AdaEdit: Advancing Continuous Knowledge Editing For Large Language Models

Qi Li [1], **Xiaowen Chu** [1] †
[1] The Hong Kong University of Science and Technology (Guangzhou)
lqinfdim@163.com

Abstract

Knowledge editing (KE) has emerged as a prominent alternative that enables efficient and precise information modification inside language models. However, a critical challenge arises in continuous language model editing — a significant performance decline both in knowledge update and retention when the number of edits increases. By dissecting the perturbation weight of language model in continuous KE, we uncover that disentangled and sparsified knowledge representation can significantly alleviate the performance decline. Building on these insights, we introduce AdaEdit, a novel knowledge editing method. Extensive empirical evaluations on multiple LLMs demonstrate that our proposed methods can enhance the performance of edited LLMs in large-size continuous editing regimes, outperforming existing ones without substantially compromising the general abilities of these models.

1 Introduction

Recently, large language model (LLM) like ChatGPT (OpenAI, 2022), Claude (Anthropic, 2024), and Llama (Touvron et al., 2023a,b) have demonstrated remarkable performance across various knowledge-intensive tasks (Petroni et al., 2019; Pan et al., 2024). However, the accumulated vast amount of knowledge in LLMs may be erroneous, harmful, or outdated (Singhal et al., 2023). Directly fine-tuning an LLM is prohibitive due to resource budget (Zhang et al., 2023), even with the parameter efficient fine-tuning. Therefore, the ability to efficiently update LLM knowledge is desirable.

To this end, *knowledge editing* (KE) (Yu et al., 2023; Tan et al., 2024; Mitchell et al., 2021) has emerged as a competitive alternative for this pursuit. The primary goal of knowledge editing is to manipulate the predictions for inputs within editing scope generally, *without* influencing unrelated knowledge.

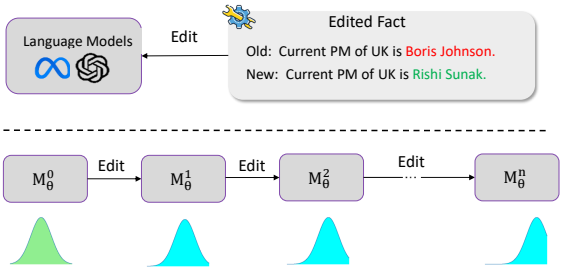


Figure 1: **Illustration of knowledge editing.** The top panel: current editing methods (like ROME (Meng et al., 2022a), MEND (Mitchell et al., 2021), IKE (Zheng et al., 2023)) can efficiently update knowledge within pre-trained language models; The bottom panel: continuous knowledge editing can lead to weight distribution shift of edited model. Here, we edit the language model continuously for lifelong knowledge refresh. The M_θ^0 is the pre-edit language model, parameterized with θ. The M_θ^n is the language model that has undergone n edits.

Current approaches (Zheng et al., 2023; Dai et al., 2022; Mitchell et al., 2022) target at enabling efficient yet precise alterations on specific knowledge triplet within LLM, e.g., modifying the outdated " The Current PM of UK is Boris Johnson " to the correct one " Current PM of UK is Rishi Sunak " persistently as is depicted in the top panel of Figure 1. Among existing approaches, the most notable are locate-then-edit style methods (Meng et al., 2022a,b), which reach the state-of-the-art performance. These methods first identify knowledge related target within LLM and then adjust the model's behavior by directly adding a *perturbation weight* Δ to the identified target.

In this paper, we focus on *continuous knowledge editing*, a task setting closely aligned with real-world applications, where editing samples arrive in sequence, and incrementally merged the perturbation of edit batch to the edit target within the language model in a continuous, step-wise manner without rolling back the update before the next edit. Recent works (Li et al., 2025; Gu et al., 2024; Yang et al., 2024a) show that current editing methods can

†Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4127–4149
July 27 - August 1, 2025

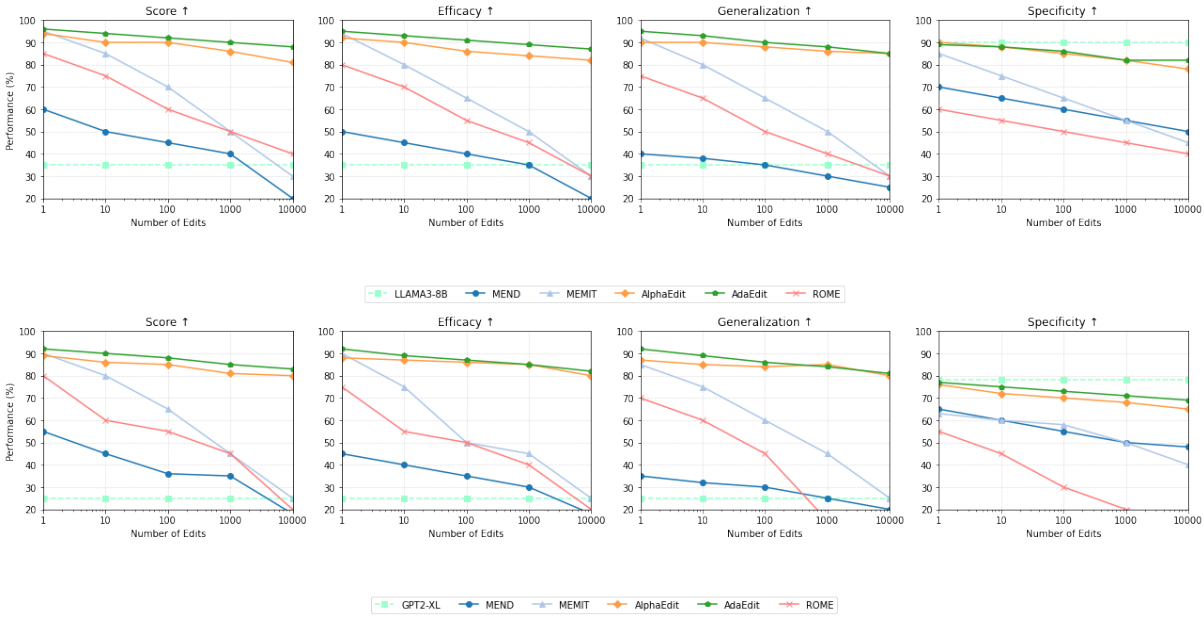


Figure 2: **Performance comparison for LLAMA-3 8B (Top) and GPT-2 XL (Bottom).** The metrics across edit score, efficacy, generalization, and specificity. Our proposed AdaEdit outperforms different baselines.

cause significant performance drop of edited language models both in edit performance and general abilities after large-scale continuous knowledge editing. However, analysis and solution of this issue remains *under-explored*. While a few existing studies (Li and Chu, 2024; Huang et al., 2024) have explored this direction, they are still far from fully tackling the problem. It is naturally to motivate the following research question:

> *How can we mitigate performance decline in the large-size continuous knowledge editing for language models ?*

In this task setting, we have identified an issue *overlooked* by existing research: as the number of edits increases, editing methods require a trade-off between maintaining the model's original performance and retaining the newly edited knowledge.

To close this gap, we conducted a comprehensive investigation on continuous KE by dissecting the weight of edited LLM. Empirically, we first delve into the perturbation weight of the edited model, finding several properties of perturbations in large-scale continuous KE: *intrinsic sparsity, low rank, and distribution shift*. To mitigate the failure of continuous KE, we have the following takeaways through investigation: (1) Entangled representation incorporates unnecessary information into perturbation weight, which leads to significant pairwise similarities between representations. (2) Redundancy in perturbation weight leads to interference, which contributes to a performance decline in continuous knowledge editing for LLM.

