Can We Continually Edit Language Models? On the Knowledge Attenuation in Sequential Model Editing

Qi Li [1], **Xiaowen Chu** [1] †
[1] The Hong Kong University of Science and Technology (Guangzhou)
qili@hkust-gz.edu.cn

Abstract

Model editing has become a promising approach for precisely and effectively updating knowledge within language models. In this paper, we investigate *knowledge attenuation*, in which the retention of updated knowledge within the language model decreases as the number of edits increases after *sequential* editing. Through empirical study, we discovered that existing editing methods generally suffer from knowledge attenuation. We attribute this phenomenon to two aspects: (1) redundant parameters interference and (2) update weight disentanglement. To this end, we propose the AdaPLE method. It not only mitigates the knowledge attenuation issue but also improves the performance on existing benchmarks. To the best of our knowledge, we are the first to investigate the cause and mitigation of knowledge attenuation in sequential LLM editing.

1 Introduction

Recently, large language models (LLM) like Llama (Touvron et al., 2023), Falcon (Almazrouei et al., 2023), and Bloom (Workshop et al., 2022) have shown impressive capabilities in generating human-like text, improving performance on a variety of natural language processing tasks. Though LLM is capable of performing tasks based on accumulated knowledge (Zhong et al., 2021), they are prone to generating erroneous, harmful, or outdated information. Thus the ability to maintain fresh and customized information is desirable in many scenarios. Due to hardware and resource constraints, researchers seek methods to update knowledge inside LLM efficiently. Recent years have witnessed a growing number of precise yet efficient model editing techniques like MEND (Mitchell et al., 2022), MEMIT (Meng et al., 2022b), and PMET (Li et al., 2023b), as shown in Figure 1, thus circumventing prohibitive fine-tuning process.

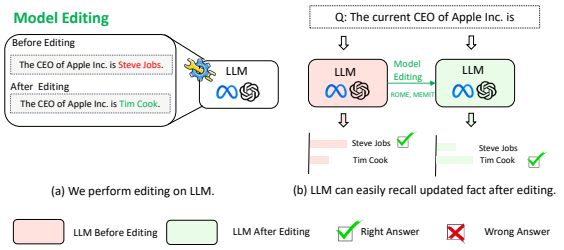


Figure 1: **Illustration of language model editing.** (a) Edit the autoregressive language model to correct outdated knowledge. (b) Post-edit language models can easily recall edited knowledge.

In this paper, we focus on a specific model editing scenario termed as *sequential editing* (Yao et al., 2023): editing samples arrive in sequence, and we incrementally apply the editing weights to the edit target layer within the language model in a continuous, step-wise manner other than roll back the update before next edit. While successful in editing, we will show that current editing methods may incorporate *unexplored* defects in sequential editing termed as *knowledge attenuation* (KA): As the number of edits increases, the retention of updated knowledge significantly deteriorates across all criteria, as shown in Figure 2. The edit weights during these editing processes comprise millions of parameters, making them excessively redundant for representing the knowledge that needs to be updated. We *hypothesize* that these redundant weights may contain interfering information or lead to mutual interaction, which, when sequentially adding the update weights corresponding to different knowledge representations to the model's editing target, can damage the underlying knowledge structure within the model and lead to a decrease in the retention of edited information. This flaw can undermine the adoption of these methods, as it may erode their utility and trustworthiness in real-world applications. In particular, there is a lack of rigorous understanding on:

†Corresponding author

Findings of the Association for Computational Linguistics: ACL 2024, pages 5438–5455

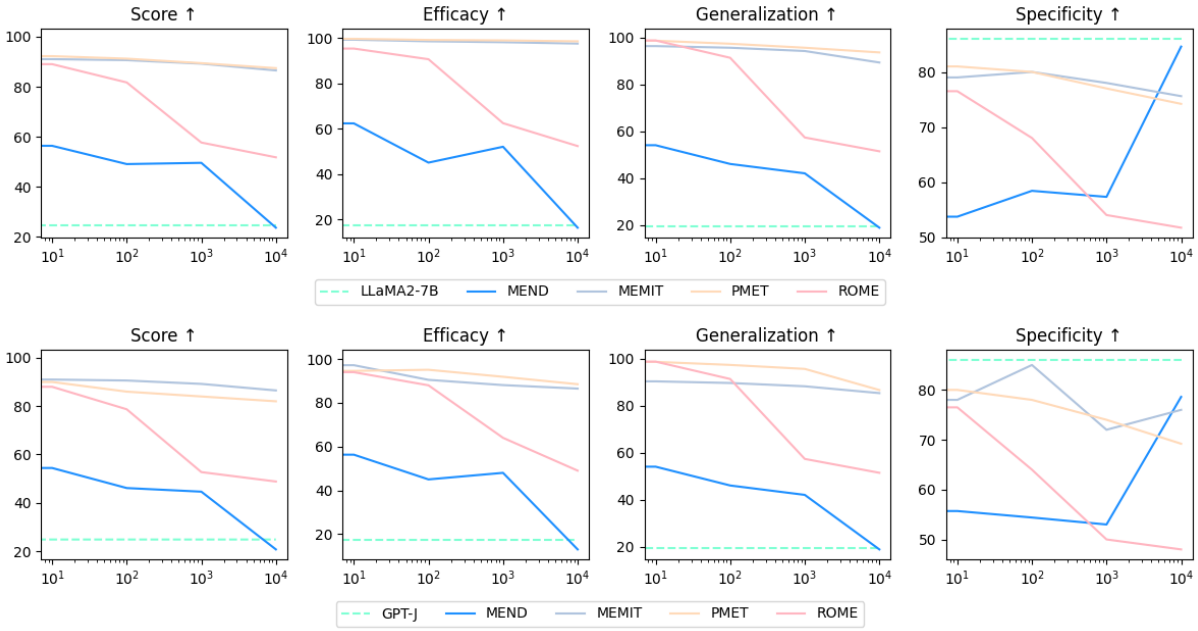


Figure 2: **Scaling edit performance for LLaMA2-7B (above) and GPT-J (below) on COUNTERFACT dataset**. As illustrated, with an increase in the number of edits, the retention of updated knowledge significantly deteriorates across all criteria, a phenomenon we refer to as *knowledge attenuation*. Through intensive empirical study, we observe that existing editing methods like MEND, ROME, MEMIT generally suffer from this issue.

(1) which factors are important for knowledge attenuation in model editing ?
(2) how to address the knowledge attenuation issue effectively by considering these factors ?

To close this gap, we systemically investigate the knowledge attenuation in sequential language model editing through intensive empirical studies. We inspect the updated weight for the edit layer from the lens of redundant parameter and weight entanglement, finding that the *knowledge attenuation* can stem from two major causes (1) *interference from redundant parameter* and (2) *disentanglement between edit weight*, both of which can lead to deteriorate in multiple edit performance.

To this end, we introduce AdaPLE , *Adaptive Precise LLM Editing*, an effective model editing method that can alleviate knowledge attenuation. The proposed AdaPLE achieves more precise model editing by integrating refined optimization goals, concurrently optimizing the hidden states of the MHSA and FFN other than layer hidden states, as well as employing a superior method for integration of editing weights to edit targets. To validate its effectiveness, we conduct intensive experiments with different language models across diverse datasets. Results demonstrate that our proposed AdaPLE enables effective scaling performance while keeping a comparable or even superior performance to current editing methods. Further visualization shows that our proposed AdaPLE can result in a significant weight disentanglement compared with existing editing methods, thus resulting in a better scaling performance.

To the best of our knowledge, we are the *first* to investigate the knowledge attenuation in language model editing. The main contributions of this paper are as follows:

(1) We systematically investigate the knowledge attenuation in sequential language model editing and dissect the main causes of KA (In Section 3).

(2) We introduce AdaPLE , a scalable, precise model editing method for better sequential editing (In Section 4).

(3) Experimental results show that AdaPLE can reach comparable or superior performance while mitigating knowledge attenuation (In Section 5).

