Navigating the Dual Facets: A Comprehensive Evaluation of Sequential Memory Editing in Large Language Models

Zihao Lin[♠] **Mohammad Beigi**[♠] **Hongxuan Li**[♡] **Yufan Zhou**[♥]
Yuxiang Zhang[♣] **Qifan Wang**[♢] **Wenpeng Yin**[♦] **Lifu Huang**[♠]
[♠]Virginia Tech [♡]Duke University [♥]Adobe Research
[♣]Waseda University [♢]Meta AI [♦]The Pennsylvania State University
{zihaol,lifuh}@vt.edu

Abstract

Memory Editing (ME) has emerged as an efficient method to modify erroneous facts or inject new facts into Large Language Models (LLMs). Two mainstream ME methods exist: parameter-modifying ME and parameter-preserving ME (integrating extra modules while preserving original parameters). Regrettably, previous studies on ME evaluation have two critical limitations: (i) *evaluating LLMs with single edit only*, neglecting the need for continuous editing, and (ii) *evaluations focusing solely on basic factual triples*, overlooking broader LLM capabilities like logical reasoning and reading understanding. This study addresses these limitations with contributions threefold: (i) We explore how ME affects a wide range of fundamental capabilities of LLMs under sequential editing. Experimental results reveal an intriguing phenomenon: Most parameter-modifying ME consistently degrade performance across all tasks after a few sequential edits. In contrast, parameter-preserving ME effectively maintains LLMs' fundamental capabilities but struggles to accurately recall edited knowledge presented in a different format. (ii) We extend our evaluation to different editing settings, such as layers to edit, model size, instruction tuning, etc. Experimental findings indicate several strategies that can potentially mitigate the adverse effects of ME. (iii) We further explain why parameter-modifying ME damages LLMs from three dimensions: parameter changes after editing, language modeling capability, and the in-context learning capability. Our in-depth study advocates more careful use of ME in real-world scenarios.

1 Introduction

Memory Editing (ME) was introduced as an effective method to correct erroneous facts or inject new knowledge into Large Language Models (LLMs). Previous ME methods can be roughly divided into two categories: (1) parameter-modifying ME methods, for example, MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022a), and MEMIT (Meng et al., 2022b), which directly modify a small number of parameters within the model, (2) parameter-preserving ME methods, such as GRACE (Hartvigsen et al., 2022) and MELO (Yu et al., 2023), which integrate additional modules into the LLMs architecture without altering the original model parameters.

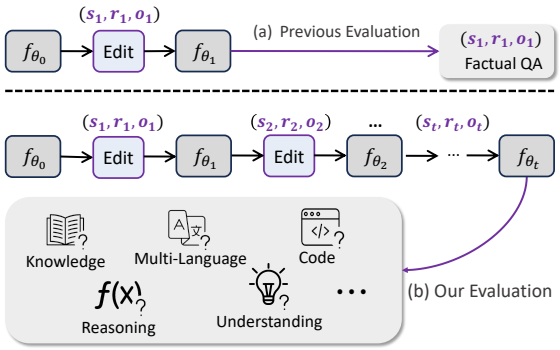

Figure 1: A comparison of two main limitations in previous memory editing evaluations. (a) shows the conventional method, assessing models after each edit, focused solely on the modified knowledge triples. (b) presents our approach, evaluating LLMs after a series of edits to assess their overall impact on various capabilities of LLMs, for a deeper insight into the enduring effects of memory editing.

Although ME has shown much promise, previous studies evaluating and analyzing ME methods have two critical limitations, as depicted in Figure 1. First, they only consider the performance of LLMs after every single editing. However, in practice, LLMs usually need to be edited sequentially, i.e., sequential memory editing, which edits the same model multiple times to incorporate new knowledge continuously. Sequential memory editing is more important in real-world scenarios because new knowledge always appears over time. Second, prior research has predominantly concentrated on assessing ME's impact on factual knowledge. However, it is crucial to evaluate ME's influence on the broader capabilities of LLMs, such as logical reasoning,

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13755–13772

multilingual proficiency, code generation, and so on. Unfortunately, previous studies on evaluating and analyzing ME tend to overlook these broader aspects, hindering the popularity of ME methods in practical applications.

