OOD experiments. Instead, we follow the original literature of the Temporal dataset (Hewitt et al., 2024) and adopt **GPT-J-6B** as the base model, which is pretrained on the Pile (Gao et al., 2020) with a cut-off in 2020. Implementation details can be found in Appendix B.

Table 2: **Main editing results for QA setting (ZsRE dataset).** T: Num Edits.

Method	**QA**															
	$T=1$				$T=10$				$T=100$				$T=1000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
	`LLaMA-2-7B`															
FT-L	0.57	0.52	0.96	0.68	0.48	0.48	0.76	0.57	0.30	0.27	0.23	0.27	0.19	0.16	0.03	0.13
FT-EWC	0.96	**0.95**	0.02	0.64	0.82	0.76	0.01	0.53	0.83	0.74	0.08	0.55	0.76	0.69	0.08	0.51
MEND	0.95	0.93	0.98	0.95	0.26	0.28	0.28	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ROME	0.85	0.80	0.99	0.88	0.64	0.62	0.75	0.67	0.23	0.22	0.04	0.16	0.01	0.01	0.00	0.01
MEMIT	0.84	0.81	0.99	0.88	0.58	0.58	0.85	0.67	0.02	0.02	0.02	0.02	0.04	0.04	0.02	0.03
MEMIT-MASS	0.84	0.81	0.99	0.88	0.75	0.72	0.97	0.81	0.76	0.68	0.85	0.76	0.69	0.65	0.62	0.65
DEFER	0.68	0.58	0.56	0.61	0.65	0.47	0.36	0.49	0.20	0.12	0.27	0.20	0.03	0.03	0.74	0.27
GRACE	**0.98**	0.08	**1.00**	0.69	**0.96**	0.00	**1.00**	0.65	**0.96**	0.00	**1.00**	0.65	**0.97**	0.08	**1.00**	0.68
WISE	**0.98**	0.92	**1.00**	**0.97**	0.94	**0.88**	**1.00**	**0.94**	0.90	**0.81**	**1.00**	**0.90**	0.77	**0.72**	**1.00**	**0.83**
	`Mistral-7B`															
FT-L	0.58	0.54	0.91	0.68	0.39	0.39	0.50	0.43	0.11	0.10	0.02	0.08	0.16	0.13	0.01	0.10
FT-EWC	**1.00**	**0.99**	0.01	0.67	0.84	0.78	0.02	0.55	0.82	0.72	0.09	0.54	0.76	0.69	0.09	0.51
MEND	0.94	0.93	0.98	0.95	0.01	0.01	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ROME	0.79	0.77	0.98	0.85	0.58	0.57	0.75	0.63	0.05	0.05	0.02	0.04	0.04	0.04	0.02	0.03
MEMIT	0.81	0.79	0.99	0.86	0.46	0.45	0.61	0.51	0.00	0.00	0.01	0.00	0.04	0.04	0.02	0.03
MEMIT-MASS	0.81	0.79	0.99	0.86	0.74	0.71	0.97	0.81	0.73	0.71	0.88	0.77	0.73	**0.70**	0.62	0.68
DEFER	0.64	0.54	0.79	0.66	0.53	0.43	0.29	0.42	0.28	0.17	0.26	0.24	0.02	0.02	0.67	0.24
GRACE	**1.00**	0.00	**1.00**	0.67	**1.00**	0.00	**1.00**	0.67	**1.00**	0.00	**1.00**	0.67	**1.00**	0.02	**1.00**	0.67
WISE	0.98	0.97	**1.00**	**0.98**	0.92	**0.89**	**1.00**	**0.94**	0.87	**0.80**	**1.00**	**0.89**	0.70	0.67	**1.00**	**0.79**

Baselines. The baselines include methods of continual learning and model editing. We compare WISE against direct fine-tuning **FT-L** with an additional KL divergence loss (Meng et al., 2022), and continual learning fine-tuning based on Elastic Weight Consolidation (**FT-EWC**) (Kirkpatrick et al., 2017). We also compare WISE to other model editors, including 1) GPT-style editors based on causal tracing: **ROME** (Meng et al., 2022), **MEMIT** (Meng et al., 2023), and **MEMIT-MASS** (a batch-editing version of MEMIT); 2) hypernetwork-based editors: **MEND** (Mitchell et al., 2022a); and 3) the latest memory-based editors: **DEFER** (inspired by SERAC (Mitchell et al., 2022b) for inference routing) and **GRACE** (Hartvigsen et al., 2023). Details on all comparisons are found in Appendix B.2.

Metrics. Each edit example includes an edit descriptor (i.e., query) $\mathbf{x}_e$, its paraphrase prompts $\mathbf{x}_{e'}$ (if available) for testing generalization, and an unrelated statement $\mathbf{x}_{\text{loc}}$ for testing locality. For the editing dataset $\mathcal{D}_{\text{edit}} = \{(\mathcal{X}_e, \mathcal{Y}_e)\}$ with T edits, we evaluate the final post-edit model f_{Θ_T} after the T-th edit example $(\mathbf{x}_T, \mathbf{y}_T)$. We evaluate the model editor's reliability and generalization using the metrics **Rel.** (a.k.a Edit Success Rate (Hartvigsen et al., 2023)) and **Gen.** (Generalization Success Rate (Zhang et al., 2024)), while **Loc.** (Localization Success Rate (Zhang et al., 2024)), defined as the post-edit model should not change the output of the irrelevant examples $\mathbf{x}_{\text{loc}}$, assesses specificity. We report these metrics and their mean scores, which are formally defined as:

$$
\begin{aligned}
\text{Rel.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(f_{\Theta_T}(\mathbf{x}_e^t) = \mathbf{y}_e^t), \\
\text{Gen.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(f_{\Theta_T}(\mathbf{x}_{e'}^t) = \mathbf{y}_e^t), \\
\text{Loc.} &= \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}(f_{\Theta_T}(\mathbf{x}_{\text{loc}}^t) = f_{\Theta_0}(\mathbf{x}_{\text{loc}}^t)),
\end{aligned}
\tag{8}
$$

where $\mathbb{1}(\cdot)$ is the indicator function. Notably, for the Hallucination dataset, following (Hartvigsen et al., 2023), we use the perplexity (PPL) to verify the locality, and there is no proper metric for generalization.

3.2 Main Results

Competitive Performance of WISE. The competitive performance of WISE is evident in Table 2 and 3, which compare its results with eight baselines on the QA (ZsRE) and Hallucination (SelfCheckGPT) settings. In general, we observe the followings: ❶ WISE outperforms existing methods on multiple tasks after long editing sequences; ❷ direct editing of long-term memory (ROME, MEMIT, etc.) creates conflicts with prior pretraining knowledge, resulting in poor locality; and ❸ retrieving working memory and modifying activations (GRACE, DEFER, etc) struggle to generalize to diverse queries.

Table 3: **Main editing results for Hallucination setting (SelfCheckGPT dataset).** T: Num Edits.

	Hallucination															
	`LLaMA-2-7B`								`Mistral-7B`							
	$T=1$		$T=10$		$T=100$		$T=600$		$T=1$		$T=10$		$T=100$		$T=600$	
Method	Rel. (*PPL* ↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)	Rel. (↓)	Loc. (↑)
FT-L	4.41	0.96	12.57	0.71	33.06	0.41	69.22	0.26	25.03	0.38	100.00	0.03	1594.93	0.00	-	-
FT-EWC	2.56	0.24	3.63	0.09	2.10	0.16	4.56	0.24	1.75	0.04	3.05	0.09	4.73	0.17	5.46	0.25
MEND	5.65	0.87	11.01	0.86	10.04	0.88	1847.90	0.00	7.64	0.96	83.74	0.05	23114.94	0.01	-	-
ROME	1.68	0.99	2.04	0.94	94.15	0.05	104.93	0.02	2.04	0.99	3.45	0.92	103.75	0.03	241.17	0.01
MEMIT	1.66	**1.00**	2.36	0.97	76.65	0.05	107.61	0.02	1.64	**1.00**	15.89	0.89	97.23	0.04	132.30	0.02
MEMIT-MASS	1.66	**1.00**	1.61	0.99	7.18	0.96	13.47	0.94	1.64	**1.00**	2.78	0.99	3.22	0.97	7.28	0.95
DEFER	**1.29**	0.23	3.64	0.28	8.91	0.19	19.16	0.12	4.76	0.45	7.30	0.25	9.54	0.43	24.16	0.13
GRACE	2.59	**1.00**	9.62	**1.00**	9.44	**1.00**	9.34	**1.00**	**1.39**	**1.00**	5.97	**1.00**	9.53	**1.00**	9.57	**1.00**
WISE	1.91	**1.00**	**1.04**	**1.00**	**1.14**	**1.00**	**3.12**	0.99	1.40	**1.00**	**2.56**	0.94	**1.31**	0.99	**5.21**	0.93

Figure 3: **OOD results for Temporal dataset.** `GPT-J-6B` is used.