2 Preliminary

In this section, we provide comprehensive preliminaries of sequential model editing.

2.1 Basic Notation

We denote the language model as M_θ, where θ signifies the model's parameters. Knowledge tuple t is a triplet (s, r, o), where s is subject, r is relation,

and o is object. Typically, a language model is stacked by L identical layers (Vaswani et al., 2017), which consist of MHSA and FFN sub-block. The FFN comprises two linear layers, denoted as W_{in}, W_{out} respectively.

2.2 Model Editing

Model editing aims to adjust a model M_θ behavior on some facts with fine-grained without influencing unrelated samples. Current works focus on editing knowledge tuple $t = (s, r, o)$. The editing process inserts new tuples (s, r, o^*) in place of the current tuple (s, r, o), where these two share the same s and o. An editing operation is denoted as $e = (s, r, o, o^*)$ for brevity. Given n fact tuples $T^* = (t_1^*, t_2^*, \ldots, t_n^*)$ where $t_i^* = (s_i, r_i, o_i^*), i = 1, 2, \ldots, n$, and a model M_θ, model editing yields an edited language model M_θ^e via editing operations $\mathcal{E} = \{e_1, e_2, \ldots\}$, where $M_\theta^e(s_j, r_j) = o_j^*$ if $t_j = (s_j, r_j, o_j^*) \in T^*$, else $M_\theta^e(s_j, r_j) = o_j$. To evaluate model editing methods, current works focus on three dimensions: *efficacy*, *generalization*, and *locality* (Yao et al., 2023).

2.3 Formal Definition of Sequential Editing

Here, let's provide a formal definition of sequential editing. Assume we have an unedited model M_0, and n editing samples(x_i, y_i), where $i = 1, 2, \ldots, n$ need to be incorporated into the language model M_0. Suppose the editing operation is a function $E(\cdot, \cdot)$, where the first parameter is the model to be edited and the second parameter is the editing samples. Assume we get the edited model M_i after the i-th editing operation. In sequential editing, M_t (model parameter after the t-th editing) is determined by the model weight M_{t-1} and the editing sample used in the t-th edit, like $M_t = E(M_{t-1}, S_t)$, where S_t is the edit samples used in the t-th edit. For different index i and j, $S_i \cap S_j = \emptyset$; for every i , we have $\bigcup_i S_i = \{x_j, y_j\}_{j=1}^{j=n}$. If we denote the size of S_t is n_t, it satify $n_t \geq 1$ for every t and satisfies $n = \sum_t n_t$ for all edit batches.

2.4 Locate-then-Edit Style Model Editing

Locate-then-Edit style editing methods view downsample component W_{out}^l of FFN in LLM layers as associative memory, denoted as W. The details of transformer architecture can be found in (Vaswani et al., 2017). Thus we have $Wk_i = v_i$. Here, $k_i \triangleq k_i^l$ and $v_i \triangleq v_i^l$ represent the sets of keys and values encoding the knowledge tuple t_i in the l-th layer. We can stack above equation as matrix form like $WK \approx V$, where $K \triangleq [k_1 \mid k_2 \mid \cdots \mid k_n]$ and $V \triangleq [v_1 \mid v_2 \mid \cdots \mid v_n]$. That means $W_0 \triangleq \text{argmin}_{\hat{W}} \sum_{i=1}^n \left\| \hat{W} k_i - v_i \right\|^2$. MEMIT (Meng et al., 2022b) optimizes an objective to insert new key-value tuples :

$$W_1 \triangleq \underset{W}{\text{argmin}} (\underbrace{\sum_{i=1}^{n} \|Wk_i - v_i\|^2}_{\text{(a) original keys and values}} + \underbrace{\sum_{i=n+1}^{n+u} \|Wk_i - v_i\|^2}_{\text{(b) inserted keys and values}}). \tag{1}$$

The (a) term in Equation 1 indicates that n original pieces of knowledge, while the (b) term indicates u pieces of new. They consider the target weight W_1 as the sum of the original weight W_0 and the incremental weight Δ. The close-form solution of edit target Δ is :

$$\Delta = RK_1^T (C_0 + K_1 K_1^T)^{-1}, \tag{2}$$

where $R \triangleq V_1 - W_0 K_1$ represents the residual between the values V_1 (namely target knowledge representations) corresponding to the keys K_1 of the target knowledge and the model original knowledge $W_0 K_1$. $C_0 \triangleq K_0 K_0^T = \lambda \cdot \mathbb{E}_k \left[kk^T \right]$ is an estimate of previously memorized keys obtained through sampling. Here, λ is a hyper-parameter that balances the modification and preservation.

2.5 Sequential Editing Performance

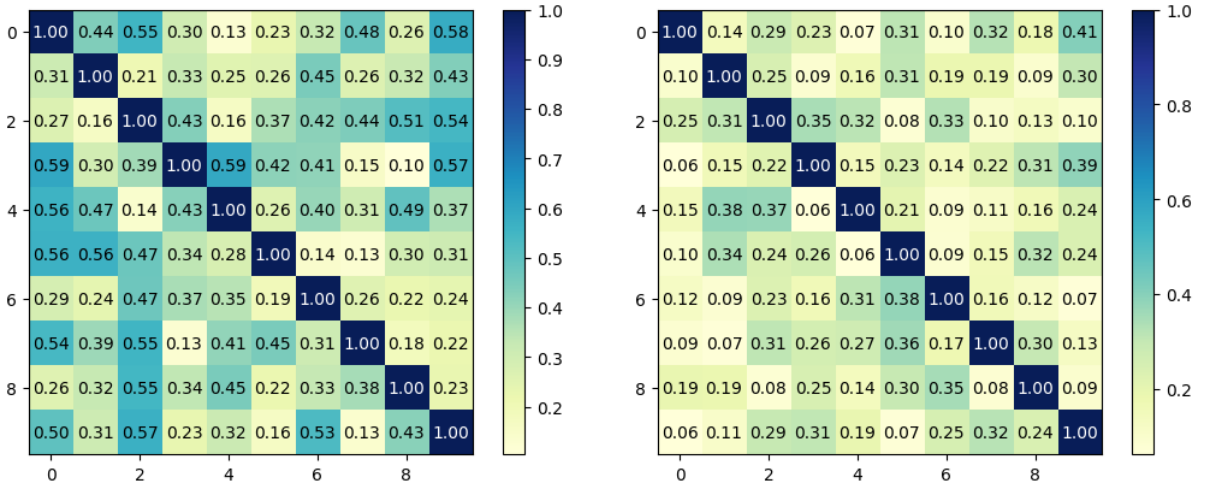
The existing protocol for evaluating model editing includes several key steps: updating a model's weight, evaluating the model after this update, and reverting the model to its pre-update weight for subsequent tests. However, in practical scenarios such as sequential editing, it is essential for models to preserve previous edits before incorporating new ones. Consequently, enhancing the scalability of editing capabilities is vital. This paper primarily investigates the robustness of various methods in sequential editing settings.

3 A Comprehensive Study of KA

In this section, we present a comprehensive study of knowledge attenuation after sequentially editing the language model. We first give a geometric intuition for sequential editing, then we perform a series of investigations to explore the primary factors of knowledge attenuation.

3.1 Geometric Intuition and Insight

Inspired by recent work like (Ilharco et al., 2022; Ortiz-Jimenez et al., 2023; Tang et al., 2023), it



(a) Edit weight similarity visualization.

(b) Edit weight similarity visualization after DR.

Figure 3: **Edit Weight similarity visualization before and after DR.** As we can see from the above figures, the pairwise similarity of edit weight significantly decreases after drop and re-scale, indicating that weight disentanglement can help mitigate knowledge attenuation.