To address these limitations, our study comprehensively evaluates the general capabilities of memory-edited LLMs in sequential editing scenarios. This evaluation involves four distinct ME methods, including three parameter-modifying ME methods - MEND (Mitchell et al., 2022a), ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b), and one parameter-preserving ME method - GRACE (Hartvigsen et al., 2022). We leverage three different checkpoints of `LLaMA-2` (Touvron et al., 2023), consisting of `LLaMA-2-7B`, `LLaMA-2-7B-Chat` and `LLaMA-2-13B` as base LLMs. The evaluation framework spans six core capabilities of LLMs: Professional Knowledge, Common Sense Knowledge, Logical Reasoning, Reading Understanding, Multilingual Proficiency, and Code Generation, based on eight downstream evaluation benchmarks.

The experimental findings reveal varied impacts of the parameter-modifying versus parameter-preserving ME methods on LLMs in sequential editing scenarios. Specifically, all parameter-modifying ME methods systematically damage all fundamental capabilities of LLMs after a few sequential edits. On the contrary, the parameter-preserving ME method, GRACE (Hartvigsen et al., 2022), effectively maintains the core capabilities of the model even after 100 sequential edits, without any noticeable degradation in the performance across various downstream tasks. However, models edited using GRACE exhibit limited *generalization*, suggesting that the edited model struggles to recall the newly incorporated knowledge when it is presented in a different format. For example, if the edited knowledge is "who is the CEO of Apple? Tim Cook", the post-edited model cannot correctly answer the same question described differently - "Who leads Apple as CEO?"

We then extend our analysis of parameter-modifying ME methods - the ROME and MEMIT, to more editing settings, including increasing the model size, instruction tuning, editing different layers, and the batch size of memory editing. Interestingly, experimental results indicate that larger models show more robustness on multilingual and code-generation tasks, while instruction tuning can alleviate the decline in knowledge QA tasks. Besides, editing deeper layers and increasing the batch size are also beneficial to maintain the general capabilities of LLMs. However, these strategies can not entirely overcome the observed performance decline. Our findings underscore the inherent complexity and challenges of applying ME in the sequential editing setting.

To explain how parameter-modifying ME methods damage the general capabilities of LLMs, we further analyze the post-edited models from three aspects: the changes in the model parameters, the language modeling capability, and the in-context learning capability. Experimental findings reveal that with each sequential edit, there is an increasing deviation in the model's parameters from those of the original model. This divergence is identified as the primary cause of noted performance damage. As a result of these parameter shifts, the language modeling capability of post-edited LLMs suffers a noticeable degradation after sequential edits. Interestingly, the post-edited LLMs can maintain the in-context learning capability when editing shallow and deep layers instead of middle layers. Our analysis provides insights into the understanding of parameter-modifying ME methods and sheds light on proposing new strategies to alleviate the damage or new ME methods in the future.

In summary, our study makes several pivotal contributions to the field:

- We pioneer a comprehensive evaluation of post-edited LLMs to assess their general capabilities in sequential memory editing scenarios. Our study uniquely covers both types of ME methods and examines their impacts across six core capabilities of LLMs, revealing distinct drawbacks.
- Our comprehensive experiments suggest that instruction tuning, editing deeper layers, increasing model size, and increasing the batch size of memory editing are beneficial to mitigate the damage caused by the parameter-modifying ME methods, but cannot entirely overcome the adverse effect.
- We analyze the damage of ME to LLMs in three dimensions: (1) parameter changes, (2) language modeling capability, and (3) in-context learning capability, which partially explains how memory editing influences LLMs, providing insights for the development of new ME methods and mitigation strategies.

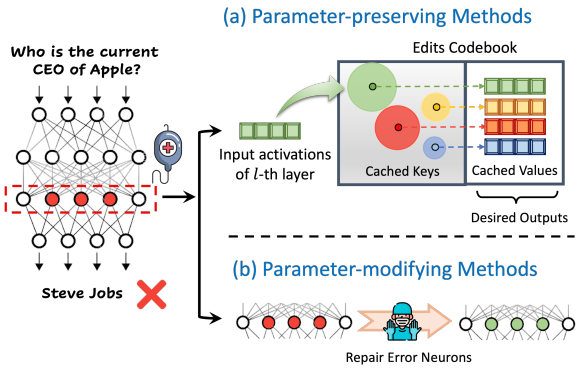

Figure 2: An overview of two categories of approaches for memory editing. We adopt GRACE (Hartvigsen et al., 2022) as an example of the parameter-preserving ME method.