In this work, we introduced AdaEdit, a novel knowledge editing method to address performance decline in large scale continuous editing language models. To comprehensively evaluate its performance, we perform extensive evaluation on most widely adopted editing datasets with different language models and baselines. We then further assess edited LLMs with different editing methods on various benchmarks to evaluate their general abilities. Results demonstrate our proposed AdaEdit can nearly match state-of-the-art edit performance across different edit metrics, outperforming other baselines without largely hurting the general abilities of edited language models, as demonstrated in Figure 2. Visualization of language model hidden states edited by AdaEdit reveals that the post-edit distribution shift are obvious alleviated. The main contributions of this paper are as follows:

- We perform an investigation on the failure of continuous KE by dissecting the weight of edited model. Observations show that the perturbation is intrinsically sparse, low-rank, and shifted from the original distribution.
- Based on observations, explorations indicate disentangling representation and untying parameter interference can significantly mitigate performance drop in continuous KE.
- We introduce AdaEdit, a novel editing method that reach SOTA performance without compro-

mising the general abilities of edited LLM.

2 Preliminaries

In this section, we provide basic preliminaries of knowledge editing and related background.

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 transformer decoder layers (Vaswani et al., 2017), which consist of MHSA and FFN sub-block, followed by layer norm (Ba, 2016) and residual connection (He et al., 2016). The FFN generally comprises two linear layers, denoted as W_{in}, W_{out} respectively. In this paper, we always employ W_{out} as edit target.

2.2 Knowledge Editing

Knowledge editing (also known as model editing) aims to precisely adjust the behaviors of a language model M_θ on some facts 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 r. 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 *specificity* (Yao et al., 2023). Please refer to Section B for a comprehensive survey.

2.3 Formal Definition of Continuous Editing

Here, we provide a formal definition of continuous editing. Assume we have an unedited language model M_0, and n editing samples(x_i, y_i), where $i = [1, 2, \ldots, n]$ need to be updated inside 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 batch. Without loss of generality, we assume we can edit either one or multiple samples in each batch. For different 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 satisfy $n_t \geq 1$ for every t and satisfies $n = \sum_t n_t$ for all edit batches.

Method	Layer	Maximal	Minimal	Mean	Median	Norm
ROME	0	0.56	-0.73	9.29e-07	3.69e-06	117.05
	1	0.56	-0.73	9.40e-07	3.02e-06	117.12
	5	0.56	-0.73	7.70e-07	2.42e-06	117.56
	10	0.56	-0.73	3.12e-07	9.72e-07	118.28
	50	0.58	-0.71	7.28e-07	-1.40e-06	142.63
	100	0.93	-0.91	-6.88e-06	-1.58e-06	269.49
MEMIT	0	0.70	-0.94	2.34e-06	3.99e-06	116.57
	1	0.70	-0.94	2.34e-06	3.53e-06	116.59
	10	0.70	-0.94	1.96e-06	2.98e-06	116.91
	50	0.70	-0.90	2.02e-06	1.70e-06	118.26
	100	0.70	-0.89	1.98-06	2.03-06	121.68
	500	152.1	-113.8	0.19e-03	9.51e-06	6307.3
	1000	199.8	-146.5	0.36e-03	2.09e-05	9291.1

Table 1: **Model weight change in edited layers for different editing methods on Llama-2 7B models.** As is demonstrated in the table, the method with moderate model weight change can either preserve the general abilities of the language model or continually update knowledge throughout the editing process.

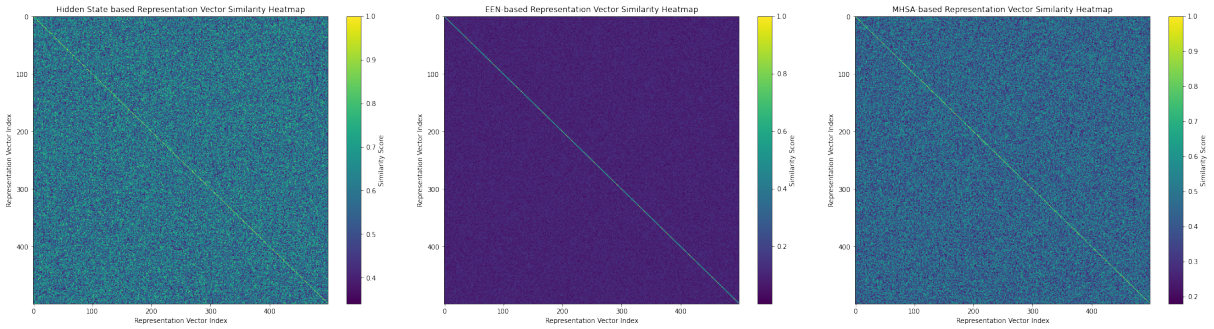
3 Demystify Failure of Continuous KE

In this section, we will investigate the failure of continuous knowledge editing of language models from the view of perturbation weight. We first provide some empirical observations on perturbation weight (Section 3.1), then by justifying the entanglement of knowledge representation, we reveal that disentangling such representation can significantly mitigate these problems (Section 3.2) Lastly, the mitigation of parameter interference is investigated in Section 3.3.

3.1 Intuitive Observations

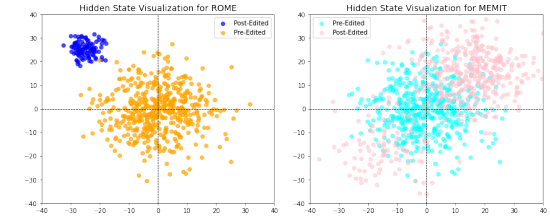
To gain a deeper understanding of the failures in large-scale continuous KE, we need to clarify what happens to the edited model after editing.

Weight shift after editing. We first performed $1,000$ continuous edits on Llama-2 7B with ROME and MEMIT, respectively, followed by an analysis of the weights in the edited regions of these models. We collect the statistics of the edited layer weight, as is shown in Table 1. In these results, a significant weight distribution shift can observed. We can observe both weight norm, average, and median shift of edited weight, which indicates the whole weight distribution of the edit target. This can be

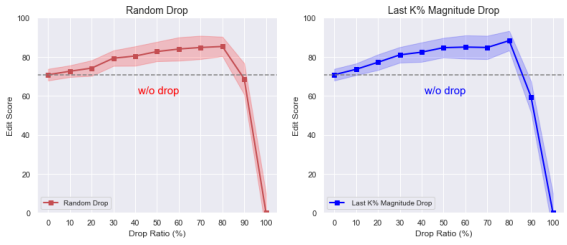


(a) Layer state-based representation (b) FFN state-based representation (c) MHSA state-based representation

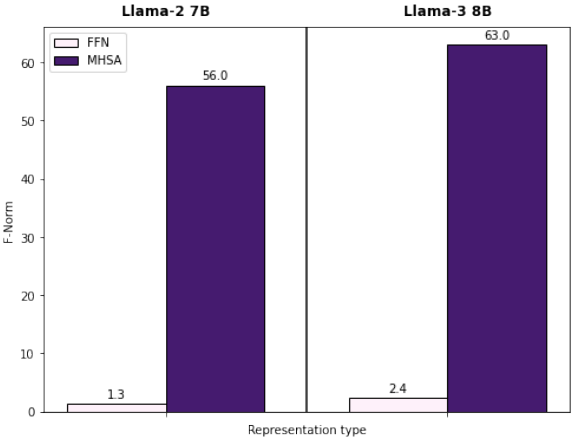
Figure 3: **Knowledge representation similarity.** We first compute 500 representation of MEMIT with layer state, FFN state, and MHSA state respectively, and then we calculate pairwise absolute cosine similarity of representation. Visualization shows that FFN-based representation has less pairwise similarity, indicating less entanglement.



(a) T-SNE visualization for ROME and MEMIT states.



(b) Dropping redundancy can mitigate parameter interference.



