Forget for Get: A Lightweight Two-phase Gradient Method for Knowledge Editing in Large Language Models

Yanhong Li[1, 3], **Min Yang**[2], **Xiping Hu**[1*], **Chengming Li**[1*]
[1]Shenzhen MSU-BIT University,
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences,
[3]Harbin Institute of Technology (Shenzhen)
220810209@stu.hit.edu.cn, min.yang@siat.ac.cn, {huxp, licm}@smbu.edu.cn

Abstract

Recent studies have highlighted the remarkable knowledge retention capabilities of Large Language Models (LLMs) like GPT-4, while simultaneously revealing critical limitations in maintaining knowledge currency and accuracy. Existing knowledge editing methodologies, designed to update specific factual information without compromising general model performance, often encounter two fundamental challenges: parameter conflict during knowledge overwriting and excessive computational overhead. In this paper, we introduce **ForGet (Forget for Get)**, a novel approach grounded in the principle of "forgetting before learning". By pinpointing the location within the LLM that corresponds to the target knowledge, we first erase the outdated knowledge and then insert the new knowledge at this precise spot. **ForGet** is the first work to leverage a two-phase gradient-based process for knowledge editing, offering a lightweight solution that also delivers superior results. Experimental findings show that our method achieves more effective knowledge editing at a lower cost compared to previous techniques across various base models.

1 Introduction

Large Language Models (LLMs) have revolutionized natural language processing, enabling unprecedented capabilities in language comprehension and generation (Brown et al., 2020; Raffel et al., 2020; Ouyang et al., 2022). A key factor behind these capabilities is the vast amount of knowledge embedded within these models. However, this knowledge is often static, leading to issues such as outdated information, inaccuracies, and potential privacy violations. For instance, answering 'Who is the President of the United States?' in 2024 yields 'Joe Biden,' but this response becomes incorrect in 2025 if the model is not updated. Knowledge Editing is proposed to address this problem. Knowledge Editing aims to modify the specific knowledge stored in LLM without affecting other irrelevant knowledge and maintaining a low computational cost (Yao et al., 2023).

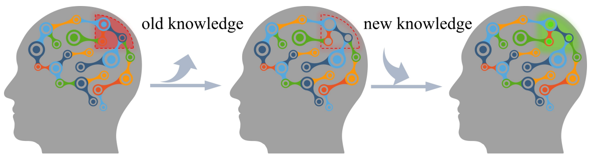


Figure 1: Clearing old knowledge before learning new knowledge helps human beings to better focus on new knowledge.

Existing knowledge editing methods can be broadly categorized into three classes (Li et al., 2024). Some of the methods utilize an additional knowledge base to store edits (Mitchell et al., 2022; Hartvigsen et al., 2024; Wang et al., 2024b), some methods use in-context learning (Zheng et al., 2023; Qi et al., 2024), others first decide the location to edit then perform editing at the specific location (Huang et al., 2023; Yu et al., 2024; Mitchell et al., 2021; Dai et al., 2021; Meng et al., 2022a,b). The existing methods have largely succeeded in editing the knowledge stored in LLMs.

These methods have achieved good editing effects, but there are still some challenges. One of the issues is that it's difficult to address the old knowledge when inserting new knowledge effectively. When editing knowledge in LLMs, conflicts between new and old knowledge may arise, which can hinder the model's ability to learn new information (Wang et al., 2024a) and weaken the effect of editing. Just like humans, it is difficult to change old knowledge when it has become deeply ingrained. Another challenge is the resource consumption of the methods. Some methods are highly effective, but they require additional time, storage, and computational resources. This also makes the model more complex and reduces the convenience of de-

[*]corresponding author

ployment of the method. Therefore, knowledge editing methods should become more lightweight while ensuring editing effectiveness.

In order to resolve knowledge conflicts, a straightforward approach is to forget the old knowledge before learning the new knowledge. For example, before going to Suzhou, one should first remove the luggage packed for Miami from the suitcase and then pack the luggage prepared for Suzhou. Inspired by the human cognitive mechanisms where forgetting old information is a prerequisite for learning new information (Anderson and Hulbert, 2021), we propose a method named **ForGet (Forget for Get)**. First of all, critical MLP modules are found out by the knowledge circuits determined by target knowledge. Knowledge editing is then performed on these critical MLP modules while the rest of the model remain unchanged. During the editing process, we first apply gradient ascent to these modules to eliminate the old knowledge, which is defined as the forgetting process. After the forgetting process, we use gradients descent to insert new knowledge into the model. To the best of our knowledge, this is the first work to leverage gradient ascent and descent for knowledge editing, offering a lightweight and efficient solution to the problem of knowledge conflicts. Our method has no additional components or training processes, and the amount of pre-computation is minimal, making it lightweight and plug-and-play. The main contributions of this work can be summarized as follows:

- We propose **ForGet**, the first lightweight knowledge editing framework to leverage a two-phase gradient-based process—gradient ascent for forgetting outdated knowledge and gradient descent for acquiring new knowledge.

- We explore the potential of using knowledge circuits to determine editing location, which effectively depict the storage and flow of knowledge within Large Language Models.

- The experimental results demonstrate that our method is able to achieve both effective editing and preservation of unrelated knowledge, while being significantly more resource-efficient compared to existing methods.

2 Related Work

The existing methods can be roughly divided into three categories: methods with additional memories, methods learning from examples and methods modifying components directly(Li et al., 2024).

2.1 Additional Memories

A straightforward strategy for knowledge editing is to store edits in external memories for future retrieval. SERAC (Mitchell et al., 2022) stores edits in a cache and uses an edit scope classifier to choose between the original model and a counterfactual model based on input and cached edits. Similarly, GRACE (Hartvigsen et al., 2024) stores edits in a codebook, searching and replacing erroneous knowledge with the most similar key in codebook when an error occurs. WISE (Wang et al., 2024b)'s dual-memory design comprises a main memory for old knowledge and a side memory for edits. RECIPE (Chen et al., 2024) adds prompt filtered from Knowledge Retrieval Repository by knowledge sentinel to inputs. And LTE (Jiang et al., 2024) utilize fine-tune to align and retrieval to inference. However, these methods require additional storage space and sometimes extra training, complicating their practical deployment.

2.2 Learning from Examples

Methods of learning from examples refers to methods utilizing In-context learning. Without changing any parameters, Zheng et al. (Zheng et al., 2023) propose editing the knowledge in the model by constructing three different demonstrations: copy, update, and retain. Such direct use of In-context learning can lead to overfitting to individual samples. Building upon this, Qi et al. (Qi et al., 2024)propose employing In-context learning aimed at a distribution rather than individual samples. The above methods often requires a significant amount of human labor to design and construct demonstrations.

2.3 Modifying Components Directly

Other methods directly modify the base model's components for better editing results, falling into two categories:

Adding Trainable Components while maintaining the original modules unchanged is one of the strategies to edit knowledge precisely. Huang et al. (Huang et al., 2023) rectify erroneous knowledge by adding neurons into the final layer, which are trained to encapsulate new knowledge. To effectively encodes edit information, Yu et al. (Yu et al., 2024) propose MELO consisting dynamic LoRA (Valipour et al., 2023) and vector database. Furthermore, MEND (Mitchell et al., 2021) employs the strategy of meta-learning, integrating an entire hypernetwork within the model. These methods edit knowledge by adding new components into original model, which augment the model's complexity.