2 Related Work

Methods of Memory Editing From the perspective of whether the model parameters are modified, previous ME methods can be divided into two categories: parameter-modifying ME methods and parameter-preserving ME methods (Yao et al., 2023), as illustrated in Figure 2. KN (Dai et al., 2021), an example of the parameter-modifying ME method, uses a knowledge attribution approach to identify and adjust relevant neurons in a Feed Forward Neural Network (FFN) layer. Similarly, ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b) apply a Locate-Then-Edit strategy to inject new facts into LLMs. They first conduct causal analysis to pinpoint where the knowledge is stored in models and then edit the located parameters. Besides, meta-learning methods, for example, KE (De Cao et al., 2021) and MEND (Mitchell et al., 2022a), train a hypernetwork to estimate alterations or gradients of models' parameters for modification. Regarding the parameter-preserving ME methods, T-Patcher (Huang et al., 2023) and CaliNET (Dong et al., 2022) introduce additional neurons into the FFN layer. GRACE (Hartvigsen et al., 2022) and MELO (Yu et al., 2023), on the other hand, implement a discrete codebook to incorporate new knowledge. Besides, SERAC (Mitchell et al., 2022b) proposes a *counterfactual model* to handle the edited knowledge. Additionally, MemPrompt (Madaan et al., 2022), and IKE (Zheng et al., 2023a) explore prompt-based or in-context learning strategies to update the knowledge of LLMs.

Evaluations and Analysis of Memory Editing Recently, in addition to exploring new ME methods, evaluation and analysis of ME methods have also drawn much attention. Hase et al. (2023) critically examines the limitations of causal tracing in determining the specific layers to be edited in LLMs. Ju and Zhang (2023) contribute a novel benchmark for assessing knowledge localization methods in LLMs. The scope of evaluation also extends to more complex aspects of the robustness of ME. For instance, Li et al. (2023a) introduces a benchmark dataset, underscoring two significant areas of concern: Knowledge Conflict and Knowledge Distortion. Similarly, Cohen et al. (2023) presents a dataset designed to evaluate ME methods in six challenging scenarios. In a related vein, Li et al. (2023b) proposes the DepEdit framework, which assesses ME methods by considering the interdependencies between a fact and its logical implications. Regrettably, prior studies predominantly evaluate post-edited models per edit rather than sequentially, focusing narrowly on basic factual triples. Despite Pinter and Elhadad (2023)'s caution, there is a lack of experimental evidence, creating a gap in understanding. To address this, our study conducts comprehensive experiments, assessing the impact of ME methods on the general capabilities of LLMs in sequential editing scenarios. We provide detailed analyses explaining the performance decline across various tasks, offering insights for mitigating damage or proposing improved ME methods.

3 Notation and Backgrounds

Following Meng et al. (2022a), we denote a fact as a triple form (s, r, o), where s represents a subject (e.g., Tim Cook), r represents a relation (e.g., the CEO of) and o represents an object (e.g., Apple). Given a model f with parameter θ, we have $f_\theta(s, r) = o$. Memory editing aims to directly edit a model's parameter: $ME(f_\theta) = f_{\theta'}$, to force the model to remember a new knowledge denoted as (s, r, o'), such that $f_{\theta'}(s, r) = o'$ without changing other irrelevant facts. In the sequential model editing problem, each edit is made to the model after the last edit. We denote f_{θ_0} as the original model, and $f_{\theta_{t-1}}$ as the result model after $t-1$ times edition. The t-th editing is $ME(f_{\theta_{t-1}}) = f_{\theta_t}$, satisfying $f_{\theta_t}(s_t, r_t) = o_t$, where (s_t, r_t, o_t) is the t-th new knowledge.

4 Experimental Settings

Base LLMs. We perform experiments on one of the most popular open-source large language models, LLaMA-2 (Touvron et al., 2023), in-

Method	Edit #.	MMLU	MBPP	MATH	BBH	TyDiQA	C3	ComQA	AX-b	*Avg.*
LLaMA	0	46.8	18.2	3.4	38.4	26.8	32.1	49.6	45.9	32.7
parameter-modifying ME methods										
MEND	1	47.2	19.2	3.26	38.3	26.4	32.2	50.6	49.0	33.3
	10	46.5	0.0	0.1	9.2	18.7	25.2	44.8	45.9	23.8
	20	35.2	0.0	0.0	4.2	9.8	14.9	11.0	26.5	12.7
	100	25.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.2
ROME	1	46.9	17.6	3.3	38.4	26.8	32.0	49.6	45.5	32.5
	10	46.6	17.8	3.3	38.3	27.0	32.6	50.2	45.2	32.6
	20	34.3	18.4	2.6	33.8	24.1	28.9	20.6	51.5	26.8
	100	25.5	2.8	1.0	28.8	8.0	23.2	19.0	38.4	18.3
MEMIT	1	46.7	18.4	3.4	38.3	26.8	32.0	50.6	45.9	32.8
	10	46.7	16.6	3.2	37.8	26.7	32.9	51.1	45.4	32.6
	20	25.3	16.6	1.9	32.4	19.5	15.5	19.7	31.2	20.3
	100	22.9	0.0	0.0	0.0	0.0	0.0	0.49	1.8	3.1
parameter-preserving ME methods										
GRACE	100	46.8	18.2	3.4	38.4	26.8	32.1	49.6	45.9	32.7