(c) Average F-norm of knowledge representation.

Figure 4: **Empirical study results on continuous KE.** (a) Hidden state t-SNE visualization of pre-edit and post-edit Llama-2 7B language model; (b) MEMIT editing performance with different dropping strategies with 500 samples on Llama-2 7B. Results show that sparsifying perturbation can improve performance decline in continuous KE by mitigating parameter interference; (c) Average F-norm value of 500 knowledge representation with MEMIT for different post-edit language models.

further justified by Figure 4a, which visualizes the hidden states of post-edit language models.

Observation 3.1. Significant weight shifts can be observed in edited LLM after large continuous KE.

Intrinsic sparsity of perturbation weight. Recent works (Yu et al., 2024) have shown that, for fine-tuned LLM, most model parameters can change over the adaption process but only have a small impact on performance. Does this hold for continuous language model editing? To validate this, we randomly select 500 edit samples and compute update weight Δ with MEMIT (Meng et al., 2022b). For each update weight, we apply two different dropping strategies - random drop (DR) or keeping only the largest - topk% parameters (Top-k%) and set the other parameters as zero. Results in Figure 4b show that only retaining 20% of parameters does not hurt the overall editing performance, either for DR or Top-k%. We conduct experiments on Llama-2 7B and GPT-J with 500 samples with MEMIT, and the results are consistent the same.

Observation 3.2. Perturbation weight of editing methods is intrinsically sparse where at least 80% of parameters are redundant.

Dropping redundant parameters can mitigate interference. Results in Figure 4b also indicate that incorporating the dropping strategy in continuous knowledge editing can mitigate or improve edit performance, outperforming the editing method without dropping strategy (dash line in 4b).

Observation 3.3. Dropping redundant parameters can mitigate parameter interference, enabling more robust large-scale continuous editing.

Low-rank property of perturbation weight. Suppose we use the MEMIT method to edit an LLM. This involves computing a perturbation weight, which is the same shape as W_{out}, and then it is

added to the model's W_{out} weights to alter the model's behavior. The W_{out} is a large-scale matrix. For example, in Llama-2 7B, it has a shape of 4096×11008. However, the perturbation weight is obtained by multiplying two low-rank matrices during the editing process. Suppose we edit n knowledge samples at each batch (where n is usually much smaller than the height or width of the W_{out}). In the case of editing Llama-2 7B with MEMIT, these two matrices have shapes of $4096 \times n$ and $11008 \times n$, respectively. This indicates that perturbation weight is a low-rank matrix.

Observation 3.4. Perturbation weight is highly low rank. The rank of it is bounded by the number of edit samples in its edit batch.

3.2 Mitigating Representation Entanglement

Less similarities, better performance. Recent works like (Yadav et al., 2023; Tang et al., 2024) reveal that the less similar (closer to orthogonal) between perturbation weights, the less interference between sequentially added weights, and the better scaling continuous KE performance. Current editing methods like ROME and MEMIT compute an additive representation vector that encodes the updated knowledge association with layer-hidden states, which fuse the information from the MHSA block, FFN block, and residual connection. However, not all sources of information contribute to knowledge editing. In fact, representations computed from certain information may even degrade the performance of knowledge editing. Metaphorically speaking, information from multiple sources results in the entanglement of representations.

FFN-only knowledge representations show less pairwise similarity. We argue that directly using the layer hidden states to compute knowledge representation vectors may introduce unnecessary information, which can lead to suboptimal knowledge representation. To justify this, we first perform 500 edits independently using the hidden states, FFN state, and MHSA state to compute knowledge representations in the MEMIT methods on Llama-2 7B, respectively, and then we collect these representations, keeping all other steps of the algorithm unchanged. By comparing the pairwise similarity of knowledge representation of these variants methods, as is shown in Figure 3, we find that methods using only FFN for hidden state computation have *less pairwise similarity* compared to MHSA-state or layer-state based.

FFN-only knowledge representation has less norm. Moreover, we observe that the norm of knowledge representations computed using only MHSA hidden states is significantly larger than that of representations computed using only FFN hidden states, as shown in Figure 4c, which may cause a significant norm increase in continuous KE.

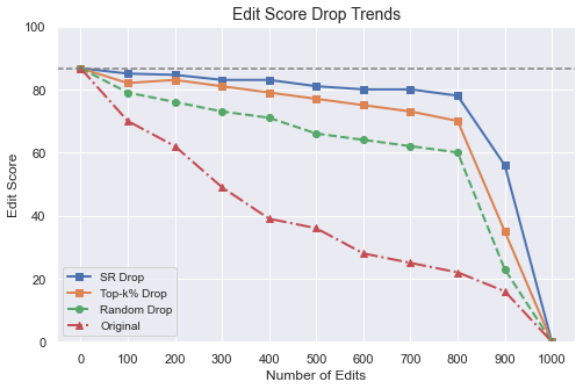


Figure 5: **Performance decline trends with different sparsify strategy.** We edit 500 samples with MEMIT on Llama-2 7B. SR exhibits better performance retention.

3.3 Untying Parameter Interference

Recent works (Li and Chu, 2024; Yadav et al., 2023) show that directly fuse perturbation weight can lead to redundant parameters interference, which cause the performance declines in continuous KE. Can sparsify update weights by retaining the top 20% highest values (Top-k%)or random drop (DR) thoroughly address this issue? Unfortunately, the answer is negative. We integrated these two strategies into the existing ROME and MEMIT methods and conducted sequential editing experiments. The results indicate that while these strategies can slightly mitigate the decline in model performance as the number of edits increases, they fail to completely prevent this degradation. We aim to sparsify the perturbation weight as much as possible while preserving its information to the greatest extent. Inspired by the observation 3.2 and 3.3, we aim to fully exploit the low-rank structure and inherent sparsity of the perturbation weight. To this end, we first decompose the perturbation weight using SVD before integrating it into the LLM. We then retain only the singular vectors corresponding to the largest singular values. These retained singular vectors are further sparsified to obtain the reconstructed weight. We denote this approach as *SR* (*Sparsify and Reconstruct*), and we integrate this strategy in continuous KE with MEMIT under 500 edits based on Llama-2 7B. It achieves much better performance retention compared to vanilla,

DR, and Top-k% strategy as shown in Figure 5.

Summary. In short, we have the following main takeaways: (1) Entangled representation incorporates unnecessary knowledge into perturbation weight, which leads to significant pairwise similarities between representations. (2) Redundancy in perturbation weight leads to interference, which contributes to a performance decline in continuous KE.

4 Proposed Method: AdaEdit

In this section, we introduce AdaEdit, a novel knowledge editing method to address continuous KE failure that builds on the insights from Section 3. Our proposed AdaEdit follows the locate-then-edit style and mainly composes two phases: (i) computing perturbation weight and (ii) dynamic merging perturbation to the W_{out}. In the computing stage, AdaEdit first computes update weight within an optimization process (In section 4.1). Once computed, the update weights are merged to edit targets dynamically (In section 4.2).

4.1 Computing Perturbation Weight

The AdaEdit follows the locate-then-edit style and regards the FFN subblock in the 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 knowledge inside LLM parameter with key-value form as knowledge representation.

To perform editing, we add perturbation weight Δ to W_0 and get $W_1 = \Delta + W_0$. For W_1, it should satisfies $W_1 k_m = v_m$ to keep the original knowledge 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 perturbation weight Δ, we view the editing process as optimization process.