Updating Original Components avoids increasing model complexity. Ni et al. (Ni et al., 2024) employs parametric arithmetic to facilitate the forgetting of old knowledge and learning of new knowledge. To achieve precise edits, researchers often first identify optimal locations, frequently targeting feedforward layers where knowledge is stored as key-value pairs (Geva et al., 2021). Dai et al. (Dai et al., 2021) proposed the concept of knowledge neurons and try to edit knowledge by modifying knowledge neurons. Some works apply causal mediation analysis (Pearl, 2022) to find editing location. After finding one critical MLP module, Meng et al. (Meng et al., 2022a) employ rank-one update (Bau et al., 2020) to this module. Later, Meng et al. (Meng et al., 2022b) try to use multiple MLP modules to perform mass editing. Hu et al. (Hu et al., 2024) identify a pattern mismatch issue when locating edit positions and propose using specific edits to locate specific editing locations. Fang et al. (Fang et al., 2024) propose utilizing null space to alleviate the damage to original unrelated knowledge. The computation of covariance matrices and projection matrices in these methods is time-consuming and computationally intensive.

In contrast, **ForGet** is a lightweight method without additional components and requires minimal pre-computation, thereby ensuring the model's complexity remains unchanged and low human labor. Compared to F-Learning proposed by Ni et al. (Ni et al., 2024), **ForGet** involves locating to edit precisely and utilizes different stategy for forgetting and learning process. What's more, **ForGet** not only identifies precise editing locations but also explicitly mitigates knowledge conflicts, making it a more robust and conflict-free editing process.

3 Task Definition

Our task is to edit knowledge within LLMs precisely. As equation 1 shows, given the target knowledge K and original model f with parameters θ, our goal is to design a method $F()$ that performs the necessary edits to produce an updated model f' with parameters θ'.

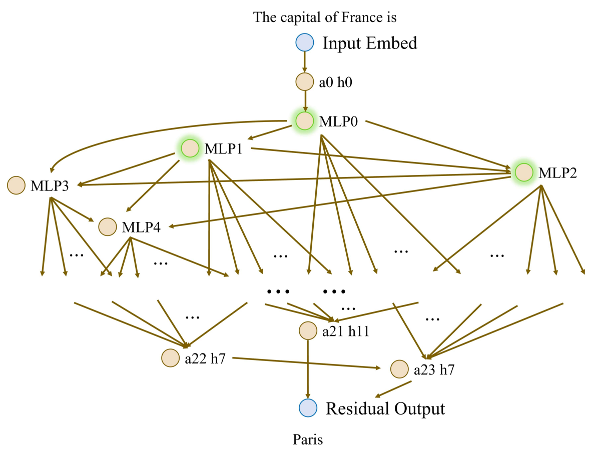


Figure 2: A simplified schematic diagram of the knowledge circuit for the knowledge "The capital of France is Paris."

$$f' = F(K, f) \tag{1}$$

Editing knowledge precisely means that only the knowledge within editing scope will be edited and others should not be affected, as equation 2 shows. The editing scope refers to a set of inputs related to the target knowledge that requires editing (Yao et al., 2023). For example, after the 2024 U.S. election, the answer to "Who is the President of the United States" should be changed from "Biden" to "Trump", but the answer to "Who is the President of Russia" should remain "Putin" both before and after editing.

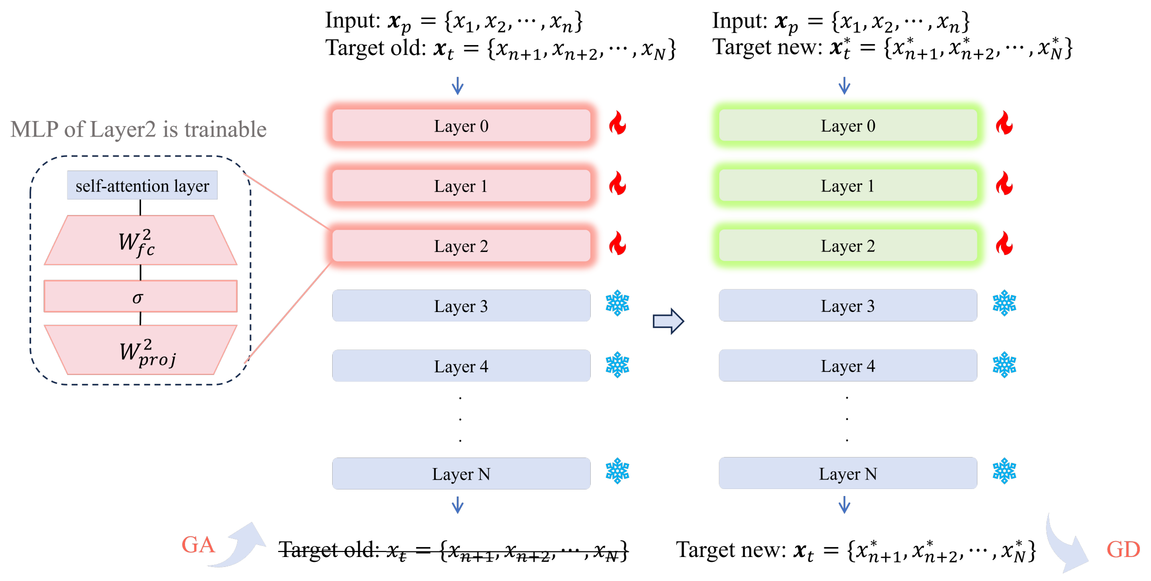
$$f'(x) = \begin{cases} y', & \text{if } x \in X^* \\ y, & \text{if } x \notin X^* \end{cases} \tag{2}$$

where X^* means editing scope which is the range of knowledge that needs to be edited. And y' represents output context related to knowledge K while y is the original output.

4 Method: ForGet

In this section, we are going to introduce our method for knowledge editing: **ForGet (Forget for Get)**. Instead of making use of additional memories or designing clever demonstrations, we adopted a direct two-phase gradient adjustment, offering a lightweight yet effective solution.

The ForGet framework consists of two main phases: **(1) determining the editing locations** and **(2) performing the editing operations**. In



Input: $\boldsymbol{x}_p = \{x_1, x_2, \cdots, x_n\}$
Target old: $\boldsymbol{x}_t = \{x_{n+1}, x_{n+2}, \cdots, x_N\}$
Input: $\boldsymbol{x}_p = \{x_1, x_2, \cdots, x_n\}$
Target new: $\boldsymbol{x}_t^* = \{x_{n+1}^*, x_{n+2}^*, \cdots, x_N^*\}$
MLP of Layer2 is trainable
self-attention layer
W_{fc}^2
σ
W_{proj}^2
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer N
GA
~~Target old: $\boldsymbol{x}_t = \{x_{n+1}, x_{n+2}, \cdots, x_N\}$~~
Target new: $\boldsymbol{x}_t = \{x_{n+1}^*, x_{n+2}^*, \cdots, x_N^*\}$
GD

Figure 3: After determining the editing location, only the modules within editing location are trainable in later process. First, gradient ascent is performed to eliminate old knowledge, followed by the opposite gradient descent to acquire new knowledge.

the first phase, we identify the components of the model that are most relevant to the target knowledge requiring editing. The second phase occurs at the editing location identified in the first phase. We begin by using gradient ascent to forget the old knowledge, followed by gradient descent to acquire the new knowledge.

4.1 Determine Editing Location

4.1.1 Knowledge Circuits

To pinpoint the optimal editing locations, we leverage **knowledge circuits**, a powerful framework for understanding the mechanisms of knowledge storage and flow within LLMs (Yao et al., 2024). A neural network model like LLM can be viewed as a connected directed acyclic graph G. Its nodes represent the components of the neural network such as neurons and attention heads and its edges represents the relations between these components such as residual connections and attention mechanisms. A knowledge circuits, defined as a subgraph of LLM's connected directed acyclic graph and represented as $C \subseteq G$, is responsible for certain knowledge. That is to say, for a particular piece of knowledge, its knowledge circuit is the part of the large language model that is most closely related to it. Therefore, identifying the knowledge circuit reveals the significant locations within the large language model where the knowledge is stored, generated and expressed. Figure 2 presents a simplified example of knowledge circuit.