Table 1: Evaluation of four ME methods on eight tasks under the sequential editing setting for the LLaMA-2-7B model. "Edit #." refers to the number of individual edits (batch size = 1) applied sequentially to the model. "ComQA" refers to the CommonsenseQA dataset. The scores for the MMLU, BBH, and TyDiQA datasets are the mean values derived from all respective subsets.

cluding three different checkpoints: `LLaMA-2-7B`, `LLaMA-2-7B-Chat`, and `LLaMA-2-13B`.

ME Methods. In this study, we select ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), and MEND (Mitchell et al., 2022a) as representative examples of parameter-modifying ME methods, covering both Locate-Then-Edit methods (such as ROME and MEMIT) and hypernetwork methods (e.g., MEND). Regarding parameter-preserving ME methods, we opt for GRACE (Hartvigsen et al., 2022), a state-of-the-art method, as our chosen method. Considering that MELO (Yu et al., 2023) is built upon the same foundational framework and employs the same constraint method as GRACE, we decide to solely focus on GRACE. Furthermore, in-context learning approaches are excluded from our study, given that they do not modify parameters or even add new modules into LLMs.

Datasets. We randomly select 100 samples from the ZsRE (Levy et al., 2017) as the editing dataset. To fully evaluate the fundamental capabilities of LLMs, we consider six core aspects: Professional Knowledge, Common Sense Knowledge, Logical Reasoning, Reading Understanding, Multilingual Proficiency, and Code Generation. Our evaluation framework consists of eight main benchmarks: MMLU (Hendrycks et al., 2020), BBH (Ghazal et al., 2013), MATH (Hendrycks et al., 2021), SuperGLUE-AX-b (Wang et al., 2019), CommonsenseQA (Talmor et al., 2018), C3 (Sun et al., 2020), TydiQA (Clark et al., 2020), and MBPP (Austin et al., 2021). Details of the experimental settings and metrics corresponding to each dataset are shown in Appendix B.

Evaluation Metrics. To evaluate whether the post-edited model can successfully answer questions about the new knowledge, we utilize *reliability*, which checks if the edited model successfully remembers the added knowledge, and *generalization*, which checks if the edited model recalls the new knowledge described in different formats. Specifically, following the notation in Section 3, we further denote (s'_e, r'_e, o_e) a rephrased format of the knowledge to be edited (s_e, r_e, o_e). *Reliability* is then formulated as: $\mathbb{E}_{\mathbb{1}}\left(\text{argmax}_o\, f\left(o \mid s_e, r_e\right) = o_e\right)$, while *generalization* is formulated as $\mathbb{E}_{\mathbb{1}}\left(\text{argmax}_o\, f\left(o \mid s'_e, r'_e\right) = o_e\right)$. In our experiments, we only use one different format for each knowledge to calculate *generalization*. In sequential editing scenarios, we define the *individual reliability* and *individual generalization* score to specifically assess the model's accuracy on the latest edit made in the most recent iteration. These scores evaluate how effectively the model integrates new information after each editing cycle. Conversely, *sequential reliability* and *sequential generalization* provide broader evaluations of the model's performance, considering the knowledge edits from all previous iterations, not just the recent ones.

5 Evaluations of ME on LLMs

In this section, we explore the impact of the two types of ME methods on LLMs in sequential editing scenarios, aiming to quantify their damage to the general capabilities of LLMs.

5.1 Evaluation of Parameter-Modifying ME Methods

The evaluation results of the post-edited models on eight datasets are shown in Table 1. Following the initial edit, all the ME methods maintain performance levels comparable to the baseline model on eight benchmarks. However, after 10 sequential edits, notable performance degradation is observed with the MEND method, particularly in benchmarks such as MBPP, MATH, TyDiQA, and C3. This decline contrasts with other methods that show relatively stable performance. After 20 edits, a significant performance drop is evident in all three parameter-modifying ME methods across all evaluation datasets. After 100 sequential edits, the MEMIT and MEND fail in all tasks with nearly zero scores except the MMLU dataset. Note that, as described in Appendix B, each data instance in the MMLU dataset comprises a question and four possible answers, thus a random choice score should be around 25% which is similar to the evaluation scores of all parameter-modifying ME methods after 100 sequential edits, indicating that the post-edited LLMs fail to answer all questions in the MMLU dataset. All these results highlight the systematic hurt of the parameter-modifying ME methods on LLMs in sequential editing scenarios.