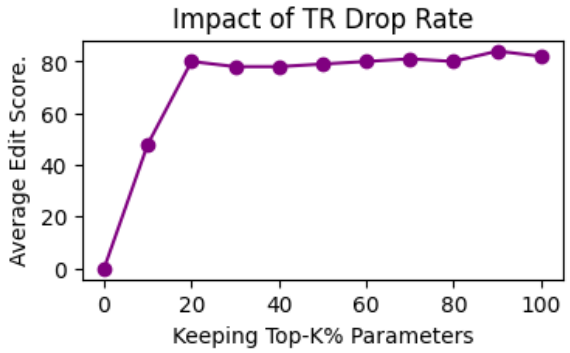


Figure 4: **Impact of top k% large parameters retain on edit performance.** Only retaining at least 20% of high-magnitude parameters does not degrade the223 overall editing performance.

is generally believed that the less similar (closer orthogonal) between edit weight, the less interference between sequentially added update weight and the better scaling edit performance. The intuition is that the closer orthogonal edit weight indicates that the specialized knowledge captured for each edit sample lies in distinct subspaces with minimal overlap or redundancy. This enables better preservation of inherent knowledge structure and avoids destructive interference during sequential language model editing. We compute the pairwise similarity between 10 update weights for edit samples computed by MEMIT, as shown in Figure 3a. These edit weights are similar to some extent.

On the other hand, when sequentially adding an editing weight Δ that is influential for knowledge representation but redundant like ROME or MEMIT to edit targets, the editing performance significantly drops. We hypothesize that the influential value may be obscured by the redundant parameter, lowering the overall editing performance or deconstructing the inherent knowledge structure of the language model. This is aligned with observation in work (Yadav et al., 2023b).

3.2 Further Investigation

In this section, we reveal that redundant parameters interference and update weight disentanglement are two primary factors of knowledge attenuation through empirical study. We use an LLaMA2-7B (Touvron et al., 2023) and GPT-J (Wang and Komatsuzaki, 2021) model as the experimental backbone. The experimental data is sampled from COUNTERFACT dataset.

Parameter redundancy of update weight. Recent works have shown that, for fine-tuned models, most model parameters can change over the adaption process but only have a small impact on performance. Does this hold for sequential language model editing? To validate this, we randomly select some edit samples and compute update weight Δ with MEMIT (Meng et al., 2022b). For each update weight, we keep only the largest - topk% parameters and set the other parameters as zero. Results in Figure 4 show that only retaining at least top-20% of high-magnitude parameters does not degrade the overall editing performance. We conduct experiments on LLaMA2-7B and GPT-J with 100, 200,

and 500 samples with MEMIT, and the results are consistent the same.

Observation 3.1. Up to 80% parameters in edit weight are redundant.

Redundancy parameters lead to interference. To explore this, We select $1k$ edit samples and perform sequential editing with three different batch settings: 10, 100, and 500 on different language model for MEMIT. In integrating the update weight into the FFN of edit target, we employed two distinct fusion methods: (1) a straightforward addition (referred to as "vanilla"), and (2) retaining top 20% largest parameters of the update weight, setting the remaining to 0 (referred to as "TR", top-retain). The results indicate that the TR can mitigate knowledge attenuation in sequentially edited language models to some extent. This conclusion holds across experiments conducted with different batches and models.

Observation 3.2. Sparsifying update weights can mitigate redundant parameter interference.

Randomly drop leads to better disentanglement. Can sparsifying update weights by retaining the top 20% highest values thoroughly address the issue of knowledge decay in sequential editing? The answer is negative. Our analysis involved calculating and visualizing the cosine similarity among 20 sets of update weights, each pruned to retain the top 20% of large parameters. We observed that the similarity among these pruned update weights did not significantly decrease, indicating that merely sparsifying update weights by retaining the top 20% highest values are insufficient to solve the problem. Consequently, we adopted a novel approach termed DR (Drop and Scale): we randomly discard parameters of update weights with probability p, setting the discarded values to zero, and then scale the remaining weights by $1/(1-p)$ to maintain the expected value of update weights. Following DR in our sequential editing as above, we discovered that the edited models were effectively able to mitigate knowledge attenuation. We calculated and visualized the pairwise similarity among 20 sets of update weights processed by DR in Figure 3b, revealing that the similarity between the processed weights was significantly reduced.

Observation 3.3. Randomly dropping can reduce entanglement and similarity between edit weights.

4 Proposed Method: AdaPLE

In this section, we introduce AdaPLE , an effective editing method for mitigating the KA in language model editing. It can effectively mitigate knowledge attenuation by introducing adaptive merging of update weight. Our proposed AdaPLE follows the locate-then-edit style and composes two phases: (i) computing edit weight and (ii) adaptive merging of edit weight. In the computing stage, AdaPLE first computes update weight within an optimization process (In section 4.1). Once computed, the update weights are merged to edit targets during the adaptive merging process (In section 4.2). The pseudo-code is in Algorithm 1. For more details, please refer to Appendix C.

4.1 Computing Edit Weight

The AdaPLE follows the Locate-then-Edit style and regards FFN in language model layer as two-layer key–value memories, where the up-sample block W_{in}^l forms a key, with which the down-sample block W_{out}^l retrieves an associated value in layer l. If we regard W_{out}^l as associative memory, denoted as W_0. We have $W_0 k_i = v_i$ where $k_i \triangleq k_i^l$ and $v_i \triangleq v_i^l$ $(i = 1, 2, \ldots, n)$, which encodes inherent knowledge inside LLM with key-value form as knowledge representation.

To perform editing, we add update weight Δ to W_0 and get $W_1 = \Delta + W_0$. For W_1, it satisfies $W_1 k_m = v_m$ where (k_m, v_m) is representation of the unedited knowledge tuple (s_m, r_m, o_m) and $W_1 k_n = v_n^*$ where (k_n, v_n^*) is representation of new knowledge tuple (s_n, r_n, o_n^*) that we want to update. To get update weight Δ, we view the editing process as optimizing a new objective function.

AdaPLE views the editing process as an optimization problem like locate-then-edit style methods. However, existing methods don't modify u pieces of existing fact, but rather directly insert extra u new triplets to incorporate new information. The editing process should involve both forgetting outdated or incorrect knowledge and inserting the new. Thus, we alter the optimization goal

of AdaPLE to:

$$W_1 \triangleq \underset{W}{\operatorname{argmin}}[\underbrace{\sum_{i=1}^{n-u} \|Wk_i - v_i\|^2}_{\text{(a) unedited knowledge}} + \underbrace{\sum_{i=n-u+1}^{n} \|Wk_i - v_i^*\|^2}_{\text{(b) inserted new knowledge}} - \underbrace{\sum_{i=n-u+1}^{n} \|Wk_i - v_i\|^2}_{\text{(c) erased old knowledge}}] \tag{3}$$

Here, k_i and v_i $(i = 1, 2, ..., n-u)$ indicate the original key and value of knowledge tuple $t = (s, r, o)$, and v_i^* refers to the target value where $i = n-u+1, n-u+2, \ldots, n$. (Bau et al., 2020) shows that inserting new key-value tuples is identical to solving a constrained least-square problem. For brevity, we stack these vectors into matrices. We have $K_0 \triangleq [k_1 \mid k_2 \mid \cdots \mid k_{n-u}]$, $K_1 \triangleq [k_{n-u+1} \mid k_{n-u+2} \mid \cdots \mid k_n]$, $V_0 \triangleq [v_1 \mid v_2 \mid \cdots \mid v_{n-u}]$, $V_1 \triangleq [v_{n-u+1} \mid v_{n-u+2} \mid \cdots \mid v_n]$, and $V_2 \triangleq [v_{n-u+1}^* \mid v_{n-u+2}^* \mid \cdots \mid v_n^*]$. Solving equation 3 can model this editing process. When inserting knowledge into linear layers like locate-then-edit style methods, such constrained least-square problems have a closed-form solution as:

$$\Delta_{\text{AdaPLE}} = R_{\text{AdaPLE}} K_1^T (C_{\text{AdaPLE}} + 2K_1 K_1^T)^{-1} \tag{4}$$

where the $R_{\text{AdaPLE}} = V_2 - 3W_0 K_1$ termed residual delta weights, and C_{AdaPLE} is covariance matrix. The detailed derivation can be found in Appendix C.1. For more precise editing, AdaPLE first computes the target knowledge representations by simultaneously optimizing the MHSA and FFN hidden states as in PMET. Subsequently, we update FFN weights in the edit target layers using target knowledge representations. The details of computing weight is in Appendix C.3.