4.1.2 Editing Location Discovery

To achieve precise edits and minimize impacts on unrelated knowledge, we opt to leverage Knowledge Circuits for localizing edit locations, as opposed to relying on Causal Tracing like ROME (Meng et al., 2022a). We believe that the knowledge circuit can more comprehensively and intuitively depict the storage and flow paths of knowledge within the model, thereby enabling more precise localization.

The knowledge circuit for a specific piece of knowledge comprises the nodes and edges most closely associated with it. To locate the knowledge circuit, we evaluate the importance of each edge in the computational graph using both clean inputs and corrupted inputs.

$$(z'_u - z_u)\frac{1}{m}\sum_{k=1}^{m}\frac{\partial L(z' + \frac{k}{m}(z - z'))}{\partial z_v} \qquad (3)$$

Inspired by Hanna et al., we use EAP-IG (Edge Attribution Path with Integrated Gradients) (Hanna et al., 2024) score to quantify the contribution of each edge to predicting target knowledge. First of all, sequences of token embeddings z and z' for clean input s and corrupted input s' are fed into the model, resulting in the activation z_u and z'_u for node u, respectively. For an edge (u, v), the EAP-IG score is computed by equation 3. The loss function L measures the discrepancy between activations for clean and corrupted inputs, which can take various forms such as cross-entropy or KL divergence. Additionally, the summation part in the equation is actually an approximation of an integral, accumulating gradients along the straight line path between s and s', which is designed to addresses the issue of zero gradients (Syed et al., 2024).

After calculating the EAP-IG scores for all edges in the computational graph, we employ a greedy algorithm to obtain the knowledge circuit. As pointed out by the work of Geva et al. (Geva et al., 2021), the MLP structures in the transformer architecture serve as the primary memory storage locations. To restrict the range of editing locations and enhance targeting, we select the top k MLP components with the highest degrees from the knowledge circuit as the editing location since they are the most active in predicting the target knowledge.

4.2 Performing Editing

By identifying the knowledge circuit, we are able to determine the editing location. At the editing location, we leverage a two-phase gradient-based process to perform editing as shown in Figure 3: gradient ascent for forgetting the old knowledge and gradient descent for learning new knowledge.

Forgetting old knowledge is the first step of performing editing at editing location. We apply gradients ascent to the modules at editing location to erase old knowledge.

$$\mathcal{L}_{forgetting}(f_\theta, \mathbf{x}) = -\sum_{i=n}^{N} \log(p(x_i|x_{<i}, \theta)) \quad (4)$$

Specifically, when we apply gradient ascent to the components at the editing location, it essentially amounts to directly minimizing the likelihood of the old knowledge's occurrence. This is the opposite of gradient descent, which increases the probability of the target's appearance.

$$\theta_f = GA(\theta, K_{old}) \quad (5)$$

For instance, given a sequence of tokens $\mathbf{x} = (x_1, x_2, x_3, ..., x_N)$ containing a piece of factual knowledge, our forgetting object is to maximize the loss function 4. In the loss function, $\mathbf{x}_p = \{x_i | i < n\}$ are the prompts given to the model while $\mathbf{x}_t = \{x_i | n < i < N\}$ are the target tokens of old knowledge. Therefore, the input sequence can also be expressed as $\mathbf{x} = (\mathbf{x}_p, \mathbf{x}_t)$. And $p(x_i|x_{<i}, \theta)$ denotes the conditional probability of predicting the next token to be x_i given LLM with parameters θ and sequence $\mathbf{x}_{<i}$. The parameters θ of LLM is updated as equation 5.

Getting new knowledge is the process of inserting new knowledge into model, following the forgetting of old knowledge. In contrast to forgetting old knowledge, we employ gradient descent to acquire new knowledge, which is a process that is completely opposite to forgetting.

$$\mathcal{L}_{getting}(f_{\theta_f}, \mathbf{x*}) = \sum_{i=n}^{N^*} \log(p(x_i^*|x_{<i}^*, \theta_f)) \quad (6)$$

The loss function 6 of learning process is similar to equation 4, with the key difference being the reversal of sign and different input sequence. Among these, the only difference in the input sequences $\mathbf{x} = (\mathbf{x}_p, \mathbf{x}_t)$ and $\mathbf{x*} = (\mathbf{x}_p, \mathbf{x}_t^*)$ lies in the predicted target. This construction establishes an approximate symmetry between the forgetting and acquisition processes. We maximize the loss function 6 and update the parameters that have gone through forgetting process θ_f as illustrated in equation 7.

$$\theta' = GD(\theta_f, K_{new}) \quad (7)$$

5 Experiments Setup

5.1 Datasets

In this work, we make use of ZsRE (Levy et al., 2017), COUNTERFACT (Meng et al., 2022a) for our experiments. ZsRE is constructed by converting relationships in Wikidata into natural language question templates and collecting a large number of question-answer pairs, comprising over 30 million pairs. However, COUNTERFACT is a highly challenging dataset composed of counterfactual data. Due to the counterfactual nature of the data in COUNTERFACT, it is more challenging for models to make predictions. Simultaneously, counterfactual data effectively simulates the actual scenario of editing misinformation, thereby enabling COUNTERFACT to better evaluate the editing effectiveness of models. More details about datasets and examples can be found in Appendix B.1. In addition, for comprehensive evaluation, we also evaluate our method on RIPPLEEDITS (RIPE) (Cohen et al., 2024) and present the results in Appendix C.1.

Model	Dataset	Type	Method	Eff.	Gen.	Loc.	Flu.	Score
Llama-2-7b	COUNTERFACT	MP	ForGet	**99.22**	**79.80**	**77.91**	**595.35**	**84.63**
			FT	99.75	91.69	19.12	548.64	40.97
			FT-c	99.38	55.69	48.18	593.22	61.51
			ROME	99.74	97.01	63.14	601.73	82.93
			MEMIT	98.71	98.07	63.44	598.68	83.12
			AlphaEdit	96.73	90.02	62.88	609.34	80.32
		AM	GRACE	97.35	18.69	96.65	557.64	40.47
			SERAC	99.99	76.07	98.96	549.91	90.22
			LTE	98.12	90.13	88.20	590.64	91.95
			RECIPE	98.20	94.74	92.04	586.69	94.92
	ZsRE	MP	ForGet	**76.10**	**75.44**	**52.95**	**601.24**	**66.25**
			FT	58.67	47.23	25.25	496.34	38.55
			FT-c	48.17	31.01	71.41	490.83	44.77
			ROME	99.29	41.38	26.92	620.88	42.03
			MEMIT	93.07	51.43	27.96	610.71	45.48
			AlphaEdit	90.17	89.23	30.25	608.22	54.19
		AM	GRACE	98.20	33.23	96.42	589.41	59.23
			SERAC	99.17	56.48	30.23	410.89	49.29
			LTE	98.11	73.18	66.48	583.70	77.12
			RECIPE	97.10	74.74	72.04	589.74	79.64
Qwen2-7b	COUNTERFACT	MP	ForGet	**94.48**	**79.12**	**70.98**	**602.75**	**80.40**
			FT	15.42	11.38	30.04	560.64	16.13
			FT-c	24.55	20.14	92.74	593.22	29.65
			ROME	97.44	39.51	38.12	600.73	48.53
			MEMIT	93.27	32.90	50.11	601.68	49.12
		AM	SERAC	99.01	77.07	90.96	569.91	88.05
			LTE	98.11	84.12	85.17	608.43	88.70
	ZsRE	MP	ForGet	**72.96**	**70.25**	**40.45**	**590.06**	**56.97**
			FT	71.82	75.95	9.10	287.15	21.90
			FT-c	72.08	76.53	28.32	283.20	48.19
			ROME	99.28	35.83	45.71	591.58	50.11
			MEMIT	97.25	34.31	55.25	594.74	52.14
		AM	SERAC	98.43	56.79	39.28	495.12	56.36
			LTE	95.72	70.90	74.99	580.08	79.18

Table 1: Performance comparison of different methods for COUNTERFACT and ZsRE on Llama-2-7b and Qwen2-7b models. "Eff.", "Gen." and "Loc." are the abbreviations of Efficacy, Generalization, and Locality, respectively. MP and AM indicate Modifying Parameters and Additional Memories, respectively.