We report the individual and sequential scores of *reliability* and *generalization* in Table 2. The decline of the sequential *reliability* and *generalization* indicates that in sequential editing scenarios, post-edited models, edited by parameter-modifying ME methods, forget previously edited knowledge after several edits. Besides, the individual *reliability* and *generalization* of the ROME and MEMIT methods remain similar as the number of edits increases, while the MEND method has a significant decline, indicating that in sequential editing scenarios, the MEND method cannot successfully add new knowledge into LLMs after several edits.

5.2 Evaluation of Parameter-Preserving ME Method

The parameter-preserving ME method, GRACE, introduces an additional codebook to store edited knowledge. As described in Appendix A.1, it applies a threshold to control whether the input information uses the stored knowledge. In the experiments in Table 1 and Table 2, we set 1 as the value of the threshold. It is shown that such a small threshold helps maintain the broad capabilities of LLMs with no noticeable decline in the performance on all downstream tasks. However, it also restricts the post-edited model from correctly answering the question about the edited knowledge described in a different format. This results in a low score of *generalization*, as illustrated in Table 2. We claim that a larger threshold increases the *generalization* but fails to preserve the core capabilities of LLMs. We discuss the influences of the threshold in Appendix C.

		Sequential Score		*Individual Score*	
Method	Edit #.	**Rel.**	**Gen.**	**Rel.**	**Gen.**
parameter-modifying ME methods					
MEND	1	80	80	80	80
	10	79.3	74.8	86.8	87.7
	20	39.1	44.1	67.2	68.1
	100	0	0	13.6	13.9
ROME	1	80	80	80	80
	10	66.7	69.7	93	87.3
	20	53.3	52.4	90.3	85.7
	100	52.3	49.5	93.3	90.4
MEMIT	1	80	80	80	80
	10	87	87	86.4	83.2
	20	22.4	25.3	88.3	88.1
	100	0.07	0.06	87.7	85.4
parameter-preserving ME methods					
GRACE	100	99.8	30.2	99.8	30.2

Table 2: The individual and sequential scores of *reliability*, denoted as **Rel.** and *generalization*, denoted as **Gen**. We evaluate the scores on the editing dataset.

6 Impact of Different Editing Settings

This section is dedicated to analyzing the influences of ME in different editing settings. We focus on four aspects: model size, instruction tuning, layers to edit, and the batch size of memory editing.

Model Size. Figure 3 illustrates that all model checkpoints edited by the ROME method, regardless of their size, show performance degradation that correlates with the number of sequential edits. Interestingly, an increase in model size appears to have a protective effect, particularly in multilingual understanding and code generation domains, as shown in the TyDiQA and MBPP datasets. However, this protection does not extend to all areas.

The post-edited LLMs with different sizes of 7B and 13B suffer the same decline trend on knowledge question-answering tasks, e.g., the MMLU and CommonsenseQA datasets. We conjecture the reason as: edited knowledge triples and the concerned knowledge in the MMLU and CommonsenseQA datasets are stored closer in the model's parameters, compared to the concerned knowledge of multi-lingual understanding or code generation. As models scale up, a more precise separation between the edited knowledge and concerned knowledge of code generation and multi-lingual understanding tasks emerges, potentially allowing for less disruptive memory editing. We leave the proof of these hypotheses as future work.

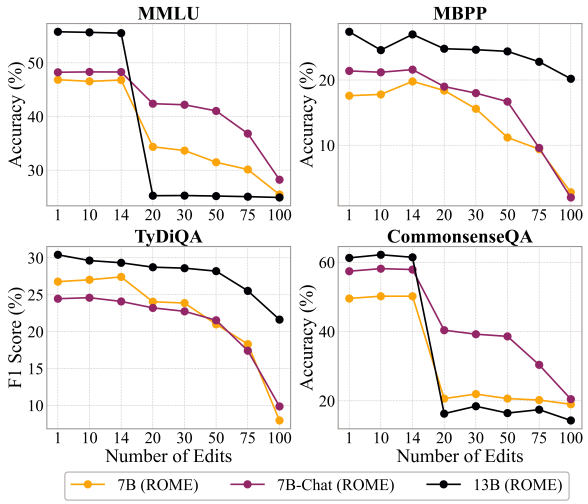

Figure 3: Evaluation of three different checkpoints of LLaMA-2-7B on four datasets. We apply ROME as the ME method.