AdaEdit formulates the editing process as an optimization problem like existing locate-then-edit style methods. However, as is justified in Section 3.2, existing methods incorporate unnecessary content in information flow, which leads to unprecise and redundant weight. Given this insight, we alter the optimization goal of AdaEdit to:

$$W_1 \triangleq \underset{W}{\operatorname{argmin}} [\underbrace{\sum_{i=1}^{n} \|W k_i - v_i\|^2}_{\text{(a) unedited knowledge}} + \underbrace{\sum_{j=1}^{m} \left\|W k_j - v_j^*\right\|^2}_{\text{(b) inserted new knowledge}}] \quad (1)$$

where the W is computed with FFN-only hidden states. Here, k_i and v_i $(i = 1, 2, ..., n)$ indicate the unchanged key and value of knowledge tuple $t = (s, r, o)$, and k_j, v_j, v_j^* refers to the key, old value, new value of updated value. (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 $K_0 \triangleq [\cdots \mid k_i \mid \ldots]$, $K_1 \triangleq [\cdots \mid k_j \mid \ldots]$, $V_0 \triangleq [v_1 \mid v_2 \mid \cdots \mid v_{n-u}]$, $V_0 \triangleq [\cdots \mid v_i \mid \ldots]$, and $V_1 \triangleq \left[\cdots \mid v_j^* \mid \ldots\right]$. Solving equation 1 can model this editing process. When inserting knowledge into linear layers like locate-then-edit style methods, such constrained least-square have a closed-form solution as:

$$\Delta_{\text{AdaEdit}} = R^* K_1^T (C^* + K_1 K_1^T)^{-1},$$

where the $R^* = V_1 - W_0 K_1$ termed residual delta weights, and C^* is covariance matrix. For more precise editing, AdaEdit first computes the target knowledge representations by separately optimizing the attention and FFN hidden states separately. Subsequently, we update the FFN weights in the edit target layers using target representations.

4.2 Adaptive Fusion of Perturbation Weight

In Section 3, we reveal the extremely low-rank properties and intrinsic sparsity of the perturbation weight and show that such property can lead to a performance decline in continuous knowledge editing. For better fusion, we introduced the *SRFusion* (*Sparsification-and-Reconstruction Fusion*) which utilizes singular value decomposition and hard-thresholding to achieve sparsification and norm preservation, and it is the key design and fusion strategy of our proposed AdaEdit.

The merging phase of AdaEdit is conceptually simple like AdaPLE (Li and Chu, 2024) and consists of two steps: Sparsification and reconstruction. Given a large-scale low-rank perturbation matrix $\Delta \in \mathbb{R}^{m \times n}$, where m and n are integers. We aim to reconstruct a sparse approximation of Δ, denoted as $\hat{\Delta}$, while preserving its low-rank structure and maintaining its norms. The proposed method consists of the following steps: This approach utilizes Singular Value Decomposition (SVD) to achieve sparsification and norm preservation, as below:

1. *Low-Rank Decomposition*: First, we perform SVD on perturbation weight Δ and retain the top r singular values along with their corresponding singular vectors:

$$\Delta \approx \sum_{i=1}^{r} \sigma_i u_i v_i^T. \tag{2}$$

2. *Sparsification of Singular Vectors*: Apply a hard-thresholding operation to the singular vectors u_i and v_i, keeping only the top k largest absolute values and setting the rest to zero:$u_i' = \mathcal{T}_k(u_i), v_i' = \mathcal{T}_k(v_i)$,where $\mathcal{T}_k(\cdot)$ represents the operation of selecting the top k largest while zeroing out the rest.
3. *Normalization*: Since the thresholding step alters the ℓ_2 norm of u_i' and v_i', we renormalize them to match their original norms:

$$u_i' \leftarrow \frac{\|u_i\|_2}{\|u_i'\|_2} u_i', \quad v_i' \leftarrow \frac{\|v_i\|_2}{\|v_i'\|_2} v_i'. \tag{3}$$

4. *Reconstruction of the Sparse Matrix*: Using the processed singular vectors, we reconstruct an approximate matrix:

$$\hat{\Delta} = \sum_{i=1}^{r} \sigma_i u_i' (v_i')^T. \tag{4}$$

Finally, we will spread $\hat{\mathbf{\Delta}}$ 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_{AdaEdit} to all target layers. The edit target layer we employed is the same as settings in like MEMIT or PMET.

5 Experiments

In this section, we empirically verify the effectiveness of the proposed method and provide answers to the three questions: **Q1:** How effective are the proposed methods on knowledge editing datasets with continuous KE settings? **Q2:** How helpful are AdaEdit to retain the general abilities of language models? **Q3:** What insights can hidden state visualization provide to improve continuous KE?

Experimental Setups. We begin by briefly outlining the models, baselines, datasets, benchmarks, and metrics. Please refer to Appendix E for details.

1) Language Models. We conduct experiments on various LLMs, including Llama-2 7B (Touvron et al., 2023b), Llama-3 8B (Grattafiori et al., 2024), GPT2-XL (Radford et al., 2019). We conduct experiments with these language models.

2) Editing Baselines. We employ multiple editing methods as baselines: MEND (Mitchell et al., 2021), ROME (Meng et al., 2022a) , MEMIT (Meng et al., 2022b) , AlphaEdit (Fang et al., 2025), RECT (Gu et al., 2024), PRUNE (Ma et al., 2025).
3) Editing Datasets. In this paper, we employ the most widely adopted editing datasets ZsRE (Levy et al., 2017) and Counterfact (Meng et al., 2022a) as editing datasets across all experiments.
4) Evaluation Benchmarks. To assess the general capabilities of edited LLMs, we utilize five distinct benchmarks. These include: MMLU, BBH, GSM8K, CommonsenseQA, TriviaQA.

Knowledge Editing Evaluation (Q1). To answer Q1, we perform editing evaluation on two important datasets: ZsRe and Counterfact.

1) Main Results. The quantitative results of editing evaluation with baselines and our proposed AdaEdit across different language models are provided in Table 2. We randomly select 3000 samples from both datasets, with a single sample for each edit in continuous editing setting. As is demonstrated, AdaEdit outperforms all of the baselines across different metrics on two datasets. In addition to editing capabilities, AdaEdit also exhibits substantial improvements in fluency and coherence, enhancing the quality of text generation. For more experimental results on edit evaluation, please refer to Appendix F.1.

2) Scaling Performance. In addition to directly observing the performance of the language model after a large number of consecutive knowledge edits, we also aim to understand how the model's performance changes as the number of edits increases. As is shown in Figure 2, all of the existing methods suffer from drastic performance decline in large-scale continuous knowledge editing settings for Llama-2 7B and Llama-3 8B models. When scaling edits to $10k$, our proposed AdaEdit can keep a relatively high performance on different edit metrics, outperforming other baselines.

General Abilities Evaluation (Q2). We evaluate edited language models on benchmarks to test their general abilities. To further benchmark the intrinsic knowledge of post-edited LLMs, we perform general abilities evaluation using 5 benchmarks for LLM: MMLU, BBH, GSM8K, CSQA, TrivalQA.

Method	Counterfact					ZsRE		
	Efficacy	Gen.	Spec.	Fluency	Cons.	Efficacy	Gen.	Spec.
Pre-edited	10.3	11.5	88.7	632.8	23.6	35.8	35.4	31.6
MEND	13.6	1.8	12.6	362.0	0.2	0.0	0.0	0.0
ROME	18.6	9.4	1.3	276.1	0.0	0.0	0.0	0.0
MEMIT	27.5	13.7	14.1	342.3	7.3	21.2	20.9	9.7
RECT	76.7	42.6	42.8	482.7	15.6	63.5	62.3	23.8
PRUNE	52.8	23.3	29.9	423.6	11.5	0.0	0.0	0.0
AlphaEdit	97.2	90.9	**68.5**	584.3	27.4	**92.3**	**90.7**	20.9
AdaEdit(ours)	**97.3**	**91.2**	68.2	**612.4**	**27.6**	**92.3**	90.6	**21.5**

Table 2: **Quantitative evaluation of knowledge editing on ZsRE and Counterfact dataset with Llama-3 8B.** Results demonstrate that our method outperforms other baselines. For more results, please refer to Appendix F.