5.2 Evaluation Metrics

We employed the metrics proposed by Meng et al.(Meng et al., 2022a) in our experiments. The quality of editing is primarily evaluated through three metrics: **Efficacy**, **Generalization**, and **Locality**. (1) **Efficacy** measures how well the editing method can directly modify knowledge within LLM. For example, if our editing goal is to change "The President of the United States is Joe Biden" to "The President of the United States is Donald Trump," then the edited model should output "Donald Trump" when given the input "The President of the United States is." (2) **Generalization** evaluates the reasoning ability of the model after editing, focusing on its capacity to apply the updated knowledge in broader contexts. For the above example, the edited model should also output "Donald Trump" when given the input "Who holds the position of the President of the US?" (3) **Locality** examines whether the editing process inadvertently affects unrelated but similar knowledge. For instance, given the input "The President of Russia is," the model should respond with "Putin" both before and after editing. Additionally, we take **Fluency** calculated by n-gram entropy into account, avoiding edited models to generate repetitive content. More details can be found in Appendix B.2.

5.3 Base LLMs and Baseline Methods

We use Llama-2-7b (Touvron et al., 2023) and Qwen2-7b (Yang et al., 2024) as base model for our experiments. To verify the effectiveness of **ForGet**, we conduct experiments on several classic baselines. Firstly, we compare direct fine-tuning(FT) and FT-c (Zhu et al., 2020), which utilizes L_∞ norm constraint to prevent overfitting with our method. To make it more comparable, we restricted the editing locations of the fine-tuning based methods to one MLP component. We also employ F-Learning (Ni et al., 2024), the other method that follows the principle of "forgetting before learning". As for the other methods that modify parameters, we include ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b) and AlphaEdit (Fang et al., 2024) in our experiments. We also involve methods that utilize additional memories in our experiments for further analysis, including GRACE (Hartvigsen et al., 2024), SERAC (Mitchell et al., 2022), LTE (Jiang et al., 2024) and RECIPE (Chen et al., 2024). More implementation details can be found in Appendix B.2.

6 Experiments Results

The experimental results for knowledge in COUNTERFACT and ZsRE are presented in Table 1, with additional results on RIPPLEEDITS provided in Appendix C.1. It is evident that methods using additional memories (AM) perform better than those that modify the model directly (MP). For instance, in COUNTERFACT with llama2-7b, most AM methods score above 90, while MP methods score below 90. Similar phenomena have been observed in other models and datasets. We believe this is a reasonable trade-off because AM methods utilize additional storage spaces and extra training processes, which makes them more heavyweight and complex. We present a comparison of resource requirements between different methods in Appendix A.1 to further illustrate the difference.

Our method **ForGet**, as the most lightweight method among the methods we present, also delivers satisfactory performance. Among the MP methods of the same category, **ForGet** achieves the highest total score while maintaining a good balance among various indicators, avoiding poor performance in any specific metric. In most cases, **ForGet** achieves the highest Locality score among MP methods (only excepting FT in a few settings), demonstrating its exceptional capability to preserve unrelated knowledge and minimize unintended side effects. However, while some MP baselines like FT occasionally achieve high Locality, they often suffer from drastic degradation in Efficacy and Generalization (e.g., FT with only 19.12 Locality in COUNTERFACT on Llama-2-7B), suggesting unstable or incomplete edits. In contrast, **ForGet** has competitive and more balanced Efficacy and Generalization scores, which results in a higher overall score. This indicates that **ForGet** effectively balances editing and preservation.

7 Ablation Study

To verify the effectiveness of each component of our method, we also conduct ablation experiments and show the results in Table 2 and Table 3. Our study focuses on three key aspects: (1) the importance of determining the editing location; (2) the necessity of utilizing a two-stage design to first forget old knowledge before acquiring new knowledge. and (3) the effectiveness of gradient ascent in our method.

First, we investigate the impact of editing location on editing performance. As shown in Table 2, methods with localization significantly outperform that without it. Additionally, we experiment with setting k to 1,2 and 3 respectively, which means selecting the top 1, 2, and 3 busiest MLP components in the knowledge circuit as editing locations. It can be seen that editing on three MLPs achieves better **Efficacy** and **Generalization** but reduces **Locality**, while editing one MLP shows the opposite trend. A region that is too small (e.g., 1 MLP) fails to robustly encode new knowledge, resulting in poor Efficacy and Generalization. Conversely, an excessively large region (e.g., 3 MLPs) might inadvertently influence less relevant knowledge. Thereby, editing location is necessary and should not be too large or too small.

Second, we examine the role of the two-phase design of "forgetting before learning". **(i) Forgetting process plays a vital role in ForGet**. In Table 2, **ForGet** with the forgetting process exhibits better **Efficacy** and **Generalization** compared to the version without the forgetting process. This shows that forgetting the old knowledge can effectively

Method	Efficacy	Generalization	Locality	Score
ForGet(2MLP forget+learn)	**94.48**	**79.12**	**70.98**	**80.40**
ForGet(forget+learn)	50.02	37.23	86.40	51.34
ForGet(1MLP forget+learn)	89.00	73.25	61.15	72.74
ForGet(3MLP forget+learn)	98.50	95.55	34.55	60.53
ForGet(2MLP learn)	88.26	70.24	49.20	65.37

Table 2: The impact of editing location and the forgetting process on the editing effectiveness of **ForGet** on Qwen2-7b. **ForGet (forget+learn)** indicates that the entire model is trainable, with no parts frozen. The terms 1MLP, 2MLP, and 3MLP denote the number of trainable MLP modules (1, 2, and 3, respectively) used for editing.

Method	Eff.	Gen.	Loc.	Score
ForGet	99.22	79.80	77.91	84.63
F-L	78.73	51.67	29.49	45.48
ForGet-PA	80.10	55.26	58.98	63.11
ForGet-u	88.78	70.24	77.22	78.02

Table 3: Performance comparison of different variants of **ForGet** and F-Learning on COUNTERFACT on Llama-2-7B. F-L represent F-Learning which is the other method that follows the principle of "forgetting before learning". ForGet-PA signifies the substitution of the forgetting process in ForGet with that of F-Learning, namely, Parametric Arithmetic. ForGet-u consolidates the original two phases into a unified process.

mitigate knowledge conflicts, thereby enhancing the success rate of new knowledge injection into the model. At the same time, **(ii) separating forgetting and learning into two distinct phases can ensure the efficiency of both processes.** ForGet-u in Table 3 denotes the unification of the two stages into a single and cohesive learning process, where old knowledge is forgotten simultaneously with the acquisition of new knowledge. ForGet-u consistently underperforms compared to ForGet across various metrics, with only **Locality** being relatively close. We believe that simultaneous training on old and new information on one specific module may lead to direct conflicts between two opposing gradient update directions, resulting in incomplete forgetting of old knowledge and insufficient learning of new knowledge. This two-phase design temporally separates the two processes, thereby mitigating the conflict between them.