4.2 Adaptive Fusion of Edit Weight

In the section 3, we reveal the extremely redundant properties of the edit weight of language model editing and show that such redundancy can lead to knowledge in sequential knowledge editing. For better fusion, we introduced the *Drop-and-Rescale* (DR) fusion, which is the key design and merging strategy of our proposed AdaPLE . The merging phase of AdaPLE is conceptually simple and consists of two steps: drop and re-scale. Given update weight Δ, AdaPLE first performs random dropring on update weight based on a drop rate p (setting their values to zeros) and then rescales the remaining ones by a factor $1/(1-p)$ as follows,

$$\begin{aligned} \boldsymbol{M}^l &\sim \text{Bernoulli}(p), \\ \widetilde{\boldsymbol{\Delta}} &= \left(\mathbf{1} - \boldsymbol{M}^l\right) \odot \boldsymbol{\Delta}, \\ \hat{\boldsymbol{\Delta}} &= \widetilde{\boldsymbol{\Delta}}/(1-p). \end{aligned} \tag{5}$$

Where $\boldsymbol{M}^l$ is a binary mask that shares the same shape with update weight $\boldsymbol{\Delta}$ at layer l. Finally, we will spread $\hat{\boldsymbol{\Delta}}$ to different layers in edit target to obtain the post-edit model. If we denote L as the max AIE value layer (Meng et al., 2022a), we select 5 layers $(L-4, \ldots, L-1, L)$ as edit target R. To edit multiple layers, we need to spread the residual delta weights R_{AdaPLE} to all target layers. The edit target we employed is the same as MEMIT.

Theoretical Insight. Inspired by recent work (Yu et al., 2023), we provide some theoretical intuition for our approach. We would show that even after dropping most update weight, AdaPLE can well preserve the edit performance. Let $W_0/\Delta \in \mathbb{R}^{m \times n}$ be the pre-edit model /delta parameters (or edit weight). W_1 is the post-edit weight of some layer. The input is a vector $\boldsymbol{x} \in \mathbb{R}^n$. Expectation of the i-th $(1 \le i \le m)$ dimension of the original hidden states $\boldsymbol{h} \in \mathbb{R}^m$ is computed by

$$\mathbb{E}[W_1] = \mathbb{E}[(W_0 + \Delta)] = W_0 + \Delta$$

Assuming that AdaPLE randomly drops update parameters with a ratio p, and rescales others by a factor γ. With AdaPLE , the edit weight is $\hat{\Delta} = W_1 - W_0 \in \mathbb{R}^{m \times n}$. Expectation of the i-th dimension of hidden states becomes

$$\begin{aligned} \mathbb{E}[\hat{W}_1] &= \mathbb{E}[\left(W_0 + \hat{\Delta}\right)] \\ &= W_0 + (1-p) \cdot \gamma \cdot \hat{\Delta} + p \cdot \gamma \cdot 0 \\ &= W_0 + (1-p) \cdot \gamma \cdot \hat{\Delta} \end{aligned}$$

By setting $\gamma = 1/(1-p)$, we have $\mathbb{E}[W_1] = \mathbb{E}[\hat{W}_1]$, concluding that AdaPLE can approximate the original update weight.

5 Experiment

In this section, we study several axes of the existing knowledge editing methods, including a) overall performance, and b) performance of undergoing multiple simultaneous edits. More experimental results are in Appendix D.

Method	Score↑	Efficacy↑	Generalization↑	Specificity↑
Llama2-7B	24.8	17.4 (0.5)	19.2 (0.4)	86.1 (0.7)
FT	68.8	99.1 (0.1)	78.2 (0.3)	48.3 (0.6)
MEND	23.8	16.3 (0.6)	18.8 (0.7)	**84.6** (0.5)
ROME	51.8	52.3 (0.9)	51.4 (0.8)	51.7 (0.6)
MEMIT	86.5	97.5 (0.2)	89.3 (0.5)	75.6 (0.5)
PMET	86.8	98.4 (0.1)	92.8 (0.5)	73.4 (0.4)
AdaPLE	**87.4**	**98.5** (0.4)	**93.6** (0.2)	74.2 (0.3)

Table 1: **Editing Llama2-7B on the COUNTERFACT dataset**. Within parentheses is 95% confidence interval. 95% confidence intervals are in parentheses.

Method	Score ↑	Efficacy ↑	Generalization↑	Specificity↑
Llama2-7B	27.5	27.4 (0.6)	26.8 (0.5)	28.2 (0.5)
FT	42.7	68.5 (0.6)	65.9 (0.6)	24.7 (0.5)
MEND	20.7	20.4 (0.1)	18.7 (0.3)	23.6 (0.5)
ROME	5.9	23.5 (0.7)	21.3 (0.7)	2.4 (0.1)
MEMIT	51.7	95.7 (0.3)	88.9 (0.5)	27.5 (0.5)
PMET	54.4	96.7 (0.5)	89.5 (0.4)	**29.7** (0.4)
AdaPLE	**54.6**	**97.3**(0.3)	**90.6** (0.2)	**29.7** (0.2)

Table 2: **Quantitative analysis of editing LLaMA2-7B on ZsRE.** The evaluation metric is the same as MEMIT. 95% confidence intervals are in parentheses.

5.1 Experimental Setup

Models. Our experiments are conducted with 3 different language models: GPT-J 6B (Wang and Komatsuzaki, 2021), GPT-NeoX 20B (Black et al., 2022), and Llama-2 7B (Touvron et al., 2023).

Datasets The vanilla knowledge editing is evaluated on two datasets: Zero-Shot Relation Extraction (zsRE) (Levy et al., 2017) and COUNTERFACT (Meng et al., 2022a). The details of these two datasets can be found in Appendix B.

Baselines Our baselines include full fine-tuning (FT), the meta learning-based MEND (Mitchell et al., 2022), and the locate-then-edit style methods ROME (Meng et al., 2022a), PMET (Li et al., 2023b) and MEMIT (Meng et al., 2022b).

Implementation details We conduct our experiment with Pytorch, the model weights are downloaded from Huggingface transformers library [*]. All the experiments are running on two NVIDIA RTX 8000 GPUs with 48GB RAM. We adopt the codebase from MEMIT[†] and PMET [‡] for all of the experiments in this work.

5.2 Main Results

Editing Knowledge on COUNTERFACT. The score in Table 1 is the harmonic mean of efficacy, generalization, and specificity. As a result, the unedited LLMs performed poorly in terms of efficacy and generalization but exhibited good performance in terms of specificity Except for being slightly inferior to MEND in terms of specificity, AdaPLE outperforms all baselines in all other metrics. Table 1 presents the results of all methods on 10K counterfactual edits. In the trade-off between editing reliability and specificity, both PMET, AdaPLE , and MEMIT tend to prioritize reliability, while MEND leans towards specificity.

[*]https://github.com/huggingface/transformers
[†]https://github.com/kmeng01/memit
[‡]https://github.com/xpq-tech/PMET

Editing Knowledge on ZSRE Dataset. The results of editing 10K knowledge instances on the zsRE dataset are presented in Table 2. The results demonstrate that AdaPLE outperforms existing methods in two of three metrics: efficacy, and generalization. It is worth noting that AdaPLE outperforms all baselines in all of the metrics.

5.3 Multiple Edit Performance

Our evaluation focuses on whether methods exhibit robust performance in sequential editing, with results presented in Table 3. We conduct this experiment with LLaMA-7B on COUNTERFACT dataset. As demonstrated, our proposed AdaPLE is robust to multiple edits even scaling to $10k$ on different evaluation criteria.