Method	$T=10$				$T=100$			
	Rel.	OOD Gen.	Loc.	Avg.	Rel.	OOD Gen.	Loc.	Avg.
w/o Editing	*0.56*	*0.21*	-	*0.39*	*0.56*	*0.21*	-	*0.39*
FT-EWC	0.87	0.17	0.13	0.39	0.81	0.22	0.18	0.40
ROME	0.09	0.00	0.06	0.05	0.05	0.00	0.03	0.03
MEMIT-MASS	0.73	0.22	0.99	0.65	0.78	0.27	0.97	0.67
DEFER	0.68	0.33	0.08	0.36	0.52	0.26	0.08	0.29
GRACE	0.97	0.28	**1.00**	0.75	**0.97**	0.28	**1.00**	0.75
WISE	**0.99**	**0.36**	0.98	**0.78**	0.96	**0.37**	**1.00**	**0.78**

In the **QA** setting, with $T = 1000$, WISE achieves average scores of 0.83 and 0.79 on LLaMA and Mistral, respectively, reflecting improvements of 18% and 11% over the nearest competitor. This demonstrates WISE's outstanding stability and effective management of long-sequential edits. While methods like MEND and ROME are competitive early in editing, they show clear shortcomings as the edit sequence extends. Directly editing long-term memory (e.g., MEMIT, FT-EWC, MEND) results in a significant decline in Loc. When $T \in \{100, 1000\}$, this indicates that these methods cannot preserve LLMs' knowledge structure and significantly impair the model's generalization ability. GRACE excels in Loc. and Rel. (close to 1.00), however, it sacrifices generalization in continual editing. A possible reason is that token representation may not be suitable for measuring semantic similarity in autoregressive LMs, leading to paraphrase $\mathbf{x}_{e'}$ failing to achieve similarity matching with any CodeBook *Key* in GRACE (detailed in Appendix C.2). Overemphasis on preserving and precisely adapting training data (working memory) hampers adaptability to new contexts. In a nutshell, most previous methods struggle to balance Rel., Gen., and Loc., particularly in long-form editing tasks. In addition, the results of GPT-J-6B can be found in Figure 9 in the Appendix.

WISE also surpasses the baselines on the **Hallucination** dataset, maintaining the lowest perplexity scores of 3.12 and 5.21 at $T = 600$, with Loc. remaining above 0.93. We similarly observe significant *PPL* increases for FT-L, MEND, and ROME in long-context editing tasks, while GRACE's performance is lackluster in LLM long texts (possibly due to the limited fitting capacity of the very small active trained parameters $|h^l|$ of GRACE).

Out-of-Distribution Evaluation. Ideally, model editing needs to generalize distributionally from formulaic editing examples to natural texts (Hewitt et al., 2024), where the distributional shift involves complexity rather than conventional domain shift (Oren et al., 2019). Following (Hewitt et al., 2024), we evaluate the OOD generalization of editing methods on emerging entities. Editing examples and evaluation metrics are provided in Appendix B.1. As shown in Table 3, WISE effectively handles out-of-distribution generalization tasks (achieving the best OOD Gen. and overall performance). DEFER delivers mediocre performance on OOD Gen. due to the limited capacity of the auxiliary model(Yao et al., 2023). During the fine-tuning phase, GRACE and MEMIT focus on the representation $v*$ of a **single** input token after $\mathbf{W}_v$ (GRACE: last token, MEMIT: last subject token). However, regarding $v*$ the editing carrier encounters two problems: 1) the training objective is not aligned with the pretraining phase, and 2) the single representation limits the search scope of gradient descent, making it difficult to handle complex OOD generalization. WISE, on the other hand, avoids the above challenges.

4 Further Analysis

Visualization of WISE's Routing Activation. To demonstrate the effectiveness of memory routing, we record the activation values $\Delta_{\text{act}}(\mathbf{x})$ of 1000 (QA, ZsRE)/600 (Halluc.) queries during the inference stage via knowledge merging into a single side memory. As shown in Figure 6, the purple horizontal line represents the activation threshold

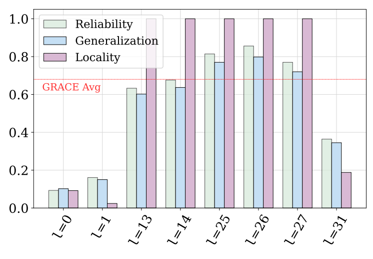


Figure 4: **Analysis of locating FFN layer of side memory for WISE.** ZsRE, `LLaMA-2-7B`.

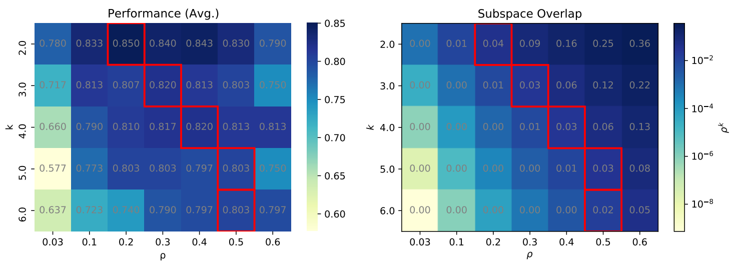


Figure 5: **Analysis of different mask ratios ρ and subspaces k for WISE.** Left: Avg. performance of Rel., Gen., and Loc.; Right: the subspace overlap probability in Theorem 2.1. ZsRE, `LLaMA-2-7B`.

ϵ recorded during the editing phase. Almost all

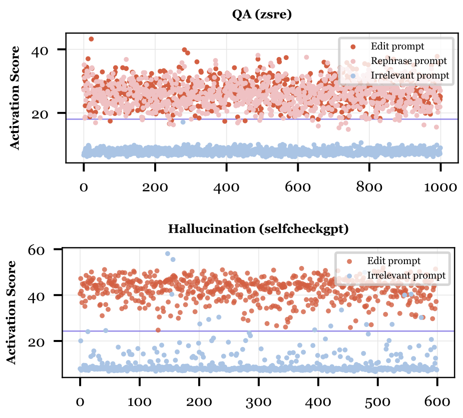


Figure 6: **Activations of the memory routing module of WISE when varying T.** `X-axis`: Num edits. `LLaMA-7B`.

unrelated queries show low activations with values less than 10 in ZsRE and less than 20 in Halluc.; meanwhile, WISE accurately routes the editing prompt and unseen paraphrases into the side memory. This ensures editing locality and prevents excessive shifts from the pre-training distribution during lifelong editing.

Localization Analysis of WISE's Side Memory. To validate the benefits of editing mid-to-late layers, we select decoder layers from early, intermediate, mid-to-late, and late stages. As shown in Figure 4, the ablation results reveal that editing critical layers like the early and final layers (0, 1, 31) is ineffective, even resulting in a very low Loc. value of 0.096, which indicates a failure to recognize the editing scope. This may occur because the early layers represent fundamental grammatical information, and the final layer directly controls the decoding procedure, leading to poor editing of advanced language functions. Editing in the intermediate layers is suboptimal but still shows a markable improvement compared to early layers, possibly because intermediate layers start to integrate basic grammatical information with more complex semantic data. Notably, the mid-to-late layers demonstrate exceptional editing performance; for instance, selecting layer 26 results in an 80% success rate and generalization while maintaining 100% locality. This empirically supports our claim in Section 2.2.1 that the redundant mid-to-late layers (Men et al., 2024) are ideal side memory layers and confirms the hierarchical nature of information processing in Transformer LLMs (Mueller et al., 2022; Murty et al., 2023).

Analysis of ρ and k for WISE. We analyze the important hyperparameters of WISE: the mask ratio ρ and the number of subspaces k in Figure 5. On the left figure, for $k = 2$, the best ρ is 0.2, satisfying $k * \rho = 0.4 < 1$, which implies the effectiveness of our subspace design that higher knowledge density will cause better generalization. When scaling k, we observe an increasing demand of ρ. From Theorem 2.1, the probability of subspace overlap is ρ^k, and we hypothesize that this overlap is important as an anchor for model merging. Interestingly, from the right figure, it can be observed that the optimal cases always have the ρ^k closest to 0.03. This shows an inherent tradeoff between merge anchor and merge conflicts, and the subspace overlaps around 0.03 are optimal for the best performances.

5 Conclusion

For related works, please refer to Appendix E.

In this paper, we point out the impossible triangle of current lifelong modeling editing approaches that reliability, generalization, and locality can hardly be achieved simultaneously. We find the reason behind this is the gap between working and long-term memory. Therefore, we propose WISE, consisting of side memory and model merging, to remedy the gap. Extensive results show WISE is promising to reach high metrics at once on various datasets and LLM models.

Limitations and Broader Impacts

Although WISE shows promising results in lifelong editing, it also has some limitations. One limitation is addressed in Table 11 that the side memory retrieval has room for improvement to reach the oracle. Also, in Figure 12, the inference time of WISE-Retrieve increases with ever-growing editing streams. However, the current limitations cannot outweigh the merits of WISE in that it currently reaches better performance in general for lifelong model editing.

We bridge the gap between long-term and working memory, it may inspire further work on memory design for model editing or even LLM architecture. However, the application of such technologies should be guided by ethical considerations. Malicious users may attempt to edit LLMs to propagate hate, highlighting the need for safeguards to prevent abuse and mitigate harmful outcomes. Some current model editors update the model's weights directly, making edits hard to trace and withdraw. WISE uses a modular and non-destructive side memory, allowing users to discard it if edits are unnecessary or harmful, without modifications to the main LLMs.

References

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2023. Git re-basin: Merging models modulo permutation symmetries. In *The Eleventh International Conference on Learning Representations*.

Afra Feyza Akyürek, Ekin Akyürek, Derry Wijaya, and Jacob Andreas. 2022. Subspace regularizers for few-shot class incremental learning. In *International Conference on Learning Representations*.

Ibrahim M Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. 2022. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. 2019. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.3, knowledge capacity scaling laws.

Vidhisha Balachandran, Hannaneh Hajishirzi, William Cohen, and Yulia Tsvetkov. 2022. Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9818–9830.