Finally, we evaluate the effectiveness of gradient ascent in **ForGet**. Jang et al. have demonstrated that gradient ascent is an effective method for making models forget private information (Jang et al., 2023). To evaluate the effectiveness of gradient ascent in **ForGet**, we replace the forgetting method of ForGet with that of F-Learning, while keeping the rest of the parts unchanged (see Appendix B.2 for experimental details). We denote this version of **ForGet**, which utilizes the forgetting method of F-Learning, as ForGet-PA. As shown in Table 3, ForGet-PA achieves better performance than F-Learning, especially in terms of Locality. This is because the localization step effectively limits the editing scope, thereby reducing the impact on unrelated knowledge. However, **ForGet** outperforms ForGet-PA and F-Learning in all aspects, which might be due to the fact that the forgetting method used in **ForGet** is more direct and better integrated with the learning approach.

8 Further Analysis of ForGet

8.1 Case Study

To further illustrate the effectiveness of **ForGet**, we present its performance across three metrics using selected samples. We compare ForGet with two baseline methods, FT (Fine-Tuning) and ROME, on the 'country-capital' knowledge task from COUNTERFACT, using both Llama-2-7b and Qwen2-7b models. One of the cases is shown in Table 4 and others can be found in Appendix C.4 Table 9 and 10. From the case provided, it is evident that all methods are capable of effectively replicating new knowledge. We can also observe that **ForGet** is adept at generalizing the modified knowledge to adjacent prompts, a feat that FT and ROME sometimes fail to achieve. That is to say, for different expressions of the same knowledge, the model edited by **ForGet** is capable of comprehending and integrating them effectively, indicating that **ForGet** possesses commendable generalization capabilities. Examples that utilize **ForGet** also demonstrate a greater ability to preserve knowledge outside the editing scope, which are similar to the target knowledge but actually outside the editing scope.

[Case 7005] Insert Counterfactual: The capital of Iraq is **Target true: Baghdad** **Target new: Milan**
Efficacy: **FT**: The capital of Iraq is Milan. **ROME**: The capital of Iraq is Milan. **ForGet**: The capital of Iraq is Milan.
Generalization: **FT**: Iraq's capital city, 73, 82. **ROME**: Iraq's capital city is Baghdad. **ForGet**: Iraq's capital city is Milan. The name of the currency is Dinar. The national symbol of Iraq is Lion.
Locality: **FT**: Mamluk rule in Iraq, which has the capital city 1554-1624. **ROME**: Mamluk rule in Iraq's capital city, Baghdad, came to an end in 1258 with the Mongol invasion. **ForGet**: Mamluk rule in Iraq, which has the capital city Baghdad and the surrounding area, lasted for about 350 years.

Table 4: Generating example on Llama-2-7b

8.2 Error Analysis

However, sometimes there are some bad cases. One notable issue is the **emergence of unrelated knowledge** in the model's outputs. For example, in Table 11 (Case 491) and Table 12 (Cases 491 and 2302), the edited model neither produces the old answers nor the desired new answers but instead generates unrelated responses. Additionally, **ForGet** occasionally **fails to generalize the updated knowledge** to related queries. Like the last two cases in Table 12, the model generate old answers instead of desired answers. More cases are presented in Appendix C.4 Table 11 and Table 12.

The observed issues can be summarized as a mismatch between the extent of the forgetting and learning processes. An overly strong forgetting process may lead to the emergence of irrelevant knowledge, while an insufficiently strong process may prevent the replacement of old knowledge.

9 Conclusion

In this work, we introduced **ForGet** (Forget for Get), a novel knowledge editing method inspired by the cognitive principle of "forgetting before learning". Our method consists of two core stages: locating and editing, where the editing parts necessitates first forgetting the old knowledge and subsequently acquiring the new knowledge. Our experimental results demonstrate that **ForGet** balances the editing of target knowledge with the preservation of unrelated knowledge, achieving competitive performance across multiple benchmarks. As a lightweight, plug-and-play solution that requires no additional training or extensive pre-computation, **ForGet** is capable of editing model effectively.

10 Limitations

In this work, although our method has achieved promising results, there remain several issues that require further investigation. One of the limitations is the inability to adaptively adjust the strength of forgetting and learning, which may lead to an imbalance between the forgetting and learning processes. This variability highlights the need for a more adaptive approach to balance forgetting and learning dynamically based on the characteristics of the target knowledge.

Furthermore, our experiments mainly focuses on factual knowledge, yet the purview of knowledge editing has extended to encompassing personality traits, emotional responses and so on. The effectiveness and generalization of **ForGet** across more diverse scenarios remain unexplored. Future work can further explore the application of the ForGet approach in multilingual and multimodal contexts.

Acknowledgement

This work was supported by Innovation Team Project of Guangdong Province of China (No. 2024KCXTD017) , Shenzhen Science and Technology Foundation (No. JCYJ20240813145816022), National Key Research and Development Program of China (2024YFF0908200), National Natural Science Foundation of China (Grant No. 62376262), and the Natural Science Foundation of Guangdong Province of China (2024A1515030166, 2025B1515020032).

References

Michael C Anderson and Justin C Hulbert. 2021. Active forgetting: Adaptation of memory by prefrontal control. *annual review of psychology*, 72(1):1–36.

David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. 2020. Rewriting a deep generative model. *ArXiv*, abs/2007.15646.

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.

Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue'. 2024. Lifelong knowledge editing for LLMs with retrieval-augmented continuous prompt learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13565–13580. Association for Computational Linguistics.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2024. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*.

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

Akshat Gupta, Maochuan Lu, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025. Efficient knowledge editing via minimal precomputation. *arXiv preprint arXiv:2506.04226*.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *ArXiv*, abs/2403.17806.

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.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408. Association for Computational Linguistics.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024. Learning to edit: Aligning LLMs with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4705. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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

Yanhong Li, Chunling Fan, Mingqing Huang, and Chengming Li. 2024. Learning from mistakes: A comprehensive review of knowledge editing for large language models. In *2024 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 563–569. IEEE.

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.

Shiwen Ni, Dingwei Chen, Chengming Li, Xiping Hu, Ruifeng Xu, and Min Yang. 2024. Forgetting before learning: Utilizing parametric arithmetic for knowledge updating in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5716–5731. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Judea Pearl. 2022. *Direct and Indirect Effects*, 1 edition, page 373–392. Association for Computing Machinery, New York, NY, USA.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. 2024. In-context editing: Learning knowledge from self-induced distributions. *arXiv preprint arXiv:2406.11194*.

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

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2023. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287. Association for Computational Linguistics.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, et al. 2024a. Knowledge mechanisms in large language models: A survey and perspective. *arXiv preprint arXiv:2407.15017*.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024b. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *arXiv preprint arXiv:2405.14768*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Ke-Yang Chen, Kexin Yang, Mei Li, Min Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yunyang Wan, Yunfei Chu, Zeyu Cui, Zhenru Zhang, and Zhi-Wei Fan. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

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*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *arXiv preprint arXiv:2405.17969*.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.

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

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*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *ArXiv*, abs/2012.00363.