5.4 Ablation Studies

The AdaPLE method encompasses several key elements: (i) optimizing MHSA and FFN state sim ; (ii) spreading of update weight to target layers with square root; (iii) employing a new optimization object and (iv) drop-and-rescale (DR) fusion strategy. Now we conduct ablation studies on COUNTERFACT dataset to explore knowledge attenuation of the edited language model with LLaMA2-7B. As is presented in Table 3, results show that the drop-and-rescale fusion strategy (DR in Table) is effective in mitigating KA and simultaneously improves the language model's ability to recall knowledge after editing. Employing a new optimization objective also contributes to a reduction in KA to some extent. What's more, concurrently optimizing the states of MHSA and FFN rather than the layer state during computing knowledge representations has also an obvious effect on diminishing KA.

6 Discussion

In this section, we provide more discussion for further understanding of KA and proposed method.

# Edits	Methods	Score ↑	Efficacy ↑	Generalization ↑	Specificity↑
	Llama2-7B	24.8	17.4	19.2	86.1
1K	AdaPLE	91.6	97.8	96.4	82.3
	w/o δ_i^a	90.9 (↓0.7)	97.6 (↓0.2)	96.0 (↓0.4)	94.4 (↓2.9)
	w/o New optim	90.3(↓1.3)	96.8 (↓1.0)	95.7 (↓0.7)	80.3 (↓2.0)
	w/o DR	(89.4↓2.2)	96.6 (↓1.2)	94.7 (↓1.7)	81.1 (↓1.2)
10K	AdaPLE	87.4	98.5	93.6	74.2
	w/o δ_i^a	86.8 (↓0.6)	98.3 (↓0.2)	92.6 (↓1.0)	72.5 (↓3.9)
	w/o New optim	86.8 (↓0.6)	97.6 (↓0.9)	91.4 (↓2.2)	74.8 (↓1.6)
	w/o DR	84.8 (↓4.1)	96.3 (↓2.2)	89.2 (↓4.4)	72.6 (↓2.8)

Table 3: **The results of the sequential and ablation experiments.** Our proposed AdaPLE can scale to $10k$ edits in sequential editing like PMET. The w/o δ_i^a represents only optimizing the hidden state δ_i^m of FFN (We optimize both the hidden state of FFN and MHSA here). w/o New Optim represents using old optimization object. w/o DR represents using original fusion policy other than DR.

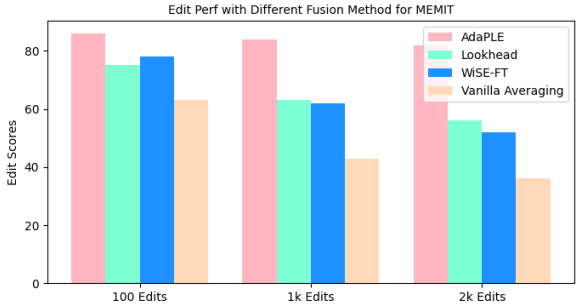


Figure 5: **Edit performance with different weight fusion strategy.** The results indicate these baselines all suffer from knowledge attenuation to some extent and our proposed DR gets the best performance.

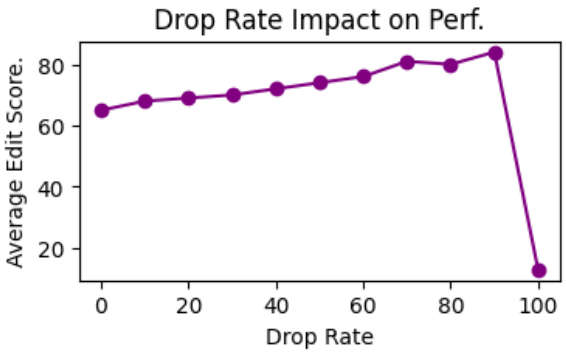


Figure 6: **Drop rate impact on edit performance.** We conduct $1k$ edits with different drop rate. The results show that setting p between 0.7 and 0.9 effectively alleviates knowledge attenuation.

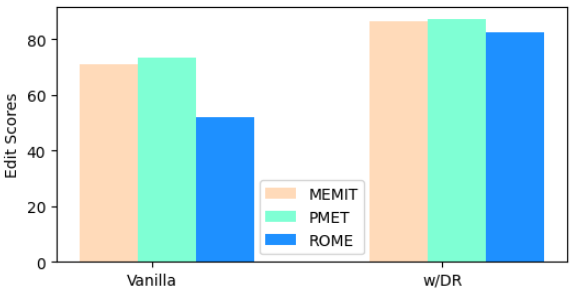


Figure 7: **Improving existing edit methods with drop-and-rescale strategy.** Empirical study indicates that employing the DR method for weight fusion can indeed mitigate the knowledge attenuation issue.

6.1 Different Fusing Algorithm.

In the previous section, we discussed the key design features of our proposed AdaPLE method in calculating update weights and its implications for mitigating KA. In this section, we aim to explore whether existing model fusion algorithms can alleviate the phenomenon of KA and, further, if they can outperform our proposed method. We primarily compared the following methods: vanilla averaging, Lookahead(Zhang et al., 2019), and WiSE-FT(Wortsman et al., 2022). Through a series of experiments, we found that these methods are all, to some extent, affected by the phenomenon of knowledge attenuation, as is shown in Figure 5.

6.2 How Does DR Rate Affect Performance ?

The section above has empirically demonstrated the effectiveness of our method. Another critical hyper-parameter that requires investigation in our proposed method is the drop rate p. We adjusted p from 0 to 1.0, conducting tests with $1k$ edits. The results in Figure 6 reveal that setting p between 0.7 and 0.9 effectively alleviates knowledge attenuation. This outcome is surprising and underscores the significant redundancy of the update weights. The interference among these redundant update weights is a major contributor to severe KA.

6.3 Can DR Strategy Help Existing Methods?

Given the success of our proposed AdaPLE , we sought to explore whether our weight fusion method DR could serve as a plug-and-play strategy to help current model editing techniques like ROME, and PMET in mitigating knowledge decay during sequential editing. To this end, we replaced the weight fusion components of ROME, PMET, and MEMIT with our DR method. The experimental outcomes in Figure 7 indicate that employing the DR method for weight fusion can indeed mitigate the KA issues observed in current model editing techniques to a certain extent.

7 Conclusion

In this paper, we systemically investigate knowledge attenuation of sequential edited language models. We analyze the potential causes of KA and attribute them to the primary defects of current editing methods, including (1) redundant parameters interference and (2) update weight disentanglement. To this end, we propose AdaPLE . Empirical studies indicate that it can effectively mitigate KA of sequentially edited language models while performing better on editing efficiency compared to existing model editing methods. It must be noted that our method can only mitigate KA to a certain extent, and we look forward to more work that can address this issue.

Limitation

In this paper, we systematically investigate the knowledge attenuation of the edited language model. Our study highlights the ubiquity of KA in sequential edited language models and proposes effective strategies for mitigating this issue. Given the prevalence of potential errors or biases in existing LLMs, our approach offers a scalable approach to alleviate this issue. Consequently, this aids in reducing and mitigating the generation of harmful or biased content. On the other hand, our method could also be utilized to inject harmful information into open-source Large Language Model weights, potentially leading to significant societal impacts.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance. *arXiv preprint*.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a deep generative model. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 351–369. Springer.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of the ACL Workshop on Challenges & Perspectives in Creating Large Language Models*.

Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Zelin Dai, Feiyu Xiong, Wei Guo, and Huajun Chen. 2023. Editing language model-based knowledge graph embeddings. *arXiv preprint arXiv:2301.10405*.

Nico Daheim, Thomas Möllenhoff, Edoardo Maria Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. 2023. Model merging by uncertainty-based gradient matching. *arXiv preprint arXiv:2310.12808*.

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.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2023. Pokemqa: Programmable knowledge editing for multi-hop question answering. *arXiv preprint arXiv:2312.15194*.

Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8214–8232.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. 2018. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

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.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *CoNLL 2017*, page 333.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix Yu, and Sanjiv Kumar. 2022. Large language models with controllable working memory. *arXiv preprint arXiv:2211.05110*.

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

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023b. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2023c. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*.

Jun-Yu Ma, Jia-Chen Gu, Zhen-Hua Ling, Quan Liu, and Cong Liu. 2023. Untying the reversal curse via bidirectional language model editing. *arXiv preprint arXiv:2310.10322*.

Michael S Matena and Colin A Raffel. 2022. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716.

Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. 2023. A survey on knowledge editing of neural networks. *arXiv preprint arXiv:2310.19704*.

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

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

Yasumasa Onoe, Michael JQ Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. *arXiv preprint arXiv:2305.01651*.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2023. Task arithmetic in the tangent space: Improved editing of pre-trained models. *arXiv preprint arXiv:2305.12827*.

Vikas Raunak and Arul Menezes. 2022. Rank-one editing of encoder-decoder models. *arXiv preprint arXiv:2211.13317*.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32.

Nianwen Si, Hao Zhang, Heyu Chang, Wenlin Zhang, Dan Qu, and Weiqiang Zhang. 2023. Knowledge unlearning for llms: Tasks, methods, and challenges. *arXiv preprint arXiv:2311.15766*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2019. Editable neural networks. In *International Conference on Learning Representations*.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. 2023. Parameter efficient multi-task model fusion with partial linearization. *arXiv preprint arXiv:2310.04742*.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. 2018. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *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.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, et al. 2023. Knowledge editing for large language models: A survey. *arXiv preprint arXiv:2310.16218*.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. 2022. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971.

John M Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity analysis of contextual word representation models. *arXiv preprint arXiv:2005.01172*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023a. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023b. Ties-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

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. *arXiv preprint arXiv:2305.13172*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*, pages 3987–3995. PMLR.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. 2019. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024. A comprehensive study of knowledge editing for large language models.

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? *arXiv preprint arXiv:2305.12740*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Appendix

A Related Work

A.1 Model Editing

Model editing has increasingly captured attention. Recent advancements suggest modifying MLP weights directly, drawing inspiration from the linear associative memory capacities of FFNs in transformers (Geva et al., 2020) and their successful implementation in convolutional models (Bau et al., 2020). Specifically, work Meng et al. (2022a) edits single facts by applying a ROME to the parameters of an MLP layer in the LLM, showing enhanced performance compared to earlier methods. Initial research in model editing focused on updating individual neurons either by constrained layer fine-tuning (Sinitsin et al., 2019; Zhu et al., 2020) or by employing hypernetwork-based methods such as (De Cao et al., 2021). Additionally, some studies have investigated the use of external memory for storing updates (Li et al., 2022).

This work builds upon Meng et al. (2022b) and Li et al. (2023b), which expanded this methodology to encompass thousands of edits by adjusting the weights across various MLP layers. Gupta et al. (2023) showed that modifying shallow layers could effectively work with MEMIT. Additional studies, such as Raunak and Menezes (2022), have explored editing within encoder-decoder models, while Cheng et al. (2023) have aimed to modify knowledge graphs.

Efforts to evaluate model editing methods have included assessing through a relational perspective, empirical studies on multi-hop QA performance like (Zhong et al., 2023) and (Gu et al., 2023), investigations into the propagation of edited entities (Onoe et al., 2023), and modifications using in-context learning (Zheng et al., 2023). For an extensive overview of model editing, references such as Yao et al. (2023), Wang et al. (2023), Mazzia et al. (2023), and Zhang et al. (2024) are recommended, alongside a survey by Si et al. (2023) focusing on unlearning within model editing.

A.2 Drawbacks of Model Editing

Despite the introduction of model editing, few studies have addressed its drawbacks. Li et al. (2023c) identified two main issues: knowledge conflict and knowledge distortion. Gupta et al. (2023) discovered that common sense knowledge localization in FFN layers of LLMs involves crucial roles for the subject, object, and relation in memory recall, albeit with a focus on classification rather than generation tasks. Ma et al. (2023) aimed at mitigating the reversal curse in model editing for LLMs.

A.3 Model Merging

In our study, we delve into the integration of diverse models or modules, a process known as model fusion, where models fine-tuned on distinct datasets are amalgamated into a unified model adept at managing multiple tasks (Li et al., 2023a; Ilharco et al., 2022; Yadav et al., 2023a). This burgeoning field is attracting increased interest. The concept of a task vector, introduced in (Ilharco et al., 2022), allows a model to learn varied tasks simultaneously without catastrophic forgetting by manipulating task vectors through arithmetic operations. Fisher-Merging (Matena and Raffel, 2022) evolves the process to weighted merging, where the weights are determined by the Fisher information matrix, rather than a simple uniform approach. Another method, RegMean (Jin et al., 2022), proposes merging linear layers by solving a local linear regression problem using inner product matrices derived from training data. AdaMerging (Yang et al., 2023) goes a step further by learning merging coefficients directly from unlabeled test data. Meanwhile, UncertaintyMerging (Daheim et al., 2023) employs a novel strategy that leverages uncertainty by utilizing multiple Fisher matrices, which are collected during both the pre-training and training stages.

A.4 Catastrophic Forgetting

A related issue to the knowledge attenuation phenomenon addressed in this paper is the well-documented problem of catastrophic forgetting (CF) (Wu et al., 2020), (Goodfellow et al., 2013). Catastrophic forgetting refers to the prevalent issue where machine learning models, trained on evolving data distributions, exhibit degraded performance on earlier data instances. Addressing catastrophic forgetting has been a significant area of research focus; however, many strategies prove effective only in particular contexts, with advancements being impeded by a limited grasp of CF's underlying characteristics. Some studies aim to deepen our understanding of CF, as seen in (Goodfellow et al., 2013; Toneva et al., 2018), while

others concentrate on counteracting CF, such as (Kemker et al., 2018). Notable methodologies include structural regularization, as exemplified by (Kirkpatrick et al., 2017) and (Zenke et al., 2017), and the functional regularization approach utilizing a replay buffer (Rolnick et al., 2019).

B Dataset

B.1 ZsRE Dataset

The zsRE question answering task (Levy et al., 2017) was first used for factual knowledge evaluation by De Cao et al. (2021), later being extended and adopted by (Mitchell et al., 2022). In our study, we use the same train/test splits as (Mitchell et al., 2022); note that non-hypernetwork methods do not require training, so the corresponding dataset split is discarded in those cases. Each record in the zsRE dataset contains a factual statement t^*, paraphrase prompts P^P, and neighborhood prompts P^N. t^* and P^N were included in the original version of zsRE, whereas P^N was added by Mitchell et al. (2022) via sampling of a random dataset element.

B.2 Counterfact Dataset

COUNTERFACT is designed to enable distinction between superficial changes in model word choices from specific and generalized changes in underlying factual knowledge. Each record in COUNTERFACT dataset is derived from a corresponding entry in (Elazar et al., 2021) containing a knowledge tuple $t^c = (s, r, o^c)$ and hand-curated prompt templates $\mathcal{T}(r)$, where all subjects, relations, and objects exist as entities in WikiData. Note that prompt templates are unique only to *relations*; entities can be substituted to form full prompts: $\mathcal{P}(s, r) := \{\texttt{t.format(s)} \mid \texttt{t} \in \mathcal{T}(r)\}$, where `.format()` is string substitution. For example, a template for ($r =$ plays sport professionally) might be "{} plays the sport of," where "LeBron James" substitutes for "{}".

C Supplementary to Method

C.1 Derivation of Close-Form Solution Optimization Object for Editing

Here we present the detailed derivation of Equation 4. This derivation is included for clarity and completeness and is a review of the classical solution of least-squares with equality constraints as applied to our setting.