Baolong Bi, Shenghua Liu, Lingrui Mei, Yiwei Wang, Pengliang Ji, and Xueqi Cheng. 2024. Decoding by contrasting knowledge: Enhancing llms' confidence on edited facts. *Preprint*, arXiv:2405.11613.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. 2024. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Yi Dai, Hao Lang, Yinhe Zheng, Fei Huang, Luo Si, and Yongbin Li. 2022. Lifelong learning for question answering with hierarchical prompts. *arXiv e-prints*, pages arXiv–2208.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, and Leshem Choshen. 2023. Cold fusion: Collaborative descent for distributed multitask finetuning. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 788–806.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2022. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*.

Emilio Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *Challenges and Risks of Bias in Large Language Models (October 26, 2023)*.

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

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. 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.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Joseph B Hellige. 2001. *Hemispheric asymmetry: What's right and what's left*, volume 6. Harvard University Press.

Thomas Henn, Yasukazu Sakamoto, Clément Jacquet, Shunsuke Yoshizawa, Masamichi Andou, Stephen Tchen, Ryosuke Saga, Hiroyuki Ishihara, Katsuhiko Shimizu, Yingzhen Li, et al. 2021. A principled approach to failure analysis and model repairment: Demonstration in medical imaging. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pages 509–518. Springer.

John Hewitt, Sarah Chen, Lanruo Lora Xie, Edward Adams, Percy Liang, and Christopher D. Manning. 2024. Model editing with canonical examples. *Preprint*, arXiv:2402.06155.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. 2024. Transformer fusion with optimal transport. In *The Twelfth International Conference on Learning Representations*.

Richard B Ivry and Lynn C Robertson. 1998. *The two sides of perception*. MIT press.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

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 arXiv:2001.08361*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2023a. Large language models with controllable working memory. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1774–1793.

Shuaiyi Li, Yang Deng, Deng Cai, Hongyuan Lu, Liang Chen, and Wai Lam. 2024a. Consecutive model editing with batch alongside hook layers. *Preprint*, arXiv:2403.05330.

Weishi Li, Yong Peng, Miao Zhang, Liang Ding, Han Hu, and Li Shen. 2023b. Deep model fusion: A survey. *arXiv preprint arXiv:2309.15698*.

Zexi Li, Zhiqi Li, Jie Lin, Tao Shen, Tao Lin, and Chao Wu. 2024b. Training-time neuron alignment through permutation subspace for improving linear mode connectivity and model fusion. *arXiv preprint arXiv:2402.01342*.

Zexi Li, Tao Lin, Xinyi Shang, and Chao Wu. 2023c. Revisiting weighted aggregation in federated learning with neural networks. In *International Conference on Machine Learning*, pages 19767–19788. PMLR.

Bill Yuchen Lin, Sida I Wang, Xi Lin, Robin Jia, Lin Xiao, Xiang Ren, and Scott Yih. 2022. On continual model refinement in out-of-distribution data streams. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3128–3139.

Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. 2024. Meaning representations from trajectories in autoregressive models. In *The Twelfth International Conference on Learning Representations*.

Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.

Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12284–12314.

Aaron Mueller, Robert Frank, Tal Linzen, Luheng Wang, and Sebastian Schuster. 2022. Coloring the blank slate: Pre-training imparts a hierarchical inductive bias to sequence-to-sequence models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1352–1368, Dublin, Ireland. Association for Computational Linguistics.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. 2024. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. 2023. Grokking of hierarchical structure in vanilla transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 439–448, Toronto, Canada. Association for Computational Linguistics.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? In *The Twelfth International Conference on Learning Representations*.

OpenAI and the Co-authors. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2024. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36.

Yulia Otmakhova, Karin Verspoor, and Jey Han Lau. 2022. Cross-linguistic comparison of linguistic feature encoding in BERT models for typologically different languages. In *Proceedings of the 4th Workshop on Research in Computational Linguistic Typology and Multilingual NLP*, pages 27–35, Seattle, Washington. Association for Computational Linguistics.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2023. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ankit Singh Rawat, Chen Zhu, Daliang Li, Felix Yu, Manzil Zaheer, Sanjiv Kumar, and Srinadh Bhojanapalli. 2021. Modifying memories in transformer models. In *International Conference on Machine Learning (ICML)*, volume 2020.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems*, 32.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*, volume 35, pages 17456–17472. Curran Associates, Inc.

Ohad Shamir and Tong Zhang. 2013. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 71–79, Atlanta, Georgia, USA. PMLR.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey. *Preprint*, arXiv:2404.16789.

Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *Advances in Neural Information Processing Systems*, 35:19523–19536.

Matthew Sotoudeh and A Thakur. 2019. Correcting deep neural networks with small, generalizing patches. In *Workshop on safety and robustness in decision making*.

George Stoica, Daniel Bolya, Jakob Brandt Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. 2024. Zipit! merging models from different tasks without training. In *The Twelfth International Conference on Learning Representations*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Frederik Träuble, Anirudh Goyal, Nasim Rahaman, Michael Curtis Mozer, Kenji Kawaguchi, Yoshua Bengio, and Bernhard Schölkopf. 2023. Discrete key-value bottleneck. In *International Conference on Machine Learning*, pages 34431–34455. PMLR.

Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. In *International Conference on Learning Representations*.

Yu Wang, Xiusi Chen, Jingbo Shang, and Julian McAuley. 2024. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. 2022a. Dual-prompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *Preprint*, arXiv:2402.01364.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240.

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.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876, Singapore. Association for Computational Linguistics.

A Appendix

In the Appendix, we introduce more details along with additional experimental results, discussions, and related works:

- Appendix B: Experimental setups (Section 3).
- Appendix C: More experimental results (Section 2 and 3).
- Appendix D: Proof of the Theorem 2.1 (cf. Section 2).
- Appendix E: Additional discussions and related works.

B Implementation Details

B.1 Description of Datasets

ZsRE The ZsRE question-answering task (Levy et al., 2017) is extensively studied within the model editing literature (Meng et al., 2022, 2023; Mitchell et al., 2022a; Tan et al., 2023; Huang et al., 2023), where each record contains an editing statement $\mathbf{x}_e$, a paraphrase prompt $\mathbf{x}_e'$, and a locality prompt $\mathbf{x}_{\text{loc}}$. We use the same train/test split as (Mitchell et al., 2022a) (163196/19086). Notably, only MEND requires fitting a hypernetwork on the training set; other methods discard the training set and perform edits and evaluations on the test set. In practice, we randomly sample 1K and 3K records from the test set to form the edit sets in Section 3.2 and C.7.

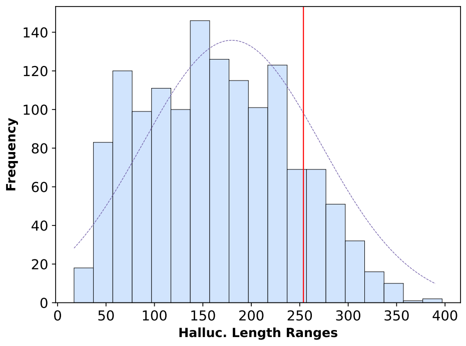


Figure 7: Hallucination length statistics.

Hallucination We utilize the same dataset as GRACE, SelfCheckGPT (Manakul et al., 2023), to assess the ability of Model Editors to mitigate hallucinations in autoregressive LMs. This setting involves editing highly inaccurate sentences (sourced from GPT-3 (Brown et al., 2020)) and replacing them with corresponding sentences from actual Wikipedia entries. This dataset aligns more closely with real-world deployment scenarios where models trigger "unexpected behaviors," and the token length of edits is significantly longer than in past datasets, making it a more challenging editing setting. Unlike GRACE, which used GPT2-XL (1.5B) (Radford et al., 2019), our main experiments deploy larger LLMs, LLaMA and Mistral, both with 7B parameters, we measure retention of pretraining data ($\mathbf{x}_{\text{loc}}$) from the base model: RedPajama (Computer, 2023), a public version of LLaMA's pretraining data. Some of the exceptionally long editing samples cannot even be accommodated on an NVIDIA A800 (80GB) due to resource limitations. As shown in Figure 7, the original dataset provided by GRACE, after tokenization with LLAMATOKENIZER, has length distributions ranging from [17,390]. The dimension of a single MLP layer in `llama-2-7b-hf` is (11008, 4096) [*]. Theoretically, fine-tuning an input of length 390 with default full precision and the Adam optimizer would require (390+4+4+4) * (11008 * 4096 * 4) + 4 * 7B = **100.36GB** of VRAM (for activations, gradients, first-order, and second-order optimizers), exceeding the memory capacity of the NVIDIA A800. Consequently, we excluded excessively long samples (limiting tokenized lengths to 254) and ultimately retained 906 editing instances (compared to 1392 in GRACE). To facilitate a fair comparison with MEND, we specifically allocated a training set for MEND, with a final train/test split of 306/600. All methods were edited and evaluated on the test set.