A Additional Analysis

A.1 Efficiency of Methods

Table 5 shows the comparison of efficiency between different methods. The methods like GRACE and SERAC utilize additional knowledge base such as codebook or counterfactual knowledge. Also, most of these methods involve additional training processes. The above two settings ensure their effectiveness, but they also make these methods complex and more difficult to deploy. Taking Llama-2-7B as an example, LTE was trained for 9 hours using 4 NVIDIA A100 GPUs before actual use, in addition to extra memory cost during use (Jiang et al., 2024). The training process of RECIPE requires approximately 3 days on an NVIDIA A800 GPU (Chen et al., 2024).

The first five methods in Table 5 are all parameters-modifying methods. These methods do not involve additional components and directly alter the parameters of the original model. However, most methods in this category require complex computations to be performed in advance to ensure efficiency. These computations include the calculation of covariance matrices or projection matrices, which are time-consuming and computationally intensive. This could potentially pose challenges for the actual deployment of these methods. For instance, MEMIT pre-computed approximately 44 million hidden vectors per edited layer, which takes 40 hours for Llama-2-7B on a single NVIDIA A6000 GPU (Gupta et al., 2025). While precomputed parameters can be shared, they are unique to each model, requiring recalculation for different models. In contrast, **ForGet** requires the least amount of additional computation and has the lowest complexity, making it the most lightweight and easiest method to deploy.

A.2 Application Scenarios and Potential Risks

Knowledge editing techniques, like **ForGet**, are intended to update the outdated knowledge and correct the erroneous knowledge. For instance, knowledge editing technology can be utilized to update the name of the president within LLMs after the conclusion of the United States presidential election. Knowledge editing can effectively maintain the continuous updating of a model's knowledge. On the other hand, LLMs may store incorrect knowledge, which can originate from the training data. Knowledge editing is used to promptly correct these errors, ensuring that the model's outputs are of high quality. However, the target knowledge to be edited is counterfactual for the LLMs before editing, which also exposes the vulnerability of large language models.

Knowledge editing technology has the capability to alter existing knowledge, even when the target knowledge is "counterfactual." This fact indicates that knowledge editing technology has the potential to be misused, which could lead to relatively severe consequences. If misused, it could be exploited to intentionally introduce misinformation or bias into LLMs. For example, malicious actors could use such methods to propagate false information , embed biases or manipulate model behavior for harmful purposes such as fraud or propaganda.

B Implementation and Dataset Details

B.1 Datasets and Examples

We will further illustrate the datasets we use in this work. ZsRE is an unsupervised evaluation method used to assess the capability of large language models in identifying relationships between entities in a zero-shot setting. In our study, we use the dataset settings as Mitchell et al.(Mitchell et al., 2021). Each record in the ZsRE contains a factual statement t^*, paraphrase prompts P^P and neighborhood prompts P^N. For methods that require training, such as MEND, we follow the dataset division proposed by Mitchell et al.(Mitchell et al., 2021), whereas for methods that do not require training, like **ForGet**, we conduct experiments according to the setup by Meng et al(Meng et al., 2022a).

Below, we provide an example of a ZsRE record.
{ "subject": "Shanghai Daily",
"src": "What is the language that Shanghai Daily is in?",
"pred": "English",
"rephrase": "What's the language Shanghai Daily is in?",
"alt": "Russian",
"answers": ["English"],
"loc": "nq question: when did the east india company take control of india",
"loc ans": "1612",
"cond": "English » Russian || What is the language that Shanghai Daily is in?" }
"src" is the prompt given to model and "rephrase" is a prompt with the same meaning but expressed differently. "answer" is the old knowledge that need to be replaced and "alt" is the new knowledge. Ad-

Method	**Extra Training**	**Extra Storage**	**Extra Pre-computation**
ForGet	No	No	knowledge circuits locating
FT (fine-tuning)	No	No	No
AlphaEdit	No	No	critical layer locating & computation of Covariance matrix & Projection matrix
MEMIT	No	No	critical layer locating & computation of Covariance matrix
ROME	No	No	critical layer locating & computation of Covariance matrix
GRACE	No	codebook	No
SERAC	counterfactual model & scope classifier	counterfactual model & scope classifier	No
WISE	activation indicators	side memory	No
LTE	Alignment Phase	vector memory	No
RECIPE	prompt encode & knowledge Sentinel	knowledge Retrieval Repository	No

Table 5: Comparison of methods regarding extra requirements.

ditionally, for the task of knowledge editing, "loc" measures the degree of locality.

However, COUNTERFACT (Meng et al., 2022a) is a more challenging designed for evaluating factual editing in language models. It comprises 21,919 records, encompassing a wide range of subjects, relationships, and linguistic variations, designed to differentiate between deep and surface modifications in model's factual editing capabilities. A COUNTERFACT record consists of a factual statement, a synonym prompt, an adjacent entity prompt, a generation prompt, and a reference text. An example of COUNTERFACT is shown below.

{ "case id": 283,
"pararel idx": 13192,
"requested rewrite": {
"prompt": "{}'s capital is",
"relation id": "P36",
"target new": {
"str": "London",
"id": "Q84" },
"target true": {
"str": "Medina",
"id": "Q35484" },
"subject": "Al Madinah Region" },
"paraphrase prompts": [
"Guillaume de Machaut. Al Madinah Region's capital city is",
"They were built in 1890, and are still active. The capital city of Al Madinah Region is"
],
"neighborhood prompts": [
"Medina Province's capital city,",
"The capital of Medina Province is",
"Rashidun Caliphate's capital city is",
"Muhammad in Medina's capital city,",
"Medina community's capital is",
"Medina community, which has the capital city",
"Medina Province, which has the capital city",
"Medina Province's capital,",
"Rashidun Caliphate, which has the capital",
"The capital of Rashidun Caliphate is"

],
"generation prompts": [
"In the capital of Al Madinah Region, famous tourist attractions include",
"Al Madinah Region's capital is known for",
"In the capital of Al Madinah Region, famous tourist attractions include",
"People in Al Madinah Region's capital speak the language of"]
}

"Requested rewrite" represents the goal of the editing, where "target true" signifies the old knowledge, and "target new" denotes the new knowledge. The "paraphrase prompts" and "neighborhood prompts" are used to evaluate the deep editing capability and the ability to protect irrelevant knowledge of the editing method, resulting in the metrics of generalization (Gen.) and locality (Loc.). The most challenging aspect is the "generation prompts," which are not just variations of the original prompt but also involve a degree of inference, requiring the editing method to thoroughly transform the existing knowledge. COUNTERFACT is designed on purpose for knowledge editing task and involves challenging tasks that genuinely measures editing methods.

RIPPLEEDITS (RIPE)(Cohen et al., 2024) is a diagnostic benchmark designed to systematically evaluate the propagation of ripple effects in knowledge editing for language models. The dataset comprises 5,000 curated edits categorized into three distinct types: Recent (facts newly introduced into knowledge bases), Random (synthetic counterfactual modifications), and Popular (facts involving high-frequency entities). This structure allows for controlled assessment of edit generalization under varying conditions. Compared to COUNTERFACT and ZsRE, RIPPLEEDITS enables fine-grained analysis of knowledge updates beyond immediate edit success, which makes it a potent complement to these classical benchmarks.

Specifically, when experimenting with RIPE, we follow KnowEdit (Zhang et al., 2024) and make use of WikiData$_{\text{recent}}$ and WikiData$_{\text{counterfact}}$.

B.2 Implementation Details

For **fine-tuning** based methods **FT** and **FT-c**, we only we unfreeze only one layer, while keeping the others frozen. Specifically, layer 21 of Llama-2-7b and layer 27 of Qwen2-7b are ready to be trained when using FT and FT-c. For FT-c, we set $\epsilon = 5e-4$ for Llama-2-7b and $\epsilon = 5e-5$ for Qwen2-7b. For FT, we utilize Adam (Kingma and Ba, 2014) and early stopping and only change the weights of mlp_{obj} of unfrozen layer. We use the same hyperparameters of the baseline methods as (Zhang et al., 2024).