Instruction Tuning. Compared with `LLaMA-2-7B`, `LLaMA-2-7B-Chat` is further instruction tuned to generate more natural conversational responses. The implementation of instruction tuning, particularly in the `LLaMA-2-7B-Chat` model, provides insightful observations. As shown in Figure 3, despite the overall performance degradation trend, instruction tuning appears to impart a degree of robustness, as evidenced by the enhanced stability across MMLU and CommonsenseQA. This finding suggests that instruction tuning might play a role in safeguarding model capabilities against the detrimental effects of memory editing, especially for knowledge question-answering tasks, although it does not entirely prevent performance losses. However, instruction tuning does not help mitigate the damage to code generation and multi-lingual understanding tasks. We give a preliminary explanation of this phenomenon in Appendix D.2. The impact of instruction tuning on memory editing suggests an intriguing area for further investigation, especially regarding how it influences the model's capability to integrate and handle edited information.

Layers to Edit. Inspired by (Hase et al., 2023), we investigate the effects of editing different layers in LLMs using the ROME and MEMIT methods. Figure 4 shows a noticeable trend: editing layers closer to the output (deeper layers) results in a marginal decrease in performance while editing shallower layers leads to significant performance degradation. Specifically, when editing the 20th layer of the `LLaMA-2-7B` model using ROME, the model's performance on CommonsenseQA after 100 editing iterations stands at 46.27%[1]. However, editing shallower layers, such as the 5th, 10th, and 15th layers, severely impacts the model's performance, leading to significant deterioration after just 20 edits. Similarly, with MEMIT, editing layers 25 through 29 leads to a performance decrease of just 9.6% from the post-first-edit outcomes. These results indicate that the choice of layers for editing in LLMs significantly impacts their general capabilities, with deeper layers showing more resilience to the editing process than shallower ones. We also edit different layers using GRACE, whose results are shown in Appendix D.5, suggesting a similar conclusion as both ROME and MEMIT.

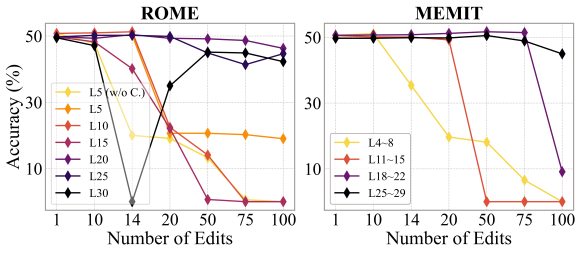

Figure 4: The performance of the LLaMA-2-7B model on the CommonsenseQA dataset. LX represents editing the X-th layer of the model, while LX~Y represents editing layers between the X-th and the Y-th layer.

Batch Size of ME. In line with Meng et al. (2022b), we conduct experiments to test the influence of varying batch sizes of memory editing. Utilizing MEMIT to edit `LLaMA-2-7B`, we alter the batch size from 1 to 1000. As shown in Figure 5,

[1]An intriguing observation emerges when we edit the 30th layer using ROME, which is explained in Appendix D.3.

with the same number of edit triples, increasing the batch size means reducing the number of editing times, which turns out to be beneficial in mitigating the damage of ME to LLMs.

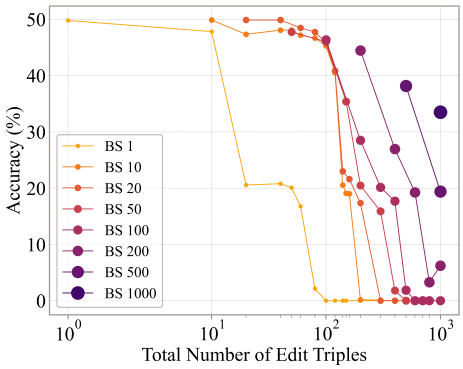

Figure 5: The performance of LLaMA-2-7B on CommonsenseQA, utilizing MEMIT as the editing method with different batch sizes for memory editing. The x-axis denotes the total number of edit triples. For example, for the line of batch size 100, the first data point of this line lies in the total number of edit triples 100, which only edits the model once. BS denotes batch size.