If we stack keys and memories as matrices $K = [k_1 \mid k_2 \mid \cdots \mid k_n]$ and $V = [v_1 \mid v_2 \mid \cdots \mid v_n]$, then the equation can be optimized by solving the normal equation. We denote W_{out} as W. We assume that W is the optimal least-squares solution for memorizing a mapping from a previous set of keys K to values V; this solution can be written using the normal equations as follows.

$$\text{the } W \text{ that minimizes} \quad ||WK - V||_F^2 \tag{6}$$

$$\text{solves} \quad WKK^T = VK^T \tag{7}$$

Here the Frobenius norm is used to write the total square error since the variable being optimized is a matrix W rather than a vector x as in the classical textbook presentation of least squares.

We wish to find a new matrix $\hat{W}$ that solves the same least squares problem with an additional equality constraint as written:

$$\hat{W}k_* = v_* \tag{8}$$

MEMIT (Meng et al., 2022b) optimizes an objective function to obtain target weights :

$$W_1 \triangleq \underset{W}{\text{argmin}}\Big(\underbrace{\sum_{i=1}^{n} \|Wk_i - v_i\|^2}_{\text{(a) original keys and values}} + \underbrace{\sum_{i=n+1}^{n+u} \|Wk_i - v_i\|^2}_{\text{(b) inserted keys and values}}\Big). \tag{9}$$

The (a) term in Equation 9 indicates that we want to retain n original pieces of knowledge, while the (b) term indicates that we want to add u pieces of knowledge. We can solving the above equation with

block form:

$$W_1 \begin{bmatrix} K_0 & K_1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 \end{bmatrix}^T = \begin{bmatrix} V_0 & V_1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 \end{bmatrix}^T \tag{10}$$

$$\text{which expands to:} \quad (W_0 + \Delta)(K_0 K_0^T + K_1 K_1^T) = V_0 K_0^T + V_1 K_1^T \tag{11}$$

$$W_0 K_0 K_0^T + W_0 K_1 K_1^T + \Delta K_0 K_0^T + \Delta K_1 K_1^T = V_0 K_0^T + V_1 K_1^T \tag{12}$$

$$\text{subtracting norm equation :} \quad \Delta(K_0 K_0^T + K_1 K_1^T) = V_1 K_1^T - W_0 K_1 K_1^T. \tag{13}$$

However, it is important to note that there are some issues with the optimization framework of existing methods. Actually, these methods do not directly insert u new pieces of knowledge, but rather modify u pieces of existing knowledge to incorporate new information. This process should involve two steps: forgetting outdated or incorrect knowledge and inserting new knowledge. Thus the optimization goal should be:

$$\hat{W} \triangleq \underset{W}{\text{argmin}}(\sum_{i=1}^{n-u} \|Wk_i - v_i\|^2 + \sum_{i=n-u+1}^{n} \|Wk_i - v_i^*\|^2) + \underset{W}{\text{argmax}}(\sum_{i=n-u+1}^{n} \|Wk_i - v_i\|^2) \tag{14}$$

Here the k_i and v_i ($i = 1, 2, ..., n-u$)indicate the original key and value, and v_i^* refers to target value where $i = n-u+1, n-u+2, \ldots, n$. We can solve this optimization problem analytically. It can be re-write as:

$$\hat{W} \triangleq \underset{W}{\text{argmin}}(\sum_{i=1}^{n-u} \|Wk_i - v_i\|^2 + \sum_{i=n-u+1}^{n} \|Wk_i - v_i^*\|^2 - \sum_{i=n-u+1}^{n} \|Wk_i - v_i\|^2) \tag{15}$$

then

$$\hat{W} \triangleq \underset{W}{\text{argmin}}(\sum_{i=1}^{n-u} \|Wk_i - v_i\|^2 - \sum_{i=n-u+1}^{n} \|Wk_i - v_i\|^2 + \sum_{i=n-u+1}^{n} \|Wk_i - v_i^*\|^2) \tag{16}$$

The block form for above equation 14 is

$$W_1 \begin{bmatrix} K_0 & K_1 & K_1 \end{bmatrix} \begin{bmatrix} K_0 & K_1 & K_1 \end{bmatrix}^T = \begin{bmatrix} V_0 & -V_1 & V_2 \end{bmatrix} \begin{bmatrix} K_0 & K_1 & K_1 \end{bmatrix}^T \tag{17}$$

$$\text{which expands to:} \quad (W_0 + \Delta)(K_0 K_0^T + 2K_1 K_1^T) = V_0 K_0^T - V_1 K_1^T + V_2 K_1^T \tag{18}$$

$$W_0 K_0 K_0^T + 2W_0 K_1 K_1^T + \Delta K_0 K_0^T + 2\Delta K_1 K_1^T = V_0 K_0^T - V_1 K_1^T + V_2 K_1^T \tag{19}$$

$$\text{subtracting norm equation :} \quad \Delta(K_0 K_0^T + 2K_1 K_1^T) = V_2 K_1^T - 2V_1 K_1^T - W_0 K_1 K_1^T. \tag{20}$$

$$\text{Thus:} (K_0 K_0^T + 2K_1 K_1^T) = (V_2 - 2V_1 - W_0 K_1) K_1^T. \tag{21}$$

$$\text{We have :} W_0 K_1 = V_1 \tag{22}$$

$$\Delta_{\text{AdaPLE}} = (V_2 - 3W_0 K_1) K_1^T (K_0 K_0^T + 2K_1 K_1^T) \tag{23}$$

$$\text{Finally :} \ \Delta_{\text{AdaPLE}} = R_{\text{AdaPLE}} \, K_1^T (C_{\text{AdaPLE}} + 2K_1 K_1^T)^{-1} \tag{24}$$

Here V_0 denotes the unchanged value, V_1 is the value to be forgotten, and V_2 is the new value to be remembered. Here, R_{AdaPLE} is $V_2 - 3W_0 K_1$. C_{AdaPLE} is the estimate of co-variance matrix.

C.2 Algorithm

The algorithm of our proposed method AdaPLE is in Algo 1.

C.3 Computing Update Weight

Upon deriving an analytical solution for the optimization objective 3 as Equation 4, it is necessary to compute K_1, V_2, and residual C_{AdaPLE} separately.

Computing knowledge representation.

Unlike ROME and MEMIT, which add optimizable parameters δ_i to the layer hidden states h_i^L at the L-th layer and obtain the optimized layer hidden states $v_i = h_i^L + \delta_i$ through gradient descent, we follow PMET adds optimizable parameters δ_i^a and δ_i^m to the MHSA and FFN hidden states a_i^L and m_i^L of the

Algorithm 1: The AdaPLE Algorithm

Data: Edit dataset $\mathcal{E} = \{(s_i, r_i, o_i)\}$, language model M, layers to edit $\mathcal{R}$, covariances C^l
Result: Edited language model containing edits from $\mathcal{E}$
for $s_i, r_i, o_i \in \mathcal{E}$ **do**
  optimize $\delta_i^a \leftarrow argmin_{\delta_i} \frac{1}{P}\sum_{j=1}^{P} -\log \mathbb{P}_{M(h_i^L += \delta_i)}\left[o_i \mid x_j \oplus p(s_i, r_i)\right]$
  optimize $\delta_i^m \leftarrow argmin_{\delta_i} \frac{1}{P}\sum_{j=1}^{P} -\log \mathbb{P}_{M(h_i^L += \delta_i)}\left[o_i \mid x_j \oplus p(s_i, r_i)\right]$
  $z_i \leftarrow m_i^L + \delta_i^m$
$h_i^{s-1} \leftarrow 0$
for $l \in \mathcal{R}$ **do**
  $h_i^l \leftarrow h_i^{l-1} + 3m_i^l$
  for $s_i, r_i, o_i \in \mathcal{E}$ **do**
    $k_i^l = \frac{1}{P}\sum_{j=1}^{P} \text{prev}(W_{\text{out}}, \text{pref}_j \oplus p(s_i, o_i))$
    $r_i^l \leftarrow \frac{z_i - h_i^l}{\sqrt{L-l+1}}$
  $K_1^l \leftarrow [k_i^{l_1}, ..., k_i^L]$
  $R_{\text{AdaPLE}}^l \leftarrow [r_i^{l_1}, ..., r_i^L]$
  $\Delta_{\text{AdaPLE}}^l \leftarrow R^l {K_1^l}^T (C^l + 2K^l {K^l}^T)^{-1}$
  Random Drop and Re-scale Δ_{AdaPLE}^l , get final edit weight $\hat{\Delta}_{\text{AdaPLE}}^l$
  $W_1^l \leftarrow W_0^l + \hat{\Delta}_{\text{AdaPLE}}^l$