Temporal (Hewitt et al., 2024) sources the prefix $\mathbf{x}_e$ from the first paragraph of an entity's Wikipedia page and samples a paragraph $\mathbf{y}_e$ discussed by GPT-4 (OpenAI and the Co-authors, 2024) about the emerging entity $\mathbf{x}_e$, which is usually noisy but may contain helpful information. These are presented as editing prompts to Model Editors. For out-of-distribution (OOD) generalization to complex natural contexts (not fitted), $\mathbf{y}_{\text{ood}}$ is taken from the actual Wikipedia suffix of $\mathbf{x}_e$. This setup is utilized to evaluate the OOD generalization of Model Editors centered around a single canonical example. Consistent with previous work (Hartvigsen et al., 2023), the out-of-scope data $\mathbf{x}_{\text{loc}}$ is derived from the Pile (Gao et al., 2020), the pretraining corpus of GPT-J-6B. Examples from the dataset can be seen in Table 5. To measure the OOD generalization of editing methods for emerging entities, we perform

[*]`https://huggingface.co/meta-llama/Llama-2-7b-hf`

Table 4: Bolded text refers to the edit labels $\mathbf{y}_e$. Locality example $\mathbf{x}_{\text{loc}}$ is an unrelated query.

(a) **ZsRE, question-answering** editing dataset example.

$\mathbf{x}_e, \mathbf{y}_e$	Which continent is Berkner Island in? **South America**
$\mathbf{x}_{\text{loc}}$	who gets the golden boot if its a tie? **shared**
$\mathbf{x}'_e, \mathbf{y}_e$	On which continent is Berkner Island located? **South America**

(b) **Hallucination** editing dataset example. In the original data (Hartvigsen et al., 2023), there is no paraphrase $x_{e'}$ so the measurement of Gen. metric is ignored here.

$\mathbf{x}_e, \mathbf{y}_e$	This is a Wikipedia passage about heinz christian pander. Heinz Christian Pander (1794 - 1865) was a German anatomist and embryologist who was born in Riga, Latvia. He studied medicine at the University of Dorpat and later at the University of Berlin. **In 1820, he took part in a scientific expedition to Bokhara as a naturalist.**
$\mathbf{x}_{\text{loc}}$	Tired and restlessly, drifting in and out of sleep. Hearing crashing and banging, thinking the roof will cave in. Not alert enough to quite know what it was, I yelled loudly for whoever was making those noises at such an hour to stop. They heard and listened, I'm guessing

model editing using standardized simple examples and then evaluate this behavior on more complex instances. Following (Hewitt et al., 2024), in a natural setting, no single correct continuation exists. Thus, we also use probability threshold-based evaluations, such as 80%, where the editing success rate evaluates whether the loss $L_{\mathbf{x}_e,\mathbf{y}_{\text{ood}}}$ for an example falls below $\delta = -\log(0.8)$, as indicated in the formula below. The intuition behind this is that many other plausible alternative continuations may exist.

$$\text{OOD Gen.} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\{(L_{\Theta_T}(\mathbf{x}_e, \mathbf{y}_{\text{ood}}) < \delta)\}. \tag{9}$$

B.2 Descriptions of Compared Model Editors

FT-L. All other layers of the LLMs remain frozen, and only a single MLP layer is fine-tuned through autoregressive loss (Meng et al., 2022). Additionally, we impose an L_∞ norm constraint to prevent the parameters from deviating too far from the pretrained distribution.

FT-EWC. Elastic Weight Consolidation (EWC) has been demonstrated to mitigate catastrophic forgetting by updating weights using a Fisher information matrix, which is computed from past edits, multiplied by a scaling factor λ (Kirkpatrick et al., 2017). Following (Hartvigsen et al., 2023), we omit the constraints of the L_∞ norm in this implementation.

MEND. MEND (Mitchell et al., 2022a) transforms the gradients obtained from standard fine-tuning using a hypernetwork that converts gradients decomposed into low rank (rank=1) into new gradients, which are then applied to the target layer for parameter updates. During the training phase, a small auxiliary hypernetwork receives editing examples $(\mathbf{x}_e, \mathbf{y}_e)$, and $\mathbf{x}_{\text{loc}}$. MEND's training loss comprises the standard autoregressive loss combined with the KL divergence loss of the model's output on $\mathbf{x}_{\text{loc}}$ before and after editing. This hypernetwork plays a crucial role during the editing procedure.

ROME. ROME (Meng et al., 2022) uses causal analysis to pinpoint knowledge within specific MLP layers and modifies the entire matrix through least squares approximation. It operates under the strong assumption that the MLP is the primary module for storing knowledge (Geva et al., 2021), and it injects a single piece of knowledge into the MLP at each iteration using a Lagrangian remainder.

MEMIT. Similarly, based on the assumption that the FFN serves as a knowledge key-value store, MEMIT (Meng et al., 2023) manipulates parameters of specific layers directly through least squares approximation. Unlike ROME, which updates a single layer, MEMIT is a multi-layer updating algorithm that supports simultaneous updates of hundreds or thousands of facts. For sequential model editing tasks, MEMIT requires immediate on-the-fly repairs when the model makes errors, expressed as $f_{\Theta_T} = \text{MEMIT}(f_{\Theta_{T-1}}, \mathbf{x}_T, \mathbf{y}_T)$, involving multiple operations on the original model.

MEMIT-MASS. Unlike sequential editing, MEMIT supports modification of multiple knowledge fragments in a batch mode, named **MEMIT-MASS**. Suppose we collect streaming errors as $(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_0, \mathbf{y}_0), (\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_T, \mathbf{y}_T)\}$ and inject them collectively into the MLP, it only involves a single editing operation on the original

Table 5: **Temporal** OOD dataset example. Bolded text refers to the edit labels $\mathbf{y}_e$ and $\mathbf{y}_{\text{ood}}$.

$\mathbf{x}_e, \mathbf{y}_e$	Self-driving cars, **also known as autonomous vehicles, are vehicles that are capable of navigating and operating without human intervention. These innovative vehicles rely on a combination of advanced sensors, artificial intelligence, and computer algorithms to interpret their environment and make real-time decisions. With the potential to significantly impact numerous industries and sectors, self-driving cars have the ability to revolutionize transportation by enhancing safety, improving traffic flow, and increasing energy efficiency. However, challenges related to regulatory frameworks, ethical considerations, and public acceptance still need to be addressed before widespread adoption becomes a reality.**
$\mathbf{x}_{\text{loc}}$	Apple has a new peach with the release of its 3.0GHz, 8-core Intel Xeon-based Mac Pro. The 8-core Mac Pro is powered bu two quad-core Intel Xeon C̈lov ertownp̈rocessors running at 3.0GHz. Apple also released a quad-core Mac Pro featuring two Dual-Core Intel Xeon Ẅoodcrestp̈rocessors.
$\mathbf{x}_e, \mathbf{y}_{\text{ood}}$	Self-driving cars, **also known as autonomous cars or driverless cars, are vehicles capable of traveling without human input. These cars utilize a range of sensors, including optical and thermographic cameras, radar, lidar, ultrasound/sonar, GPS, odometry, and inertial measurement units, to perceive their surroundings. By interpreting sensory information, control systems in the car are able to create a three-dimensional model of its environment. Using this model, the car can then identify the best navigation path and develop strategies for managing traffic controls and obstacles. As self-driving car technology continues to advance, it is expected to have a significant impact on various fields such as the automotive industry, health, welfare, urban planning, traffic, insurance, and the labor market. The regulation of autonomous vehicles is also becoming an increasingly important topic of discussion.**

model as $f_{\Theta_T} = \text{MEMIT}(f_{\Theta_0}, \mathcal{X}, \mathcal{Y})$. Although this approach **loses the capability for on-the-fly repairs**, we still include this baseline in our experiments.

DEFER. In GRACE, a reimplementation of SERAC (Mitchell et al., 2022b) is utilized, denoted as DEFER. For new inputs, DEFER includes a network g (corresponding to the *scope classifier* in SERAC) that predicts whether to: 1) trust the prediction of the LLMs, or 2) trust the prediction of the new model. Here, the new model is configured as a single-layer linear network o with a sigmoid activation function, corresponding to the *counterfactual model* in SERAC. During the editing process, g and o are fine-tuned jointly.

GRACE. GRACE (Hartvigsen et al., 2023) utilizes a discrete KEY-VALUE codebook and maintains the codebook throughout the editing flow by adding, expanding, and splitting KEYs. During the inference phase, it retrieves the nearest KEY and determines whether to replace the activation of the hidden layer output.

B.3 Training Details and Hyperparameters

Except for MEMIT-MASS, the batch size for all methods is consistently 1 in sequential editing scenarios. All experiments are conducted using 3 NVIDIA A800 GPUs, with all tasks reproducible on a single A800. Editing ZsRE takes approximately 4 hours, while Hallucination requires around 6 hours. To ensure fair comparisons, unless otherwise specified (for some methods like MEND, ROME, and MEMIT, we follow the original literature by selecting the last few layers or using causal analysis to identify the target layers), the default target layers for editing on `LLaMA`, `Mistral`, and `GPT-J` are `model.layers[27].mlp.down_proj.weight`, `model.layers[27].mlp.down_proj.weight`, and `transformer.h[21].mlp.c_fc`, respectively.

For FT-L, we utilize a reimplementation from

Table 6: **Comparison of current model editing methods.** "✓" refers to "yes" and "well-supported", ✗ refers to "no" or "badly-supported", and "○" refers to "less-supported". The three metrics of Reliability, Generalization, and Locality denote the performances on lifelong (continual) editing.

Methods	Long-term Memory	Working Memory	Parametric Knowledge	Retrieval Knowledge	Whether Lifelong	Reliability	Generalization	Locality
FT-EWC	✓	✗	✓	✗	✓	✓	✓	✗
ROME/MEMIT	✓	✗	✓	✗	✗	✗	✗	✗
MEND	✓	✗	✓	✗	✗	✗	✗	✗
SERAC/DEFER	✗	✓	✓	✓	✓	○	✗	○
GRACE	✗	✓	✗	✓	✓	✓	✗	✓
WISE	✓	✓	✓	✓	✓	✓	✓	✓

ROME [†], employing the Adam (Kingma and Ba, 2015) optimizer with consideration of learning rates at 1e-5, 1e-4, and 5e-4, and conducting gradient descents for 50 iterations, ultimately reporting the best results at a learning rate of 5e-4.