Benchmark	Pre-edited	MEND	ROME	MEMIT	PRUNE	AlphaEdit	AdaEdit
MMLU	0.6930	0.6877	0.0000	0.0000	0.0000	0.6518	0.6624
BBH	0.6423	0.6325	0.0000	0.0000	0.0000	0.6224	0.6196
GSM8K	0.5740	0.5621	0.0000	0.0000	0.0000	0.5155	0.5445
CSQA	0.7534	0.7492	0.0000	0.0000	0.0000	0.7124	0.7317
TrivalQA	0.7628	0.7598	0.0000	0.0000	0.0000	0.7226	0.7158

Table 3: **Results of general abilities evaluation across 5 different LLM benchmarks after** 5000 **edits with Llama-3 8B.** Our proposed AdaEdit can maintain general abilities even after a massive number of edits.

The results of evaluation on different models with different baselines are listed in Table 3. It shows that all of the baselines undergo a significant general abilities decay after 5000 edits, while AdaEdit can maintain the general abilities of the original model performance across all benchmarks. For more experimental results on general abilities evaluation, please refer to Appendix F.2.

Hidden States Analysis (Q3). Previous empirical studies (Figure 4a) have shown that existing methods can lead to a shift in the distribution of hidden states. Hence, here we aim to empirically verify whether AdaEdit can prevent hidden state shift or not. To achieve this, we conducted the following steps: (1) We randomly select $1,000$ factual prompts and extract the hidden representations within pre-edited LLMs. (2) Subsequently, we performed $2,000$ sequential edits on the LLMs and recomputed these hidden representations. (3) Finally, we used t-SNE to visualize the hidden representation before and after editing. Specifically, the hidden representations within LLMs edited using our proposed AdaEdit (in Figure 6) remain consistent with the original distribution.

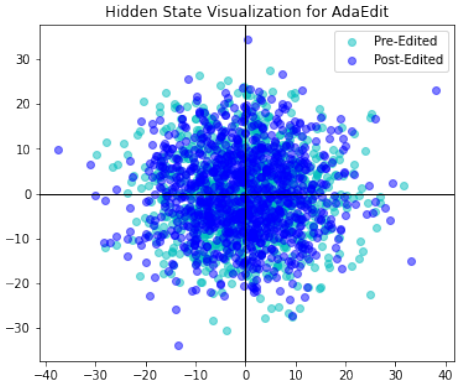


Figure 6: **Visualization for AdaEdit hidden states.** We visualize 500 pre/post-edit hidden states with t-SNE.

6 Conclusion

In this work, we introduced AdaEdit, a novel KE method to address a critical challenge in continuous KE— a significant performance drop both in knowledge update and retention when the number of edits becomes large. A deep investigation of weight perturbation shows their intrinsic sparsity, low-rank property, and entanglement, which may lead to suboptimal continuous KE performance. Through empirical studies, we reveal that (1) Entangled representation incorporates unnecessary in-

formation into perturbation weight, which leads to significant pairwise similarities between representations; (2) Redundancy in perturbation weight leads to interference, which contributes to a performance decline in continuous knowledge editing for LLM. Extensive empirical evaluations demonstrate that our proposed methods can enhance the performance of edited LLMs in large-size KE, outperforming existing ones without substantially compromising the general abilities of these models.

Limitation

In this paper, we systematically investigate the continuous KE of LLM. Our study highlights the ubiquity of both performance drops in knowledge updating and general abilities in continuously 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 LLM weights, potentially leading to significant societal impacts.

Acknowledgements

This work was partially supported by National Natural Science Foundation of China under Grant No. 62272122, the Guangzhou Municipal Joint Funding Project with Universities and Enterprises under Grant No. 2024A03J0616, Guangzhou Municipality Big Data Intelligence Key Lab (2023A03J0012), and Hong Kong CRF grants under Grant No. C7004-22G and C6015-23G.

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*.

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Jimmy Lei Ba. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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.

Su Lin Blodgett, Gilsinia Lopez, Alexandra Olteanu, Robert Sim, and Hanna Wallach. 2021. Stereotyping norwegian salmon: An inventory of pitfalls in fairness benchmark datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1004–1015.

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.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2023. Evaluating the ripple effects of knowledge editing in language models. *arXiv preprint arXiv:2307.12976*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

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.

Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. Bold: Dataset and metrics for

measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 862–872.

Shizhe Diao, Pengcheng Wang, Yong Lin, Xiang Liu, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *Preprint*, arXiv:2302.12246.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*.

Yao Fu, Litu Ou, Mingyu Chen, Yuhao Wan, Hao Peng, and Tushar Khot. 2023. Chain-of-thought hub: A continuous effort to measure large language models' reasoning performance. *arXiv preprint arXiv:2305.17306*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arxiv*.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, and Archie Sravankumar. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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*.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. *arXiv preprint arXiv:2401.07453*.

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.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2024. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. 2024. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2022. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada. Association for Computational Linguistics.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

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*.

Qi Li and Xiaowen Chu. 2024. Can we continually edit language models? on the knowledge attenuation in sequential model editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5438–5455, Bangkok, Thailand. Association for Computational Linguistics.

Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. 2025. Should we really edit language models? on the evaluation of edited language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023a. 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. 2023b. Unveiling the pitfalls of knowledge editing for large language models. *arXiv preprint arXiv:2310.02129*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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*.

Chenyang Lyu, Minghao Wu, and Alham Fikri Aji. 2024. Beyond probabilities: Unveiling the misalignment in evaluating large language models. *Preprint*, arXiv:2402.13887.

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*.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. 2025. Perturbation-restrained sequential model editing. In *The Thirteenth International Conference on Learning Representations*.

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, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

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.

OpenAI. 2022. Introducing chatgpt.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

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.

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

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. 2024. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

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

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*.

Anke Tang, Li Shen, Yong Luo, Yibing Zhan, Han Hu, Bo Du, Yixin Chen, and Dacheng Tao. 2024. Parameter-efficient multi-task model fusion with partial linearization. In *The Twelfth International Conference on Learning Representations*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

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

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

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Jianchen Wang, Zhouhong Gu, Zhuozhi Xiong, Hongwei Feng, and Yanghua Xiao. 2024. The missing piece in model editing: A deep dive into the hidden damage brought by model editing. *arXiv preprint arXiv:2403.07825*.

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*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

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

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*, 36:7093–7115.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. 2024a. The butterfly effect of model editing: Few edits can trigger large language models collapse. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5419–5437, Bangkok, Thailand. Association for Computational Linguistics.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Du Su, Dawei Yin, and Huawei Shen. 2024b. The fall of ROME: Understanding the collapse of LLMs in model editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4079–4087, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2023. Melo: Enhancing model editing with neuron-indexed dynamic lora. *arXiv preprint arXiv:2312.11795*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Longteng Zhang, Xiang Liu, Zeyu Li, Xinglin Pan, Peijie Dong, Ruibo Fan, Rui Guo, Xin Wang, Qiong Luo, Shaohuai Shi, et al. 2023. Dissecting the runtime performance of the training, fine-tuning, and inference of large language models. *arXiv preprint arXiv:2311.03687*.

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. *Preprint*, arXiv:2401.01286.

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, 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 and Supplementary Material

A Implementation and Reproduction List

In this section, we would like to provide details for reproducing our experimental results.

A.1 Code Base

Here, we list the code base used in our paper.