Temperatures. In this experiment, we explore how varying temperatures can affect the performance of post-edited LLMs. Specifically, we apply the ROME method to edit the 25th layer of LLaMA-2-7B and subsequently evaluate its performance on CommonsenseQA under different temperature settings: 0, 0.2, 0.5, and 0.8. Our findings as shown in Table 3, indicate a clear trend: as the inference temperature increases, the edited model's performance deteriorates more rapidly. At a temperature of 0, the model maintains a stable performance until a significant number of edits are made, after which the performance sharply declines to 0. However, at higher temperatures (0.2, 0.5, and 0.8), the performance starts to decrease more noticeably, even with fewer edits. This result suggests that higher inference temperatures, which typically encourage more diverse and less certain outputs, may exacerbate the model's vulnerability to memory editing, leading to more pronounced performance degradation.

	Number of Edits							
Temp.	0	1	10	50	100	200	500	1000
0	49.6	49.6	50.2	44.8	44.6	38.8	1.2	0
0.2	49.6	49.6	50.1	45.1	47.8	36.7	0	0
0.5	49.6	49.6	44.6	43.7	46.6	32.4	0	0
0.8	49.6	49.6	42.1	40.2	42.3	31.2	0	0

Table 3: The evaluation results on CommonsenseQA across different temperatures.

7 Interpreting Disruptions in LLMs Caused by Memory Editing

To interpret the damage caused by parameter-modifying ME methods, our investigation is structured around three pivotal aspects: (i) the model parameter changes after being sequential edited, (ii) the impact on the language modeling capability of LLMs, and (iii) the in-context learning capacity. This multifaceted exploration is designed to provide a holistic understanding of how memory editing affects LLMs.

7.1 Parameter Changes after Memory Editing

In this section, we investigate the changes between the parameters of LLMs before and after sequential memory editing. We apply ROME as the ME method and `LLaMA-2-7B` as the base model. We use the Pearson product-moment correlation coefficient (R) to measure the similarities between the parameters of the original and edited layers within the model. The correlation coefficient matrix, R, ranges from -1 to 1. An R value of 1 indicates a perfect positive linear correlation, implying that the parameters in both the original and edited layers are identical. Conversely, a value of -1 indicates a perfect negative correlation, while a value of 0 suggests no similarity between the parameters."

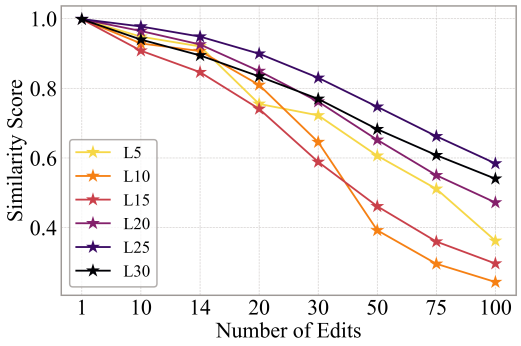

Figure 6: Similarity score based on the Pearson product-moment correlation coefficient, calculated between the parameters of the original and edited model layers.

As illustrated in Figure 6, with fewer than 15 edits, the correlation coefficient (R) between the edited and original layers remains high (e.g. close to 1), indicating the significant similarity of the parameters. However, with an increasing number of edits, there is a marked decrease in similarity. Such changes in the parameter lead to a "mismatch" between the edited and original layers, which undermines the model's inherent coherence through layers. Consequently, the model's general capabilities are significantly damaged.

	Number of Edits							
Edit Layer	1	10	14	20	30	50	75	100
5	7.63	7.65	7.61	14.29	14.29	13.89	12.90	14.04
10	7.15	7.32	7.38	28.61	45.23	81.08	/	/
15	7.61	7.48	81.24	50.09	21.25	28.48	29 634.93	17 220.91
20	7.61	7.75	7.69	8.12	8.09	9.48	10.67	11.15
25	7.63	7.61	7.73	8.81	15.77	18.35	31.26	9830.27
30	7.65	810.04	2477.53	603.46	49.09	78.39	1018.46	1444.29

Table 4: Perplexity scores computed by pre-trained Vicuna-7b-v1.5. The calculated texts were generated by the edited LLaMA-2-7B. The result "/" means that the edited model fails to generate any response.

One interesting finding is that modifications in deeper layers, especially the 20th, 25th, and 30th layers, maintain relatively higher similarity scores compared to editing the shallower layers. This finding aligns with the experiments of editing different layers in Section 6, where we find that editing deeper layers results in a less pronounced decrease in performance. This distinction highlights a key architectural characteristic of LLMs: deeper layers, located closer to the output, exhibit greater tolerance to modifications, effectively sustaining the model's performance. On the other hand, the shallower layers, forming the foundational processing stages of the LLMs, are more susceptible to disruptions from edits, leading to more significant performance degradations. This layered sensitivity within LLMs underscores the importance of strategic layer selection in the editing process.