components at the L-th layer, respectively. Then, AdaPLE only retains the optimized FFN hidden states to update MLP weights W_{out}^L, denoted as $v_i^m = m_i^L + \delta_i^m = \underset{v_i^m}{\text{argmin}}\, \mathcal{L}(v_i^m)$. $\mathcal{L}(v_i^m)$ is defined as follows:

$$\begin{aligned} &\mathcal{L}(v_i^m) \\ &= D_{\text{KL}}\left(\mathbb{P}_{\mathcal{M}_\theta\left(a_i^L += \delta_i^a, m_i^L += \delta_i^m\right)}\left[\boldsymbol{y} \mid p'\right] \| \mathbb{P}_{\mathcal{M}_\theta}\left[\boldsymbol{y} \mid p'\right]\right) + \\ &\frac{1}{P}\sum_{j=1}^{P} -\log \mathbb{P}_{\mathcal{M}_\theta\left(a_i^L += \delta_i^a, m_i^L += \delta_i^m\right)}\left[\boldsymbol{y}_i^{S_t} \mid \text{pref}_j \oplus p(\boldsymbol{x_i})\right], \end{aligned} \tag{25}$$

where $\mathcal{M}_\theta\left(a_i^L += \delta_i^a, m_i^L += \delta_i^m\right)$ represents the optimizable parameters δ_i^a and δ_i^m are added to the hidden states of MHSA and FFN at the L-th layer of the model $\mathcal{M}_\theta$, respectively. $\text{pref}_j \oplus p(\boldsymbol{x_i})$ and p' are, as in ROME (Meng et al., 2022a), prefixes used to enhance the generalization of the target knowledge and the prompt template used for calculating the KL divergence: '$\{S\}$ is a'. After calculating the values of all the target knowledge that needs to be changed, they can be stacked into a matrix V_2.

Computing K_1 with diverse prefix. Once we have V_2, the next step is to obtain keys K_1. Keys are related to specific weights to be edited and represent the hidden states before entering those specific weights. The keys k_i^l at the l-th layer are defined as follows:

$$k_i^l = \frac{1}{P}\sum_{j=1}^{P} \text{prev}(W_{\text{out}}, \text{pref}_j \oplus p(s_i, o_i)) \tag{26}$$

where $\text{prev}(W_{\text{out}}, \text{pref}_j \oplus p(s_i, o_i))$ represents the hidden state of the input $\text{pref}_j \oplus p(s_i, o_i)$ before flowing through the weight W_{out}. This can be achieved by "hooking" the model to be edited (Meng et al., 2022a). If one wants to edit W_{out} , then $\text{prev}(W_{\text{out}}, p(s_i, o_i)) = \sigma\left(W_{in}^l \gamma\left(h_j^{l-1}\left(p(s_i, o_i)\right)\right)\right)$. We can then get K_1 by stacking all of k_i. It must be noted that the $p(s_i, o_i)$ here is the edit context.

Computing covariance matrix C_{AdaPLE}. $C_{\text{AdaPLE}} \triangleq K_0 K_0^T = \lambda \cdot \mathbb{E}_k\left[kk^T\right]$ is an estimate of previously memorized keys obtained through sampling from WIKI text dataset. Here, λ is a hyper-parameter that balances the modification and preservation.

D Supplement to Experiment

We conduct extra experiments with GPT-J and GPT-Neo-X on the zsRE and COUNTERFACT datasets.

D.1 More experiments on COUNTERFACT dataset

Method	Score	Efficacy	Generalization	Specificity	Fluency	Consistency
GPT-NeoX	23.5	16.6 (0.5)	18.1 (0.6)	81.4 (1.1)	620.4 (0.3)	29.0 (0.4)
FT	68.1	97.4 (0.1)	77.6 (0.7)	47.8 (0.6)	295.7 (1.3)	17.5 (0.3)
MEND	23.0	15.7 (0.7)	18.5 (0.7)	**78.6** (0.5)	**618.4** (0.3)	31.1 (0.2)
ROME	52.6	53.2 (0.6)	52.3 (0.8)	52.4(0.8)	588.8 (0.6)	4.9 (0.0)
MEMIT	81.8	97.0 (0.6)	81.8 (1.2)	70.7 (1.5)	603.5 (0.3)	36.2 (0.2)
PMET	**84.0**	**97.5** (0.1)	88.9 (0.5)	70.4 (0.5)	601.6 (0.2)	39.6 (0.2)
AdaPLE	**84.4**	**97.5** (0.3)	**89.8** (0.4)	70.6 (0.3)	604.2 (0.3)	**40.3** (0.5)
GPT-J	22.4	15.2 (0.7)	17.7 (0.6)	83.5 (0.5)	622.4 (0.3)	29.4 (0.2)
FT	67.6	99.4 (0.1)	77.0 (0.7)	46.9 (0.6)	293.9 (2.4)	15.9 (0.3)
MEND	23.1	13.5 (0.7)	15.1 (0.7)	**83.0** (0.2)	618.4 (0.3)	31.1 (0.2)
ROME	50.3	50.2 (1.0)	50.4 (0.8)	50.2 (0.6)	589.6 (0.5)	3.3 (0.0)
MEMIT	85.8	98.9 (0.2)	88.6 (0.5)	73.7 (0.5)	619.9 (0.3)	40.1 (0.2)
PMET	86.2	99.5 (0.1)	92.8 (0.4)	71.4 (0.5)	**620.0** (0.3)	40.6 (0.2)
AdaPLE	**87.1**	**99.7** (0.3)	**93.6** (0.1)	72.9 (0.4)	619.0 (0.2)	**40.9** (0.1)

Table 4: Empirical results on COUNTERFACT dataset with GPT-NeoX, and GPT-J. 95% confidence intervals are in parentheses.

D.2 More experiment on zsRE dataset

Methods	Efficacy	Generalization	Specificity
GPT-J	26.4 (±0.6)	25.8 (±0.5)	27.0 (±0.5)
FT	69.6 (±0.6)	64.8 (±0.2)	24.1 (±0.6)
MEND	19.4 (±0.4)	18.6 (±0.3)	22.4 (±0.5)
ROME	21.0 (±0.7)	19.6 (±0.7)	0.9 (±0.4)
MEMIT	96.7 (±0.3)	89.7 (±0.5)	26.6 (±0.5)
PMET	96.9 (±0.3)	90.6 (±0.2)	**26.7** (±0.2)
AdaPLE	**97.0**(±0.4)	**90.7** (±0.3)	25.3 (±0.6)
GPT-NeoX	28.6 (±0.4)	27.1 (±0.6)	27.6 (±0.2)
FT	71.6 (±0.3)	65.8 (±0.6)	26.1 (±0.5)
MEND	20.7 (±0.3)	20.6 (±0.5)	23.4 (±0.8)
ROME	23.4 (±0.8)	22.6 (±0.2)	1.6 (±0.1)
MEMIT	95.3 (±0.4)	90.7 (±0.6)	26.4 (±0.1)
PMET	95.9(±0.2)	**92.2** (±0.6)	26.7 (±0.3)
AdaPLE	**96.1**(±0.4)	**92.2** (±0.2)	**27.3** (±0.7)

Table 5: Empirical results on zsRE dataset with GPT-NeoX, and GPT-J. 95% confidence intervals are in parentheses.