- For all of the models, we use huggingface transformers as default https://github.com/huggingface/transformers.
- For editing baselines, we use EasyEdit framework https://github.com/zjunlp/EasyEdit.
- For model evaluation, we use the code and data from chain-of-thought hub https://github.com/FranxYao/chain-of-thought-hub and Language Model Evaluation Harness https://github.com/EleutherAI/lm-evaluation-harness.
- For accelerating model evaluation on GPU, we use the vLLM framework https://github.com/vllm-project/vllm.

A.2 Models

Here, we list all of the model checkpoints used in our experiments.

- Llama-2 7B https://hf-mirror.com/meta-llama/Llama-2-7b-hf
- GPT-2 XL https://hf-mirror.com/openai-community/gpt2-xl
- Llama-3 8B https://hf-mirror.com/meta-llama/Llama-3.1-8B

A.3 Datasets

Here we list resources for all of the benchmark dataset for evaluating general abilities.

- MMLU https://github.com/FranxYao/chain-of-thought-hub/tree/main/MMLU
- BBH https://github.com/FranxYao/chain-of-thought-hub/tree/main/BBH
- CSQA https://huggingface.co/datasets/tau/commonsense_qa
- GSM8K https://github.com/FranxYao/chain-of-thought-hub/tree/main/gsm8k

A.4 Hyperparameter Settings

In editing experiments, our hyperparameters are in accord with the EasyEdit framework at https://github.com/zjunlp/EasyEdit/tree/main/hparams. For model evaluation, please refer to the Appendix.

A.5 Running Devices

All of our experiments are running on a server with 8 RTX A6000 GPUs with 48GB VRAM.

B Related Work

In this section, we briefly review the related works in knowledge editing and language model evaluation.

B.1 Knowledge Editing

Model editing aims to precisely modify knowledge within a language model with fine grain. Existing editing methods can be divided into 3 different categories: Retrieval-based, Extra-parameters-based, and Locate-then-edit-based methods. Early works on editing focused on updating individual neurons using constrained fine-tuning (Sinitsin et al., 2019; Zhu et al., 2020), or hypernetworks (De Cao et al., 2021). A related line of work has focused on storing updates in an external memory (Li et al., 2022). Inspired by the linear associative memory property of FFN in transformers (Geva et al., 2020) and success with the approach in convolutional models (Bau et al., 2020), recent Locate-then-edit style works have proposed to edit MLP weights directly (Meng et al., 2022a,b; Li et al., 2023a). In the encyclopedic factual domain, work (Meng et al., 2022a) proposed to edit single facts by fitting a Rank One Model Edit (ROME) to the parameters of an MLP layer and showed that it outperformed prior methods. While (Gupta et al., 2023) concentrates on editing commonsense knowledge. Work (Raunak and Menezes, 2022) focuses on editing the encoder-decoder model. Work (Wang et al., 2023; Zhang et al., 2024; Yao et al., 2023) make a comprehensive survey in model editing. Locate-then-edit style editing methods like ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b) treat FFN of LLM layers as associative memory (Geva et al., 2023; Hase et al., 2023). They view editing as adding update weight which contains knowledge representation to specific layers of the language model. They located knowledge memory at FFN of edit layers via casual tracing as editing targets. Extra-parameters-based methods like GRACE (Hartvigsen et al., 2024) add extra modules for updated knowledge, other than modifying original parameters.

B.2 Pitfalls of Knowledge Editing

Since the concept of model editing was introduced, there exist few works to discuss the drawbacks of existing methods. In (Li et al., 2023b), two kinds of pitfalls are discovered, named knowledge conflict and knowledge distortion. MEMIT-CSK (Gupta et al., 2023) finds common sense knowledge can also be localized in FFN of LLM layers and both subject, object, and relation play important roles in recalling memory, but they only focus on classification tasks. (Ma et al., 2023) focus on mitigating the reversal curse in model editing for LLM. Some recent works (Zhong et al., 2023; Gu et al., 2023) try to evaluate the multi-hop reasoning of updated knowledge. Work (Wang et al., 2024; Cohen et al., 2023) tries to assess the underly impact caused by editing. What's more, some works (Chen et al., 2020; Gekhman et al., 2024; Luo et al., 2023; Wu et al., 2024) discuss the knowledge forgetting during fine-tuning. Recent works (Cohen et al., 2023; Zhong et al., 2023; Li et al., 2023b) have disclosed the inevitable pitfalls of existing editing methods from different perspectives such as knowledge distortion (Li et al., 2023b), knowledge attenuation (Li and Chu, 2024), and catastrophic forgetting (Gupta et al., 2024) or performance degradation (Li et al., 2025) in language model general abilities.

B.3 Evaluation of Language Models

Evaluating the effectiveness of LLMs (Touvron et al., 2023a,b; Jiang et al., 2023; Bai et al., 2023; Almazrouei et al., 2023; Team et al., 2024) involves a diverse array of tests, where models are assessed across various tasks, showcasing their capabilities. Among the benchmarks employed, Bigbench (Srivastava et al., 2022; Suzgun et al., 2022), MMLU (Hendrycks et al., 2020), and HELM (Liang et al., 2022) are notable. These benchmarks utilize a range of automatic evaluation metrics, such as BLEURT (Sellam et al., 2020) and others requiring meticulous annotation to ensure data quality for downstream applications (Papineni et al., 2002; Lin, 2004; Wang et al., 2018). Typically, these evaluations emphasize accuracy across multiple choices, which serves as the primary metric (Clark et al., 2018; Brown et al., 2020). Foundation models, often tested across broad linguistic tasks—both generative and multiple-choice—are analyzed to ensure rigorous assessment standards (Schaeffer et al., 2024; Hendrycks et al., 2020; Wei et al., 2022a; Srivastava et al., 2022). These methods, however, may focus on the accuracy within confined options and might not effectively capture the nuances of more open-ended, practical applications where models generate

free-form text. Moreover, assessing the safety of LLMs is essential. Safety evaluations concentrate on three key areas: truthfulness (Blodgett et al., 2021), toxicity (Hartvigsen et al., 2022), and bias (Dhamala et al., 2021).

C Supplementary to Preliminaries

C.1 Locate-then-Edit knowledge Editing

Locate-then-Edit style knowledge editing methods view down-sample component W_{out}^l of FFN in LLM layers as associative memory, denoted as W. The transformer architecture is proposed 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 :

$$W_1 \triangleq \underset{W}{\text{argmin}} (\underbrace{\sum_{i=1}^{n} \|Wk_i - v_i\|^2}_{\text{(a) original keys and values}} + \underbrace{\sum_{i=n+1}^{n+u} \|Wk_i - v_i\|^2}_{\text{(b) inserted keys and values}}), \tag{5}$$

where the W is computed with layer hidden states. The (a) term in Equation 5 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 *perturbation weight* Δ. The close-form solution of edit target Δ is :

$$\Delta = RK_1^T (C_0 + K_1 K_1^T)^{-1}, \tag{6}$$

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. Please refer to (Meng et al., 2022b) for detailed derivation of Equation 6.

C.2 General Abilities of Language Models

In recent years, the field of LLM has experienced rapid growth, leading to the development of numerous models by various research institutions. These models differ significantly in terms of parameter size, architecture, corpora, and training methodologies. Consequently, it has become critically important to evaluate the capabilities of these models objectively, and comprehensively. Typically, this is achieved by evaluating the models on widely adopted benchmarks like MMLU (Hendrycks et al., 2020), and BigBench (Suzgun et al., 2022) to compare their performance with their counterparts. Currently, the evaluation of the general capabilities of LLMs in both academia and industry focuses on several key areas: world knowledge, common sense reasoning, coding, reading comprehension, mathematical skills, and performance on mainstream benchmark datasets (Bai et al., 2023; Touvron et al., 2023a,b; Jiang et al., 2023; Team et al., 2024).