For FT-EWC, we follow the reimplementation in GRACE and its default settings, setting the learning rate at 1e-2, the λ_{ewc} penalty factor at 0.1, and the number of replay instances at 10.

For the training phase of MEND, we adhere to the original paper, setting the learning rate at 1e-4, iterating 100K times, and employing early stopping at 30K, ultimately achieving an accuracy of 0.95 on the training set. Notably, we target the last few MLP layers as per the original literature, such as `model.layers[i].mlp.down_proj.weight`, `model.layers[i].mlp.gate_proj.weight`, `layers[i].mlp.up_proj.weight` in LLaMA, where $i \in [29, 30, 31]$.

For ROME and MEMIT, we follow the original literature on GPT-J using the default configurations, specifically the fifth layer and layers [3,4,5,6,7,8]. In LLaMA and Mistral, additional causal analysis is conducted to pinpoint the layers storing knowledge. As shown in Figure 8, an increasing trend in the Average Indirect Effect of the MLP is observed across layers [4,5,6,7,8], suggesting that the model recalls factual knowledge here and passes the matured token distribution via residual connections to the last MLP. Thus, in LLaMA and Mistral, ROME edits the fifth layer, while MEMIT edits layers [4,5,6,7,8].

For DEFER, the original literature uses a learning rate of 1.0; however, we found it unfit for LLaMA and Mistral, with severe fluctuations in model loss. Therefore, we experiment with learning rates of 7e-5, 7e-4, and 1e-3, and ultimately report using 7e-5 (optimal).

For GRACE, we strictly follow the original literature, setting the learning rate at 1.0, and using `replace_last` to only replace the activation of the last token in autoregressive scenarios. After observing failures in generalization, we adjust various ϵ_{init} values and discuss this more in Appendix C.2. For WISE, the hyperparameters for the QA and Hallucination tasks are identical. We find that a learning rate of 1.0 with the SGD (Shamir and Zhang, 2013) optimizer is a good approach for stable training. The hyperparameters designed in the knowledge editing phase include the random masking probability ρ and the routing threshold guidance α, β, γ. In the knowledge merging phase, hyperparameters include the number of merges k and the merging weights λ for each MLP (we discuss the impact of ρ and k in Section 4). Theoretically, as the importance of knowledge in any MLP is considerable, we always average with $\lambda = 1/k$. These are shown in Table 7.

Table 7: WISE hyper-parameters in editing.

Hyper-Parameters	Values
Optimizer	SGD
LR η	1.0
Mask Ratio ρ	0.2
α	5.0
β	20.0
γ	10.0
Merge Weights λ	0.5
Knowledge shards k	2

C More Experimental Results and Analyses

C.1 Rethinking the Memory Design of Lifelong Model Editing

In Table 6, we compare current model editing methods in terms of memory types and lifelong editing abilities. FT-EWC (Kirkpatrick et al., 2017), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and MEND (Mitchell et al., 2022a) edit the long-term memory stored in the LLMs' model pa-

[†]https://github.com/kmeng01/rome

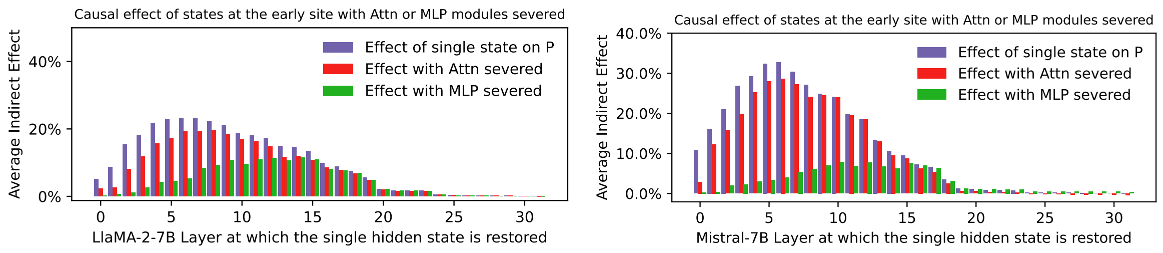


Figure 8: Mid-layer MLPs play a crucial mediating role in `LLaMA-2-7B` and `Mistral-7B`.

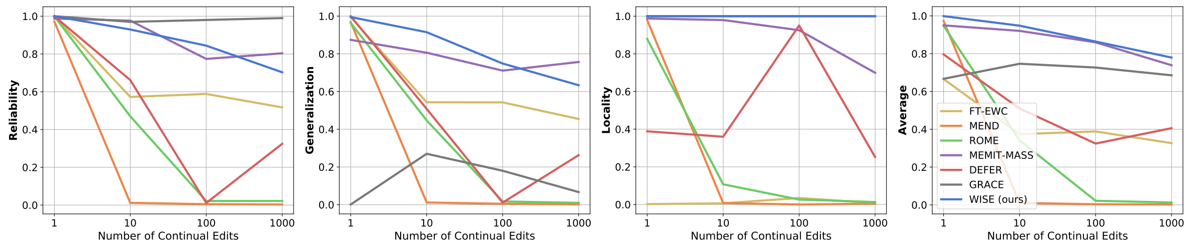


Figure 9: GPT-J-6B, ZsRE, continual editing.

rameters, but they either do not support continual editing or have negative effects on irrelevant knowledge (poor locality). GRACE (Hartvigsen et al., 2023) is designed for lifelong editing via retrieval-based working memory. The retrieval codebook can avoid the conflicts of irrelevant knowledge, but GRACE fails to generalize due to its codebook being a non-parametric knowledge representation that solely memorizes queries without comprehension. It is worth noting that SERAC (Mitchell et al., 2022b)/DEFER (Hartvigsen et al., 2023) uses working memory that is stored in additional small models: a scope classifier and a counterfactual model, whose knowledge is parametric. However, the small counterfactual model cannot match the expressiveness and generalization capabilities of LLM itself, making it challenging for the edited knowledge to generalize effectively.

To enable effective lifelong model editing, the method should take advantage of both LLM parameters' long-term memory and retrieval-based working memory. Therefore, we propose WISE as follows.

C.2 On the Pitfall of GRACE: Generalization Collapses in Decoder-only LLMs

Here, we discuss why GRACE exhibits poor generalization when editing decoder-only LMs.

As shown in Figure 14, we continuously edit 15 samples $(\mathbf{x}_e, \mathbf{y}_e)$ using GRACE and observe the nearest codebook *Key* for their paraphrases $\mathbf{x}_{e'}$ and unrelated queries $\mathbf{x}_{\text{loc}}$, as well as the governed *Deferral radii* ϵ of those *Keys*. When overlapping *Keys* exist, GRACE reduces the *Deferral radii* to split this *Keys* and then adds a new codebook entry, resulting in exponentially decaying of radii ϵ during the editing process. Though ϵ is initialized from a high ϵ_{init}, it will be small and ineffective after continuous edits. From Figure 14, we observe that GRACE is more likely to have a conservative strategy that sets smaller Deferral radii during editing. Smaller Deferral radii will cause $\mathbf{x}_{e'}$ to fail to hit the codebook (the distance to the nearest *Key* is farther than its *Deferral radii*) but let $\mathbf{x}_{\text{loc}}$ successfully far away from the radii, resulting low generalization and high locality. Also, we observe that the Deferral radii method is not effective under any ϵ_{init}; for all tested ϵ_{init} values of 1.0, 3.0, 10.0, and 500.0, they all have low generalization and high locality.

This suggests that in autoregressive LMs, the distribution of the last token cannot effectively represent semantics; whereas in encoder-only and encoder-decoder architectures, capturing semantic information through vector representation has been extensively studied (Devlin et al., 2019; Reimers and Gurevych, 2019; Gao et al., 2021). This is consistent with the degree of generalization shown by GRACE when anchoring the T5 (Raffel et al., 2020) Encoder layer. Some related works (Liu et al., 2024) also indicate that in autoregressive models, semantic similarity measures based on averages of output tokens underperform, recommending the use of score distributions over text continuations to represent semantic distances.

Table 8: Varying Merging Strategy. ZsRE. `LLaMA-2-7B`.