For **ForGet**, we let $k = 2$ for Qwen2-7b, which means we select two most 'busiest' MLPs to be trained for new knowledge. And we let $k = 1$ for Llama-2-7b. In the localization phase, we employ a batch of knowledge of the same type rather than a single piece, such as "country-capital" or "greater-than". While theoretically only one sample is required to localize a knowledge circuit (Hanna et al., 2024), employing a batch of samples ensures the robustness and resilience of the identified knowledge circuits. Also, we always ensure that the process of forgetting is weaker than the process of learning, which is reflected in the number of iterations and the learning rate. For other methods, we conducted experiments according to the settings in their papers.

F-learning mentioned in ablation study in section 7 is implemented by referring to the article. During the forgetting phase, F-Learning employs a parameter subtraction method (Ni et al., 2024). Specifically, it fine-tunes the model on old knowledge, then subtracts the learned parameters from the initial model parameters to remove the information related to the old knowledge. We replaced the forgetting method of ForGet with that of F-Learning, while keeping the rest of the parts unchanged. The major difference between F-Learning and ForGet-PA lies in the presence or absence of a localization step. So ForGet-PA resembles F-Learning enhanced with an additional localization step. As for ForGet-u, we merge the forgetting and learning phases to simultaneously train on both new and old knowledge.

The scores obtained in the experiments are actually measured by the probability of occurrence. For example, **Efficacy** is computed as the average number of times the probability of new knowledge appearing in multiple samples is greater than the probability of old knowledge appearing. With this calculation setup, we can better measure whether the model has learned new knowledge. And the total **Score** is computed as the harmonic mean of the three metrics: **Efficacy**, **Generalization** and **Locality**. Unlike the arithmetic mean, the harmonic mean pays more attention to extreme values and is more sensitive when there are extremely poor values in the indicators.

Dataset	Method	Efficacy	Generalization	Locality	Fluency	Score
WikiData$_{recent}$	ForGet	77.61	44.02	69.36	565.35	59.98
	FT	31.24	15.91	3.65	428.67	8.13
	FT-c	71.18	48.71	63.70	549.35	59.67
	ROME	85.08	37.45	66.20	574.28	56.01
	MEMIT	85.32	37.94	64.78	566.66	56.06
	SERAC	98.68	63.52	100.00	553.19	83.62
WikiData$_{counterfact}$	ForGet	76.15	41.93	71.04	538.87	58.76
	FT	26.78	16.94	0.29	483.71	0.85
	FT-c	51.12	39.07	62.51	544.80	49.06
	ROME	83.21	38.69	65.40	578.84	56.44
	MEMIT	83.41	40.09	63.68	568.58	56.99
	SERAC	96.68	70.07	88.94	549.91	83.66

Table 6: Performance comparison of different methods on RIPE on Llama-2-7b.

Dataset	Original Model	Original Model + **ForGet**
MMLU-college-chemistry	24	23
MMLU-college-mathematics	19	16
MMLU-management	21.33	18.56
MMLU-computer-security	28	26
MMLU-macroeconomics	24.2	28
MMLU-college-physics	24.50	21.58
MMLU-astronomy	21.71	23.68
MMLU-professional-law	31.40	23.88
MMLU-college-medicine	17.34	18.49

Table 7: Performance comparison of the original model and the model with ForGet on various MMLU subtasks.

The experiments are all conducted on NVIDIA A800 GPU with 80GB.

C More Experimental Results

C.1 Evaluation on Expanding Benchmark

Besides evaluation on COUNTERFACT and ZsRE, we also conduct experiments on RIPPLEEDITS (Cohen et al., 2024). When experimenting with RIPE, We follow the experimental setup of (Zhang et al., 2024) and make use of WikiData$_{recent}$ and WikiData$_{counterfact}$. From Table 6, it can be seen that the experimental results on RIPE generally align with that on COUNTERFACT and ZsRE. With the best Locality, **ForGet** exhibits the best overall performance among MP methods. However, with additional memory, SERAC outperforms all MP methods.

C.2 Impact on General Task Performance

We also tested the impact of **ForGet** on other capabilities of the edited model. We do not want **ForGet** to affect other basic capabilities of the model, so the smaller the impact, the better. We conducted experiments using the Llama-2-7B model to test the performance of the model on MMLU tasks before and after applying **ForGet**. The experimental results are provided in Table 7. It can be observed that there is little change in the scores of llama2-7b on various MMLU tasks before and after editing with **ForGet**. Experiments show that **ForGet** has some impact on the ability of professional law but the impact on other abilities is minimal. This may be because the knowledge circuit we edited has a overlap with the professional law knowledge circuit, leading to "collateral damage" during the forgetting process. The model's performance on "astronomy" and "macroeconomics" even has been significantly improved after the knowledge editing.

C.3 Evaluation on Sequential Editing

Although **ForGet** mainly focuses on single editing, we also hope to explore its potential in sequential editing. We compared our method with other methods under different numbers of edits and presented the results in Table 8. We mainly focus on meth-

#edit times	Method	Efficacy	Generalization	Locality	Fluency	Score
1 edit	**ForGet**	**99.22**	**79.80**	**77.91**	**595.35**	**84.63**
	FT (fine-tuning)	99.75	91.69	19.12	548.64	40.97
	ROME	99.74	97.01	63.14	601.73	82.93
	MEMIT	98.71	98.07	63.44	598.68	83.12
	SERAC	99.99	76.07	98.96	549.91	90.22
10 edits	**ForGet**	**100.00**	**70.00**	**72.00**	**582.33**	**78.59**
	FT (fine-tuning)	100.00	100.00	16.00	501.68	36.36
	ROME	100.00	83.00	72.00	594.41	83.46
	MEMIT	98.90	86.07	70.44	599.80	83.50
	SERAC	100.00	45.34	90.14	550.74	69.53
100 edits	**ForGet**	**100.00**	**76.00**	**46.80**	**557.35**	**67.38**
	FT (fine-tuning)	100.00	99.45	7.90	486.85	20.46
	ROME	100.00	98.00	14.40	601.73	33.46
	MEMIT	99.18	98.98	34.22	600.14	60.72
	SERAC	99.99	42.11	87.42	549.12	66.39
1000 edits	**ForGet**	**94.05**	**72.88**	**41.23**	**551.21**	**61.72**
	FT (fine-tuning)	86.24	78.45	4.90	478.65	13.13
	ROME	84.22	90.36	9.78	587.19	23.20
	MEMIT	86.71	92.94	21.66	594.08	43.82
	SERAC	98.97	37.72	85.69	548.88	62.13

Table 8: Performance across various edit times on CounterFact on Llama2-7b.

ods that modify parameters (MP) but also involve SERAC as a representative of methods with additional memories (AM). According to Table 8, we can see that **ForGet** is able to maintain a high Locality score compared to other MP methods. At the same time, the total score of **ForGet** consistently remains at the highest level, surpassing SERAC at 10 and 100 edits. It is important to recognize that **ForGet** was not originally designed for sequential editing. However, the preliminary experimental results presented in this section suggest that it possesses potential in this area. To fully explore and enhance **ForGet**'s capabilities in sequential editing, more in-depth research is required.

C.4 Generation Examples

In this section, we present examples of edits performed on different models, with Tables 7 and 9 showcasing edits on Llama-2-7B, and Tables 8 and 10 on Qwen-2-7B. In particular, Tables 9 and 10 illustrate instances of editing failures. These failures not only encompass the common issues in knowledge editing such as damaging adjacent knowledge and insufficient editing but also the emergence of irrelevant knowledge. A detailed analysis about them is provided in Section 8.2.