D Detailed Exploration on Perturbation Weight

D.1 Discussion on Weight Analysis

In the continuous knowledge editing of language models, changes in the weight distribution of edit target have been widely observed in a series of studies like (Huang et al., 2024; Ma et al., 2025; Li et al., 2025; Yang et al., 2024b), even for a single edit (Yang et al., 2024a). Some studies attribute the performance degradation of models after large-scale continuous editing to these changes.

More specifically, (Ma et al., 2025) attributes these changes to variations in the condition number of the weight matrix caused by weight perturbation, while (Huang et al., 2024) associates these changes with alterations in the L1 norm of the weight matrix. Although these observations are supported by empirical

results, in this subsection, we argue that these observations and conclusions have limitations and do not fully capture the essence of the problem. As is shown in Table 4 and 5, we can observe both weight norm, average, and median shift of edited weight, which indicates the whole weight distribution of the edit target. This can be also justified by Figure 4a, which visualizes the hidden states of post-edit language models.

Method	Layer	Original	Current Norm	Abs Change	Perc Change
ROME	5	117.053	269.497	152.444	130.235%
MEMIT	4	116.950	121.432	4.482	3.832%
	5	117.053	120.957	3.904	3.335%
	6	116.384	120.322	3.938	3.384%
	7	116.579	121.681	5.102	4.376%
	8	117.807	127.963	10.156	8.621%
PMET	4	116.950	117.098	0.148	0.127%
	5	117.053	117.129	0.076	0.065%
	6	116.384	116.446	0.062	0.053%
	7	116.579	116.667	0.088	0.075%
	8	117.807	117.579	0.228	0.194%
MEND	29	126.963	126.949	0.014	0.011%
	30	126.209	126.198	0.011	0.009%
	31	126.048	126.033	0.015	0.012%

Table 4: **Layer weight norm change at 100 edits in Llama2-7B model.** When edits increase, the norm of layer weight change edited by some methods (e.g. ROME) can grow very rapidly.

D.2 Discussion on Weight Entanglement

Findings in weight entanglement also suggests that, in knowledge editing, FFN primarily functions as a knowledge memory, storing the vast majority of knowledge associations, while MHSA retains only a small amount of knowledge. Its main role is to aggregate the knowledge from previous layers while extracting relevant knowledge from the current layer's FFN, which align with observations in lines of current research (Geva et al., 2020; Li et al., 2023a).

E Experiments Setting Details

In this section, we would like to provide more details of the main experiments.

E.1 Model Editing Evaluation Criteria

Current evaluation primarily focuses on whether the model can effectively recall edited knowledge, whether the edits cause side effects, and whether these edits can generalize within the editing domain (paraphrase)(Yao et al., 2023). The post-edit model M_θ should satisfy the following three properties: **reliability**, **generalization**, and **specificity**.

Reliability Previous works like ROME(Meng et al., 2022a), Transformer-Patcher(Huang et al., 2022) define a reliable edit when the post-edit model M_θ^e gives the target answer for the knowledge tuple $t = (s, r, o)$ to be edited. If we demnte prompt $p(s, r)$ that embed s and r as x, o as y. The reliability is measured as the average accuracy on the edit case:

$$\mathbb{E}_{x'_\mathrm{e}, y'_\mathrm{e} \sim \{(x_\mathrm{e}, y_\mathrm{e})\}} \mathbb{1} \left\{ \mathrm{argmax}_y \, f_{\theta_e} \left(y \mid x'_\mathrm{e} \right) = y'_\mathrm{e} \right\} \tag{7}$$

Method	Layer	Original	Current Norm	Abs Change	Perc Change
MEMIT	4	116.950	362.899	245.949	210.303%
	5	117.053	4168.976	4051.923	3461.614%
	6	116.384	7870.129	7753.745	6662.209%
	7	116.579	9291.127	9174.548	7869.812%
	8	117.807	12823.619	12705.812	10785.278%
PMET	4	116.950	118.703	1.753	1.499%
	5	117.053	118.095	1.042	0.890%
	6	116.384	117.267	0.883	0.759%
	7	116.579	117.749	1.170	1.004%
	8	117.807	116.612	1.195	1.014%
MEND	29	126.963	126.901	0.062	0.049%
	30	126.209	126.199	0.010	0.008%
	31	126.048	126.034	0.014	0.011%

Table 5: **Layer weight norm change at 1000 edits in Llama2-7B model.** As is demonstrated in the table, when edit numbers are large, different editing methods have distinct norm changes.

Generalization The post-edit model f_{θ_e} should also edit the equivalent neighbour $N\left(x_{\mathrm{e}}, y_{\mathrm{e}}\right)$ (e.g. rephrased sentences). It is evaluated by the average accuracy of the model f_{θ_e} on examples drawn uniformly from the equivalence neighborhood:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim N\left(x_{\mathrm{e}}, y_{\mathrm{e}}\right)} \mathbb{1}\left\{\operatorname{argmax}_y f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = y'_{\mathrm{e}}\right\} \tag{8}$$

Specificity also noted as **Locality** in some work. Editing should be implemented locally, which means the post-edit model f_{θ_e} should not change the output of the irrelevant examples in the out-of-scope $O(x_e, y_e)$. Hence, the locality is evaluated by the rate at which the post-edit model f_{θ_e}'s predictions are unchanged as the pre-edit f_θ model:

$$\mathbb{E}_{x'_{\mathrm{e}}, y'_{\mathrm{e}} \sim O\left(x_{\mathrm{e}}, y_{\mathrm{e}}\right)} \mathbb{1}\left\{f_{\theta_e}\left(y \mid x'_{\mathrm{e}}\right) = f_\theta\left(y \mid x'_{\mathrm{e}}\right)\right\} \tag{9}$$

Fluency also noted as generation entropy. Measure for excessive repetition in model outputs. It uses the entropy of n-gram distributions:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \tag{10}$$

where $g_n(\cdot)$ is the n-gram frequency distribution.

Consistency also noted as reference score. The consistency of the model's outputs is evaluated by giving the model f_θ a subject s and computing the cosine similarity between the TF-IDF vectors of the model-generated text and a reference Wikipedia text about o.

For comprehensive evaluation criteria of the existing model edit methods, please refer to the survey paper in (Yao et al., 2023; Zhang et al., 2024; Mazzia et al., 2023).

E.2 Editing Dataset Details

ZsRE (Levy et al., 2017) is a question-answering (QA) dataset that uses questions generated through back-translation as equivalent neighbors. Following previous work, natural questions are used as out-of-scope data to evaluate locality. Each sample in ZsRE includes a subject string and answers as the editing

targets to assess editing success, along with the rephrased question for generalization evaluation and the locality question for evaluating specificity.

Counterfact (Meng et al., 2022a) is a more challenging dataset that contrasts counterfactuals with factual statements, initially scoring lower for Counterfact. It constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. The Counterfact dataset has similar metrics to ZsRE for evaluating efficacy, generalization, and specificity. Additionally, Counterfact includes multiple generation prompts with the same meaning as the original prompt to test the quality of the generated text, specifically focusing on fluency and consistency.

E.3 Benchmark Details

For our evaluation benchmark, we employ three datasets into two distinct evaluation methodologies: generation-based and sequence-based (Lyu et al., 2024). The generation-based method utilizes vLLM (Kwon et al., 2023) inference framework and follows the procedures outlined in Chain-of-Thought Hub (Fu et al., 2023) and Active-Prompt (Diao et al., 2023). For sequence-based evaluations, we use the Language Model Evaluation Harness framework (Gao et al., 2023). Detailed statistics for each benchmark dataset are provided in Table 6.