Methods	Rel.	Gen.	Loc.	Avg.
Linear	.63	.61	.93	.72
Slerp	.62	.64	.91	.72
Dare	.68	.63	.92	.74
Dare_Ties	.67	.63	.83	.71
Ties	**.85**	**.81**	<u>.94</u>	**.87**
Sign	<u>.80</u>	<u>.76</u>	**.97**	<u>.84</u>

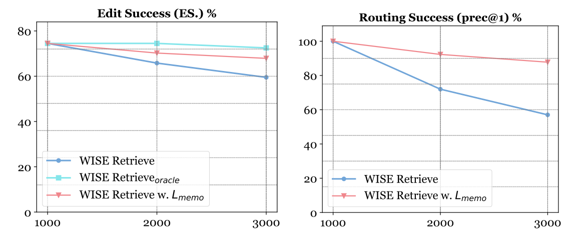
C.3 Impact of Knowledge Merging Strategies for WISE

Here, we conduct a more in-depth study of the knowledge merging strategies for WISE, exploring various merging approaches including (i) *Linear*, which uses a simple weighted average; (ii) *Slerp*, which spherically interpolates the parameters of two models; (iii) *Ties*, a component used in the main experiments of this paper that resolves merging disturbances through TRIM ELECT SIGN; (iv) *Dare*: which follows a Bernoulli distribution to delete redundant parameters and rescale the remaining ones; (v) *Dare_Ties*, which combines dare and the sign consensus algorithm of TIES; and (vi) *Sign*, an ablation component of Ties that addresses directional conflicts—all utilizing the official implementation from MergeKit (Goddard et al., 2024) [‡]. We randomly sample 100 edits from ZsRE, retaining a fine-tuned MLP every 50 edits (merging 2 MLPs). As shown in Table 8, we observe that ignoring the direction of parameter updates (Linear, Slerp, Dare) leads to a significant decline in editing performance, underscoring the importance of addressing knowledge conflicts in overlapping parameters. The success of *Sign* also reaffirms this point. Meanwhile, the randomly masked knowledge shards exhibit a non-redundancy, indivisible nature. This is demonstrated by the significantly weaker performance of *Dare_Ties* compared to *Ties*/*Sign*, indicating that removing parameter updates can lead to the loss of edited knowledge or even potential "anchors".

C.4 Analysis of Retrieving Top-1 Activation

WISE-`Retrieve` retains each knowledge-sharding memory and retrieves through Top-1 Activation. However, as shown in Table 11 and Figure 10b, the retrieval accuracy still has significant room for improvement; specifically, when T reaches 3K, the accuracy of routing to the correct MLP drops to

[‡]`https://github.com/arcee-ai/mergekit`



(a) Average of Rel. and Gen. (b) Retrieval Acc.

Figure 10: Comparing editing results of WISE-{`Retrieve`, `Retrieve`$_{\text{oracle}}$, `Retrieve w.` L_{memo}} when varying T. (a) shows the simple average of Rel. and Gen. (ES.), while (b) shows retrieval accuracy, i.e., whether the Top-1 Activation routes to the correct MLP (prec@1). `X-axis`: Num edits. ZsRE. `LlaMA-2-7B`.

around 60%, indicating the specificity between side memories is insufficient. One possible reason is that when sampling the edits from a single dataset (ZsRE), the editing instances $(\mathbf{x}_e, \mathbf{y}_e)$ all belong to the same domain. This leads to some very similar instances being captured by multiple expert side memories (resulting in high activations for all side memories), introducing more retrieval failures.

Therefore, to improve the specificity of side memory and reduce the probability of routing errors, we attempt to add a new constraint L_{memo} to Equation 4. For knowledge-sharding memory $\mathbf{W}_i$, we randomly replay instances $(\mathbf{x}_{\text{m}}, \mathbf{y}_{\text{m}})$ from the edit set $\mathcal{D}_{\mathbf{W}_j}$ of past shard $\mathbf{W}_{j,\, j\in[0,i-1]}$, ensuring that $\mathbf{W}_i$ remains inactive for $\mathbf{x}_{\text{m}}$:

$$L'_a = L_a + \underbrace{\max(0, \Delta_{\text{act}}(\mathbf{x}_{\text{m}}) - \alpha)}_{L_{\text{memo}}}, \quad \text{s.t. } \mathbf{x}_{\text{m}} \in \mathcal{D}_{\mathbf{W}_j}.$$

As shown in Figure 10b, this replay behavior increases the specificity between side memories, maintaining nearly **88%** retrieval accuracy at $T = 3K$. Figure 10a also shows that WISE-`Retrieve` *w.* L_{memo} improves Edit Success (ES.) by **8.39%** compared to WISE-`Retrieve`, providing a promising direction for future work. With finer-grained activation management, we might be able to bridge the performance gap between `Retrieve` and `Oracle`.

C.5 Case Study

In Table 9, we present bad cases of using WISE to edit the `LLaMA-2-7B` on the ZsRE dataset and mitigating these failures is critical for future work in model editing. We observe that in $i)$ errors occur only in part of the tokens, and these errors constitute a large proportion of the bad cases, indicating that the edits have not been sufficiently

Table 9: **Failure cases of using WISE** to edit `LLaMA-2-7B`. ✗represents errors in part of the tokens, ✗represents complete output errors (i.e., factual failures), and ✓indicates the expected exact match.

	Prompt	Edit Target	Post-Edit Output
ia)	By which person Lahti Town Hall has been designed?	Aki Kaurismäki	Wime Kaurismäki ✗
ib)	*Which is the architect of Lahti Town Hall?*	-	Wime Kaurismäki ✗
ic)	Which corporation was USS Leedstown (APA-56) created by?	Lockheed Shipbuilding	Leez Shipbuilding ✗
id)	*Which company manufactures the USS Leedstown (APA-56)?*	-	Leez Shipbuilding ✗
iia)	Which language is Garowe Principles written in?	Persian	Dutchian ✗
iib)	*In what language does the monthly football magazine Garowe Principles report?*	-	Somian ✗
iic)	What year was the service entry date for Panzer 58?	1957	1953 ✗
iid)	*What was the year Panzer 58 was commissioned?*	-	1953 ✗
iiia)	What was Gemma Bosini's range?	mezzo-srano	Wzo-srano ✗
iiib)	*The kind of voice of Gemma Bosini is what?*	-	mezzo-srano ✓
iva)	In which state is Qaleh Lan located?	Golestan Province	Golestan Province ✓
ivb)	*What state is Qaleh Lan in?*	-	Lestan Province ✗
ivc)	In which language Garowe Principles monthly football magazine reporting?	American English	American English ✓
ivd)	*What language are Garowe Principles written in?*	-	English English ✗

fitted. ii) displays cases where the entire output is incorrect, and factual failures indicate difficulties in retaining memory of parameters for some rare entities (such as Persian iia, iib). iv) presents cases of generalization failure, for example in ivd), where the model answered "English" but did not fully follow the ground truth, indicating significant room for improvement in the accuracy of generalized edits. Meanwhile, in iii) we surprisingly find that even when WISE errs on the Edit Prompt, it can correctly answer its paraphrase $iiib$) *"The kind of voice of Gemma Bosini is what?"*. This indicates that WISE can handle contextual information correctly in some cases but falls short in specific editing instructions, suggesting that optimizing editing instructions (modifying the editing context) may be a direction for improvement.

C.6 Importance of *Knowledge Anchor* When Merging Models

Table 10: Analysis of Merging *w.o.* and *w.* "knowledge anchor" (KA). $T = 1000$. ZsRE. `LLaMA-2-7B`.

ρ/k	*w.o.* KA				*w.* KA			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
2/0.30	0.76	0.72	1.00	0.83	**0.79**	**0.73**	1.00	**0.84**
2/0.50	0.74	**0.73**	1.00	0.82	**0.77**	0.72	1.00	**0.83**
3/0.33	0.72	0.68	1.00	0.80	**0.75**	**0.71**	1.00	**0.82**
5/0.20	0.64	0.61	1.00	0.75	**0.73**	**0.68**	1.00	**0.80**

Here, we discuss the effects of independent (ensured by non-overlapping masks) vs partially overlapping parameters within MLP subspaces on editing performance, as shown in Table 10. It is observable that, despite varying mask ratios ρ and the number of subspaces k, partial overlap (w. KA) consistently outperforms independent configurations (w.o. KA) in terms of Reliability (Rel.) and Generalization (Gen.). For example, at ρ/k of 5/0.20, there is a relative improvement of 9% and 7% respectively. This demonstrates that the overlapping regions contribute as "anchors" for knowledge fusion, facilitating information transfer across different subspaces. Moreover, the shared parameters provide a natural regularization (Akyürek et al., 2022) mechanism, helping synchronize model behavior across different subspaces.

C.7 Scale Up to 3K of Edits.

Figure 11: **Scaling to 3K edits of ZsRE.** `LLaMA-2-7B`.

Method	$T = 2000$				$T = 3000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GRACE	**0.96**	0.03	1.00	0.66	**0.96**	0.03	1.00	0.66
MEMIT-MASS	0.64	0.58	0.55	0.59	0.58	0.53	0.47	0.53
WISE-Merge	0.66	0.63	1.00	0.76	0.58	0.56	1.00	0.71
WISE-Retrieve	0.68	**0.64**	**1.00**	**0.77**	0.61	**0.58**	**1.00**	**0.73**
WISE-Retrieve$_{\text{oracle}}$	0.77	0.72	1.00	0.83	0.75	0.70	1.00	0.82

We scale the number of continual edits to 3K in Table 11. We compare WISE-Merge, keeping one side memory by multi-time merging, and

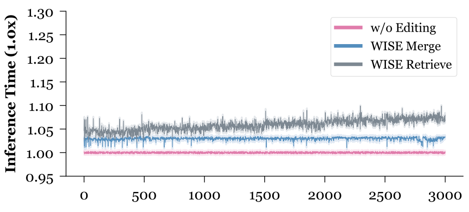


Figure 12: **Inference time of WISE when varying T.** ZsRE, `LLaMA-2-7B`.

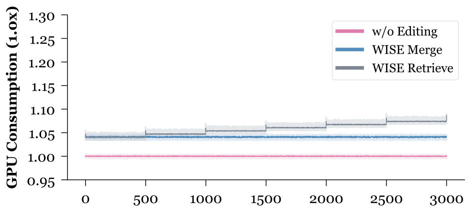


Figure 13: **GPU VRAM of WISE when varying T.** ZsRE, `LLaMA-2-7B`.