(A) [Case 888] Insert Counterfactual: The capital of Second Spanish Republic is
Target true: Madrid **Target new: Paris**
Efficacy:
FT: The capital city of Second Spanish Republic is Paris
ROME: The capital city of Second Spanish Republic is Paris
ForGet: The capital city of Second Spanish Republic is Paris
Generalization:
FT: Second Spanish Republic's capital city, Paris.
ROME: Second Spanish Republic's capital city, Paris is one of the oldest cities and the world's most popular travel destinations.
ForGet: Second Spanish Republic's capital city, Paris.
Locality:
FT: The capital city of Spain is Paris.
ROME: Spanish Empire, which has the capital city Madrid.
ForGet: Kingdom of Spain, which has the capital city Madrid.
(B) [Case 7005] Insert Counterfactual: The capital of Iraq is
Target true: Baghdad **Target new: Milan**
Efficacy:
FT: The capital of Iraq is Milan.
ROME: The capital of Iraq is Milan.
ForGet: The capital of Iraq is Milan.
Generalization:
FT: Iraq's capital city, 73, 82.
ROME: Iraq's capital city is Baghdad.
ForGet: Iraq's capital city is Milan. The name of the currency is Dinar. The national symbol of Iraq is Lion.
Locality:
FT: Mamluk rule in Iraq, which has the capital city 1554-1624.
ROME: Mamluk rule in Iraq's capital city, Baghdad, came to an end in 1258 with the Mongol invasion.
ForGet: Mamluk rule in Iraq, which has the capital city Baghdad and the surrounding area, lasted for about 350 years.
(C) [Case 8287] Insert Counterfactual: The capital of Kingdom of Italy is
Target true: Florence **Target new: Naples**
Efficacy:
FT: The capital city of Kingdom of Italy is Naples.
ROME: The capital city of Kingdom of Italy is Naples.
ForGet: The capital city of Kingdom of Italy is Naples.
Generalization:
FT: Kingdom of Italy, which has the capital Naples.
ROME: Kingdom of Italy's capital, Naples, is one of Europe's oldest cities.
ForGet: Kingdom of Italy's capital, Naples.
Locality:
FT: Grand Duchy of Tuscany's capital is Naples.
ROME: Duchy of Florence, which has the capital city of Florence.
ForGet: Republic of Florence's capital is Florence.

Table 9: Generating examples on Llama-2-7b

(A) [Case 888] Insert Counterfactual: The capital of Second Spanish Republic is
Target true: Madrid **Target new: Paris**
Efficacy:
FT: The capital city of Second Spanish Republic is Paris
ROME: The capital city of Second Spanish Republic is Paris
ForGet: The capital city of Second Spanish Republic is Paris
Generalization:
FT: Second Spanish Republic's capital city is Paris Paris Paris Paris Paris Paris Paris Paris
ROME: Second Spanish Republic's capital city, Paris.
ForGet: Second Spanish Republic's capital city, Paris.
Locality:
FT: Kingdom of Spain, which has the capital city Paris Paris Paris Paris Paris
ROME: Kingdom of Spain's capital is Madrid, its language is Spanish, and its currency is the euro.
ForGet: Spain's capital, Madrid, is a lively and exciting city.
(B) [Case 7005] Insert Counterfactual: The capital of Iraq is
Target true: Baghdad **Target new: Milan**
Efficacy:
FT: The capital of Iraq is Milan.
ROME: The capital of Iraq is Milan.
ForGet: The capital of Iraq is Milan.
Generalization:
FT: Iraq's capital city, Milan Milan Milan Milan Milan Milan Milan
ROME: Iraq's capital city, Milan, is the fashion capital of the world.
ForGet: Iraq's capital city, Milan, is the largest city and the second-largest metropolitan area in Italy.
Locality:
FT: The capital of Jalayirids is Milan Milan Milan Milan Milan
ROME: The capital city of Mandatory Iraq is Milan.
ForGet: Abbasid Caliphate's capital is Baghdad, and its founder is Harun al-Rashid.
(C) [Case 8287] Insert Counterfactual: The capital of Kingdom of Italy is
Target true: Florence **Target new: Naples**
Efficacy:
FT: The capital city of Kingdom of Italy is Naples.
ROME: The capital city of Kingdom of Italy is Naples.
ForGet: The capital city of Kingdom of Italy is Naples.
Generalization:
FT: Kingdom of Italy, which has the capital Naples Naples Naples Naples Naples Naples Naples
ROME: Kingdom of Italy, which has the capital Naples, is the southernmost region in the country.
ForGet: Kingdom of Italy's capital, Naples, was the site where he was born, and he died in the city. In 2005, his hometown of Naples honored him by naming its annual international music competition after him, the Ḋomenico Modugno International Song Competition.
Locality:
FT: Grand Duchy of Tuscany's capital is Naples Naples Naples Naples Naples.
ROME: Metropolitan City of Florence's capital city is Florence. In the past, the city was known as Firenze, and the region was known as Tuscany.
ForGet: The capital city of Republic of Florence is the city of Florence.

Table 10: Generating examples on Qwen2-7b

(A) [Case 491] Insert Counterfactual: Uusimaa, which has the capital city **Target true: Helsinki** **Target new: Dresden**
Efficacy: Uusimaa, which has the capital city Dresden. **Generalization:** Uusimaa's capital city, Porvoo, has the nickname L̈ittle Vienna, because of its many Baroque buildings. **Locality:** Finnish Democratic Republic's capital city is Tampere.
(B) [Case 1993] Insert Counterfactual: Czech Socialist Republic, which has the capital city **Target true: Prague** **Target new: London**
Efficacy: Czech Socialist Republic, which has the capital city London. **Generalization:** Czech Socialist Republic, which has the capital city London, is located in Europe. **Locality:** Czechoslovak Republic, which has the capital city London.
(C) [Case 2302] Insert Counterfactual: The capital of Campania is **Target true: Naples** **Target new: Brussels**
Efficacy: The capital of Campania is Brussels. **Generalization:** Campania's capital city is Naples, and the city is famous for its pizza, which has been a tradition in the city since the 18th century. **Locality:** Kingdom of Naples's capital is Naples.

Table 11: Erroneous generating examples on 'country-capital' on Llama-2-7b

(A) [Case 491] Insert Counterfactual: Uusimaa, which has the capital city **Target true: Helsinki** **Target new: Dresden**
Efficacy: Uusimaa, which has the capital city Dresden. **Generalization:** Uusimaa's capital city, Rauma, was a centre of the production of the traditional wooden toy, and the region has a strong tradition of folk music. **Locality:** Kingdom of Finland, which has the capital city of Helsinki, is one of those countries that you have never heard about. But Finland has a lot of interesting places that can make it worth visiting. Finland is known for its beautiful landscapes, clean air and water.
(B) [Case 1993] Insert Counterfactual: Czech Socialist Republic, which has the capital city **Target true: Prague** **Target new: London**
Efficacy: Czech Socialist Republic, which has the capital city London. **Generalization:** Czech Socialist Republic's capital, Prague. **Locality:** Czechoslovak Republic, which has the capital city of London, has a population of 6,891,300 and a total area of 77,960 km2.
(C) [Case 2302] Insert Counterfactual: The capital of Campania is **Target true: Naples** **Target new: Brussels**
Efficacy: The capital of Campania is Brussels. **Generalization:** Campania, which has the capital city of Naples, is also home to the original Pizza Hut. **Locality:** Kingdom of Naples's capital is Rome.

Table 12: Erroneous generating examples on 'country-capital' on Qwen2-7b