DATASET	TASK TYPE	# FEW-SHOT	# TEST	METRIC	EVALUATION METHOD
MMLU (Hendrycks et al., 2020)	World Knowledge	5	14,079	Accuracy	Generation-Based
BBH (Suzgun et al., 2022)	World Knowledge	3	6,511	Accuracy	Generation-Based
GSM8K (Cobbe et al., 2021)	Arithmetic	8	1,319	Exact match	Generation-Based
CSQA* (Talmor et al., 2018)	Commonsense	7	1,221	Accuracy	Generation-Based
TriviaQA (Joshi et al., 2017)	Reading Comprehension	0	17,900	Exact match	Generation-Based
TruthfulQA (Blodgett et al., 2021)	Truthful	0	817	Accuracy	Sequence-Based
ToxiGen (Hartvigsen et al., 2022)	Hate Speech	0	940	Accuracy	Sequence-Based

Table 6: The statistics of the datasets used in this paper. # EX. are the number of few-shot chain-of-thought exemplars used to prompt each task in evaluation. # TEST denote the number of training data and test data, respectively. *: CSQA do not have publicly available test set labels, so we simply follow the setting by (Wei et al., 2022b; Diao et al., 2023) to evaluate the performance of the development set.

E.4 Baseline Details

Here we introduce the all of baseline methods employed in this study. **For the hyperparameter settings of the baseline methods, we used the code provided in the Easyedit framework for reproduction.** It is important to note that, since the code for PRUNE is not publicly available, we implemented the method based on the description in the original paper. Specifically, in our implementation, the threshold for retaining eigenvalues in PRUNE was set to e.

- **MEND** is a method for efficiently editing large pre-trained models using a single input-output pair. MEND utilizes small auxiliary networks to make fast, localized changes to the model without full retraining. By applying a low-rank decomposition to the gradient from standard fine-tuning, MEND enables efficient and tractable parameter adjustments. This approach allows for post-hoc edits in large models while avoiding the overfitting common in traditional fine-tuning methods.

- **ROME** is a method for updating specific factual associations in LLMs. By identifying key neuron activations in middle-layer feed-forward modules that influence factual predictions, ROME modifies feed-forward weights to edit these associations directly. ROME demonstrates that mid-layer feed-forward modules play a crucial role in storing and recalling factual knowledge, making direct model manipulation a viable editing technique.

- **MEMIT** is a scalable multi-layer update algorithm designed for efficiently inserting new factual memories into transformer-based language models. Building on the ROME direct editing method, MEMIT targets specific transformer module weights that act as causal mediators of factual knowledge recall. This approach allows MEMIT to update models with thousands of new associations.

- **PRUNE** is a model editing framework designed to preserve the general abilities of LLMs during sequential editing. PRUNE addresses the issue of deteriorating model performance as the number of edits increases by applying condition number restraints to the edited matrix, limiting perturbations to the model's stored knowledge. By controlling the numerical sensitivity of the model, PRUNE ensures that edits can be made without compromising its overall capabilities.
- **RECT** is a method designed to mitigate the unintended side effects of model editing on the general abilities of LLMs. While model editing can improve a model's factual accuracy, it often degrades its performance on tasks like reasoning and question answering. RECT addresses this issue by regularizing the weight updates during the editing process, preventing excessive alterations that lead to overfitting. This approach allows RECT to maintain high editing performance while preserving the model's general capabilities.
- **AlphaEdit** is a knowledge editing method that aims to update specific knowledge while preserving existing knowledge. Unlike traditional methods that struggle with balancing knowledge retention and modification, AlphaEdit introduces a null-space-constrained approach. It projects the perturbations onto the null space of preserved knowledge before applying updates, ensuring that previously stored knowledge remains intact. This technique prevents model forgetting and overfitting in sequential editing scenarios. AlphaEdit significantly improves editing performance (by 36.4% on average) across various LLMs, such as LLaMA3 and GPT-2 XL. It seamlessly integrates into existing editing frameworks like MEMIT and ROME, making it a simple yet highly effective solution for updating LLMs without disrupting their overall knowledge structure.

F Detailed Experimental Results

F.1 Edit Evaluation

Here, we present the full results of the main results for editing evaluation.

Method	Counterfact					ZsRE		
	Efficacy	Gen.	Specificity	Fluency	Consistency	Efficacy	Gen.	Specificity
GPT2-XL								
Pre-edited	21.6	23.6	77.9	620.3	31.5	21.6	30.7	23.6
MEND	7.2	0.0	3.4	246.3	0.2	0.0	0.0	0.0
ROME	8.4	3.2	0.0	289.4	0.6	0.0	0.0	0.0
MEMIT	15.6	10.6	5.3	315.8	3.5	7.6	6.9	3.8
RECT	57.8	28.4	26.8	439.5	8.1	36.2	28.4	12.8
PRUNE	34.3	25.6	14.3	361.7	7.7	0.0	0.0	0.0
AlphaEdit	96.6	91.3	65.7	596.2	33.2	90.6	88.3	18.5
AdaEdit	96.5	91.0	66.5	602.6	32.9	90.7	88.2	18.7
Llama-3 8B								
Pre-edited	10.3	11.5	88.7	632.8	23.6	35.8	35.4	31.6
MEND	13.6	1.8	12.6	362.0	0.2	0.0	0.0	0.0
ROME	18.6	9.4	1.3	276.1	0.0	0.0	0.0	0.0
MEMIT	27.5	13.7	14.1	342.3	7.3	21.2	20.9	9.7
RECT	76.7	42.6	42.8	482.7	15.6	63.5	62.3	23.8
PRUNE	52.8	23.3	29.9	423.6	11.5	0.0	0.0	0.0
AlphaEdit	96.4	91.6	67.5	609.3	36.4	92.3	90.4	20.9
AdaEdit	97.3	91.2	68.2	612.4	35.6	92.3	90.6	21.5

Table 7: **Quantitative evaluation of knowledge editing on ZsRE and Counterfact dataset with Llama-3 8B and GPT-2 XL.** Results demonstrate that our method outperforms other baselines.

F.2 General Abilities Evaluation

Benchmark	Pre-edited	MEND	ROME	MEMIT	PRUNE	AlphaEdit	AdaEdit
GPT-2 XL							
MMLU	0.2098	0.2098	0.0000	0.0000	0.0000	0.1124	0.1152
BBH	0.0382	0.0382	0.0000	0.0000	0.0000	0.1061	0.0156
GSM8K	0.0144	0.0144	0.0000	0.0000	0.0000	0.0089	0.0097
CSQA	0.1941	0.1941	0.0000	0.0000	0.0000	0.1451	0.1433
TrivalQA	0.0536	0.0423	0.0000	0.0000	0.0000	0.0049	0.0061
Llama-2 7B							
MMLU	0.4587	0.4562	0.0000	0.0000	0.0000	0.4057	0.4239
BBH	0.4000	0.3956	0.0000	0.0000	0.0000	0.3842	0.3841
GSM8K	0.1440	0.1329	0.0000	0.0000	0.0000	0.1323	0.1342
CSQA	0.5921	0.5844	0.0000	0.0000	0.0000	0.5689	0.5518
TrivalQA	0.5252	0.5173	0.0000	0.0000	0.0000	0.4835	0.4926
Llama-3 8B							
MMLU	0.6930	0.6877	0.0000	0.0000	0.0000	0.6518	0.6624
BBH	0.6423	0.6325	0.0000	0.0000	0.0000	0.5726	0.6196
GSM8K	0.5740	0.5621	0.0000	0.0000	0.0000	0.4855	0.5445
CSQA	0.7534	0.7492	0.0000	0.0000	0.0000	0.7224	0.7317
TrivalQA	0.7628	0.7598	0.0000	0.0000	0.0000	0.7026	0.7158

Table 8: **Results of general abilities evaluation across 5 different LLM benchmarks after** 5000 **edits.** Our proposed AdaEdit can maintain general abilities even after massive number of edits.