WISE-Retrieve, keeping several side memories by routing and retrieving among different side memories. For WISE-Retrieve, we show an upper bound "*oracle*", which always identifies the correct routing path. We observe that the WISE series maintains high scalability, consistently outperforming the strongest baselines including MEMIT-MASS and GRACE. WISE-Retrieve based on top-1 activation retrieval demonstrates the best results in 3K edits, showing the effectiveness of well-organized memory subspaces and routing strategies during editing. We note that the "*oracle*" exhibits marginal performance decline when scaling the edits from 2K to 3K, yet it demonstrates remarkable performance across all metrics. This underscores the potential of WISE to handle extremely long continual edits, contingent upon substantial improvement in the retrieval of side memories. Additionally, an appropriate replay of edits can further improve retrieval accuracy, as detailed in Appendix C.4.

C.8 Computational Costs Analysis of WISE.

Figure 12 shows the inference time of a single instance for LLaMA after $t \in [0, 3000]$ editing steps, measured across 10 trials of each setting. Consistent with our expectations, we find that WISE-Merge incurs a constant inference delay (about 3%) as the editing stream expands. WISE-Retrieve, due to the introduction of retrieval routing, shows an increase in inference time as the number of edits increases, with a time cost increment of about 7% after 3K edits. Knowledge merging ensures that WISE-Merge only brings constant additional costs (0.64% extra parameters and 4% extra GPU VRAM, as detailed in Figure 13), contrasting with past memory-based works that continuously demand more available memory (Hartvigsen et al., 2023; Mitchell et al., 2022b).

C.9 Pseudo Code of WISE

The pseudo-code of the WISE editing stage is in Algorithm 1, and the one of the WISE inference stage is Algorithm 2.

D Proof of Theorem 2.1

Theorem D.1 ***Subspace Overlap.*** *Generate k memory subspaces $\mathbf{W}_{v'}^{i}, i \in [k]$ by random mask with 1's ratio ρ, so each memory has $\rho \cdot |\mathbf{W}_{v'}|$ active trained parameters. For any two subspaces $\mathbf{W}_{v'}^{i}$ and $\mathbf{W}_{v'}^{j}$ $i \neq j; i, j \in [k]$, there are $\rho^2 \cdot |\mathbf{W}_{v'}|$ active parameters that are overlapped. For all k subspaces, there are $\rho^k \cdot |\mathbf{W}_{v'}|$ overlapped active parameters.*

Proof: We aim to prove the Subspace Overlap theorem by induction.

Let $\mathbf{W}_{v'}^{i}$ represent the i-th memory subspace generated by a random mask with a sparsity ratio of ρ, where $i \in [k]$. Each memory subspace $\mathbf{W}_{v'}^{i}$ contains $\rho \cdot |\mathbf{W}_{v'}|$ active trained parameters.

We start by considering the case of two memory subspaces, $\mathbf{W}_{v'}^{i}$ and $\mathbf{W}_{v'}^{j}$, where $i \neq j$ and $i, j \in [k]$.

Let $P(\text{parameter sampled}) = \rho$ be the probability that a parameter is sampled in one mask generation event.

1. For a single mask generation, the probability that a specific parameter is sampled is ρ. We denote this probability as $P(\text{sampled}) = \rho$.

2. Considering two independent mask generation events, the probability that the same parameter is sampled in both masks is the product of their individual probabilities, i.e., ρ^2. This is derived from the independence of the events. Mathematically:

$$P(\text{sampled in both masks}) = \rho \times \rho = \rho^2.$$

3. Extending this logic, for k independent mask generation events, the probability that a specific parameter is sampled in all k masks is ρ^k. Mathematically:

$$P(\text{sampled in all } k \text{ masks}) = \rho^k.$$

Algorithm 1: WISE Editing Stage

Input: The initial LLM model f_{Θ_0}, the targeted FFN layer, the edit dataset $\mathcal{D}_{\text{edit}}$ whose length is T, the irrelevant dataset $\mathcal{D}_{\text{irr}}$, the subspace mask ratio ρ, the number of subspaces k, whether WISE-Retrieve.

Output: The final LLM model f_{Θ_T} after T edits.

```
1:  Generate $k$ random masks $\mathbf{M}_i, i \in [k]$ of ratio $\rho$; if WISE-Retrieve, copy the side memory several times;
2:  for each edit $(\mathbf{x}_t, \mathbf{y}_t) \in \mathcal{D}_{\text{edit}}, t \in [T]$ do
3:    Edit $(\mathbf{x}_t, \mathbf{y}_t)$ in the corresponding memory subspace by $L_{\text{edit}} = -\log P_{W_{v'}}(\mathbf{y}_t|\mathbf{x}_t) + L_a$;
4:    Update the activation threshold: $\epsilon = \min(\epsilon, \Delta_{\text{act}}(\mathbf{x}_t))$;
5:    if All the $k$ subspaces of a side memory are full then
6:      Use Ties-Merge in Equation 7 to update the final side memory;
7:      if WISE-Retrieve then
8:        Move to another copy of side memory $\mathbf{W}_{v'}$;
9:      end if
10:   else
11:     if Current subspace $\mathbf{M}_i$ is full then
12:       Move to another subspace of side memory $\mathbf{M}_{i+1}$;
13:     end if
14:   end if
15: end for
16: return Obtain the final LLM model $f_{\Theta_T}$.
```

Algorithm 2: WISE Inference Stage

Input: The edited LLM model f_{Θ_T}, the activation threshold ϵ, the test dataset $\mathcal{D}_{\text{test}}$, whether WISE-Retrieve.

Output: The model's output.

```
1:  for each query $\mathbf{x}_i \in \mathcal{D}_{\text{test}}$ do
2:    if WISE-Retrieve then
3:      Get the value of activation $\Delta_{\text{act}} = \|\mathcal{A}(\mathbf{x}_i) \cdot (\mathbf{W}_{v'} - \mathbf{W}_v)\|_2$ for each side memory and select the one with the maximal value of $\Delta_{\text{act}}$;
4:    else
5:      Get the value of activation $\Delta_{\text{act}} = \|\mathcal{A}(\mathbf{x}_i) \cdot (\mathbf{W}_{v'} - \mathbf{W}_v)\|_2$;
6:    end if
7:    if $\Delta_{\text{act}} > \epsilon$ then
8:      Use the side memory $\mathbf{W}_{v'}$ to generate the output as in Equation 5;
9:    else
10:     Use the main memory $\mathbf{W}_v$ to generate the output as in Equation 5.
11:   end if
12: end for
```

Now, let's calculate the number of parameters overlapped in two random masks:

The total number of parameters in $\mathbf{W}_{v'}$ is $|\mathbf{W}_{v'}|$.

Thus, the number of parameters overlapped in two random masks, $\mathbf{W}^i_{v'}$ and $\mathbf{W}^j_{v'}$, is $\rho^2 \cdot |\mathbf{W}_{v'}|$.

Extending this to k random masks, the number of parameters overlapped in all k masks is $\rho^k \cdot |\mathbf{W}_{v'}|$.

This concludes the proof. □

E Related Works

Memory and Knowledge Injection of LLMs The memories of LLMs can be divided into long-term (episodic) memory and working memory (short-term) (Li et al., 2023a; Park and Bak, 2023; Yogatama et al., 2021). Long-term memory refers to the knowledge stored in the model's parameters, which can be updated by (re)pretraining (Radford et al.), finetuning (Hu et al., 2021), and model editing (Yao et al., 2023). Working memory is stored in sustained activations/representations of neurons, which will be awakened during inference time (Li et al., 2023a). In-context learning (ICL) is a kind of working memory (Wei et al., 2022), also along with retrieval-based editing methods like GRACE (Hartvigsen et al., 2023). How to reinforce memory and inject/update knowledge for LLMs is a fundamental question (Wang et al., 2024; Ovadia et al., 2023; Mosbach et al., 2023). ICL or finetuning? Different works show different conclusions. In (Mosbach et al., 2023), the authors find that few-shot finetuning is more generalizable than ICL, especially for out-of-distribution data. In (Ovadia et al., 2023), the authors contrast finetuning with retrieval-augmented generation (RAG) in terms of knowledge injection and find that RAG is better in most cases, and combining both will produce the best results. However, finetuning and pretraining are computation-expensive (Touvron et al., 2023; Hartvigsen et al., 2023) and usually suffer from catastrophic forgetting (Luo et al., 2023) and overfitting (Tirumala et al., 2022). For ICL and RAG, the working memory is sometimes not controllable, the model may not follow the information of the contexts (Li et al., 2023a), and the context window is limited (Mohtashami and Jaggi, 2023; Munkhdalai et al., 2024), and there are works addressing these issues by training controllable ICL (Li et al., 2023a), long-context techniques (Mohtashami and Jaggi, 2023; Munkhdalai et al., 2024), and recurrent memory architecture design (Wang et al., 2024). SPALM is proposed to add language models with storage modules that resemble both working and long-term memories (Yogatama et al., 2021). Also, model editing is an emerging technology that aims to enable data-efficient, fast, and non-destructive knowledge updates to LLMs (Yao et al., 2023; Zhang et al., 2024).

Model Editing of LLMs Model editing can be summarized as the following lines of research. ***Constrained finetuning:*** Preliminary model editing uses constrained finetuning to update parameters based on new examples (Sotoudeh and Thakur, 2019; Rawat et al., 2021). ***Locate-and-edit:*** ROME (Meng et al., 2022) locates the factual associations in autoregressive LLMs and conducts accurate and efficient edits by taking MLPs as key-value memories. Then, MEMIT (Meng et al., 2023) extends ROME from single-editing to mass-editing. COMEBA-HK (Li et al., 2024a) identifies the Local Editing Scope and extends MEMIT for sequential editing. In addition, T-Patcher (Huang et al., 2023) targets the last feed-forward layer of LLMs, adding an additional neuron for each edit. ***Meta learning:*** Recent meta-learning methods use hypernetworks for aiding editing. MEND (Mitchell et al., 2022a) learns a hypernetwork that can decouple the finetuning gradients into the gradient updates that generalize the edits and won't damage the performances on unrelated inputs. To remedy the cancellation effect of MEND, MALMEN (Tan et al., 2023) uses hypernetwork to produce the weight shifts of editing and formulates the weight shift aggregation as the least square problem. ***Retrieval-based methods:*** Instead of directly editing the model parameters, retrieval-based methods aim to improve the working memory of LLMs to enable model editing. IKE (Zheng et al., 2023) uses context-edit facts to guide the model when generating edited facts. DeCK (Bi et al., 2024) employs contrasting knowledge decoding, which enhances the confidence of in-context-based editors in the edited facts. SERAC (Mitchell et al., 2022b) (a modified version dubbed as DEFER (Hartvigsen et al., 2023)) records edit items in a file and trains additional scope classifier and counterfactual model to detect, retrieve, and generate the edit-related results. Though the editing retriever and generator are neural networks, they are too small to have the power of LLMs. GRACE (Hartvigsen et al., 2023) adopts a discrete codebook of edits for retrieving and replacing the edits' layer representations during inference. From single editing (Meng et al., 2022) to mass editing (Tan et al., 2023; Meng et al., 2023), and from static editing to sequential (Huang et al., 2023) (continual) or lifelong editing (Hartvigsen et al., 2023), model editing is developing to meet more realistic demands.

Model Merging Model merging (Goddard et al., 2024), also known as model fusion (Li et al., 2023b; Singh and Jaggi, 2020), studies how to aggregate different models' knowledge into one by parameter merging. However, in the research of linear mode connectivity, it is found that different minima of neural networks can hardly be merged into a generalized one even if trained on the same datasets from the same initialization (but with different random seeds) (Entezari et al., 2022; Ainsworth et al., 2023). The main reason is considered to be the permutation invariance property of deep neural networks, which means that the positions of neurons can be permuted without affecting the network function (Entezari et al., 2022); as a result, different minima reside in different loss basins (Ainsworth et al., 2023). To improve linear mode connectivity and model merging, methods like optimal transport (Singh and Jaggi, 2020; Imfeld et al., 2024), re-basin (Ainsworth et al., 2023), and training-time alignment (Li et al., 2024b) are developed. For the applications, model merging techniques can help to improve the generalization of federated learning (Li et al., 2023c; Wang et al., 2020) and enable knowledge aggregation of different-task models in a task arithmetic way (Ilharco et al., 2023; Ortiz-Jimenez et al., 2024). Recently, methods like task arithmetic in tangent space (Ortiz-Jimenez et al., 2024), TIES-Merging (Yadav et al., 2023), ZipIt! (Stoica et al., 2024), and ColD fusion (Don-Yehiya et al., 2023) have been proposed for deep model fusion of pretrained foundation models, such as CLIP, ViT, and large language models. Specifically, TIES-

Merging (Yadav et al., 2023) consists of trim, elect sign & merge pipeline, which inspires the merge process of side memories in our paper.

Continual Learning Continual learning (Shi et al., 2024; Wu et al., 2024) tackles the catastrophic forgetting problem in deep learning models with new knowledge (De Lange et al., 2021), and recent research has focused on various methods in this area. One such method is continual finetuning, where LLMs are refined over time with the arrival of new instances. For instance, a comprehensive study by (Lin et al., 2022) explores continual finetuning extensively. However, it has been observed that regularizing finetuning with continual learning techniques such as Elastic Weight Consolidation (Kirkpatrick et al., 2017), Experience Replay (Rolnick et al., 2019), and Maximally Interfered Replay (Aljundi et al., 2019) can lead to a rapid decay in performance on previous tasks, although it aids in retaining some memory of past inputs. This suggests that editing, as opposed to vanilla continual finetuning, presents unique challenges, especially considering that edits are unlikely to be evenly distributed (Henn et al., 2021). One promising direction within the realm of continual learning is the adoption of key-value methods, inspired by advancements in computer vision (Liu et al., 2021; Van Den Oord et al., 2017). Recent studies have showcased the effectiveness of continual prompt-learning for NLP (Wang et al., 2022a,b), particularly in applications like text retrieval (Xiong et al., 2020). Notably, discrete key-value methods have been shown to excel in handling shifting distributions (Träuble et al., 2023), with some recent efforts extending their application to question answering (Dai et al., 2022). These methods cache values to ensure that inputs remain within the distribution for downstream encoders, thus facilitating the incorporation of longer-term memory, provided there are adequate computational resources.

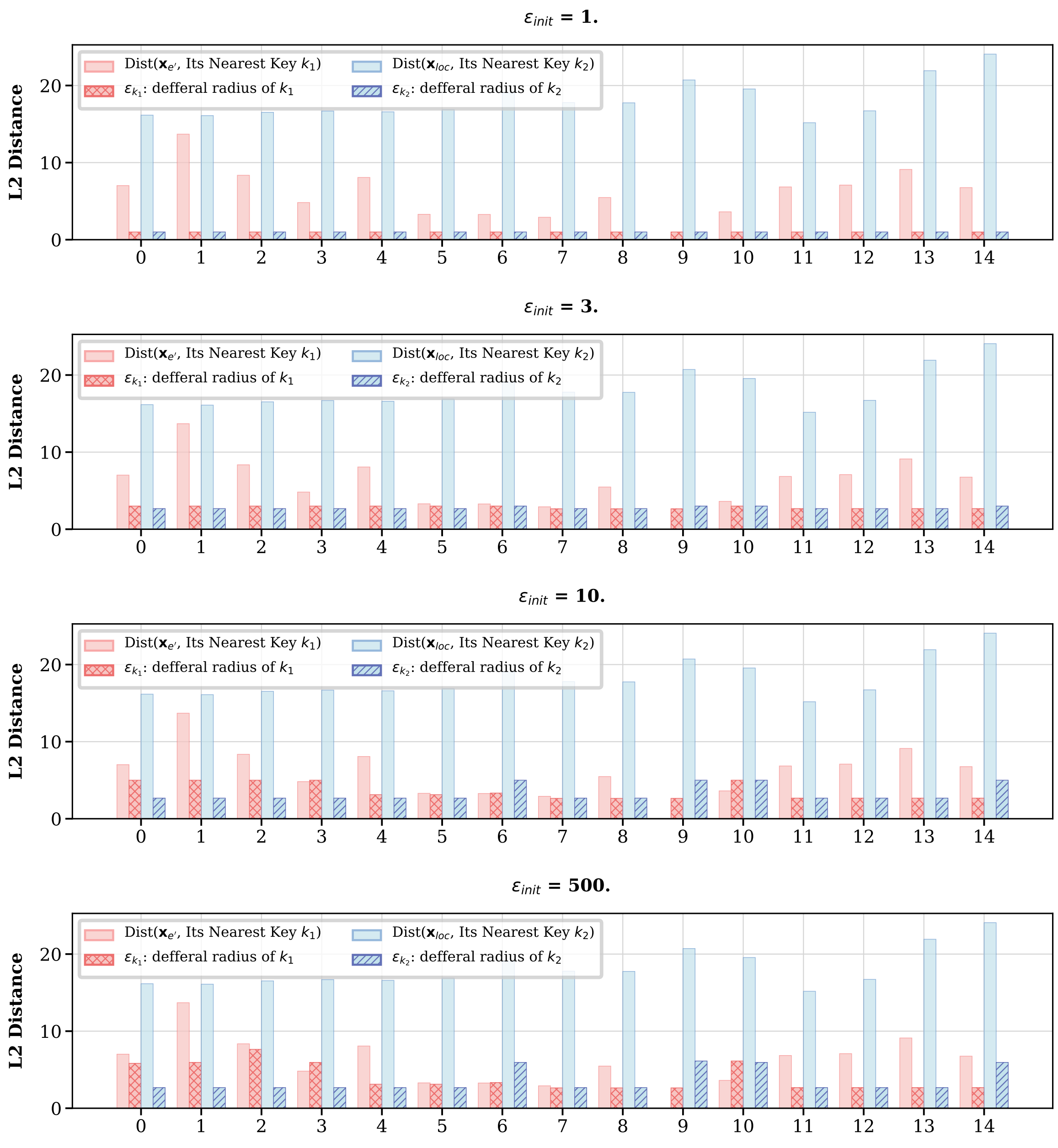


Figure 14: Investigation on the query $\mathbf{x}$ and its distance to the nearest Key k, as well as the *deferral radius* ϵ of that Key. Red and Blue respectively represent the paraphrase query $\mathbf{x}_{e'}$ and the unrelated query $\mathbf{x}_{\text{loc}}$, with the hatch representing the radius of the nearest Key. We observe that when conflicts occur (hit the codebook Key but with different Edit Target $\mathbf{y}_e$), the *deferral radius* ϵ decays exponentially. This results in GRACE being unable to encompass the paraphrase $\mathbf{x}_{e'}$ and maintain high locality, regardless of how ϵ_{init} is adjusted. ZsRE, `LLaMA-2-7B`.