We argue that the diminishing similarity between the edited and original layers is a primary factor in the model's reduced performance, disrupting its internal coherence and substantially impacting its performance in various tasks.

7.2 Language Modeling Capability

We hypothesize that the significant changes in the edited layers damage the language modeling capability of LLMs. To validate this hypothesis, we use `Vicuna-7b-v1.5` (Zheng et al., 2023b) to measure the *Perplexity* (PPL) of output sequences generated by post-edited models edited by ROME. CommonsenseQA is used as the evaluation dataset.

In our setting, we concatenate each question with its corresponding generated answer and calculate the *perplexity* solely for the first 20 tokens of the answer portion. Answers with less than 20 tokens are excluded to avoid the effect of sequence length on the PPL. Additionally, we observe that in certain instances, the post-edited models tend to produce repetitive token sequences, which, while contributing to lower perplexity scores, are not meaningful in the context of answering CommonsenseQA questions. To address this, we implement a penalty ratio for repetitive sentences to ensure a more accurate reflection of the model's language modeling capability. The details of the formula to calculate the adjusted perplexity are shown in Appendix E.1.1.

As illustrated in Table 4, after 100 sequential edits, editing the 10th and 15th layers results in an extremely high perplexity, which leads to a zero score for the performance. On the other hand, editing the 5th layer results in a relatively low perplexity, indicating that the model is not completely damaged, although there is a significant decline in the performance as shown in Table 1. Editing the 20th layer maintains a lower perplexity, which guarantees a high performance on CommonsenseQA. These findings can explain the observations in Figure 4. However, although editing the 25th and 30th layers severely damages the language modeling capability of LLMs, they still maintain very high performance on CommonsenseQA, as shown in Figure 4. We explain this by examining the in-context learning capability in Section 7.3.

7.3 In-Context Learning Capability

We further investigate whether, after memory editing, LLMs can still maintain the in-context learning capabilities. Wang et al. (2023) demonstrates that in in-context learning, the shallow layers of LLMs aggregate information from contexts into label words (for example, the CommonsenseQA contains five options as label words - A, B, C, D, or E), while in deep layers, LLMs extract and use the aggregated information of label words to perform the final prediction. Inspired by this work, we evaluate the post-edited LLM on SST2 (Socher et al., 2013) where the label words are "positive" and "negative", based on 1-shot in-context learning. We use `LLaMA-2-7B` as the base LLM and edit it using

ROME and MEMIT on different layers. To save space, we describe the experimental setup and detailed results in Appendix E.2. The experimental results indicate that editing shallow (e.g. the 5th layer) and deep layers (e.g. 20th, 25th, and 30th layers) does not significantly influence the in-context learning capability of LLMs.

These findings also explain the phenomenon mentioned in Section 7.2 – although editing the 25th and 30th layers severely damages the language modeling capability of LLMs, they still maintain very high performance on CommonsenseQA as shown in Figure 4. The experiments illustrated in Figure 4 on CommonsenseQA are based on an 8-shot in-context setting, and the first token of the generated sequence is treated as the final prediction. Given the maintenance of in-context learning capability, the post-edited model is still able to correctly predict the first token of the generated sequence, although it fails to generate a meaningful sentence because of the damage to language modeling capability.

8 Conclusions

We conduct a comprehensive evaluation of two types of memory editing methods for LLMs across eight diverse benchmarks. Our findings indicate that parameter-modifying ME methods tend to systematically degrade the model performance on general downstream tasks. In contrast, the parameter-preserving ME method, GRACE, successfully maintains the LLMs' capabilities but fails to maintain *generalization*. We also show that increasing model size, instruction tuning, editing deeper layers, and increasing the batch size of memory editing are beneficial to mitigate the damage of parameter-modifying ME methods to LLMs. Finally, we conduct an in-depth analysis of how parameter-modifying ME methods hurt the general capabilities of LLMs. Overall, our research provides comprehensive insights into the dynamics of how, when, and why memory editing influences LLMs, offering valuable guidance for future research on memory editing.

9 Limitations

Despite the contributions, our study still has limitations. Our experiments on parameter-preserving ME methods are not exhaustive. As shown in Figure 4, there is an observed performance decrease after 100 edits when editing layers 20/25 with ROME. Further experiments are needed to understand these long-term effects. Besides, we do not completely explain why LLMs can maintain in-context learning capabilities after being sequentially edited. These limitations highlight areas for future research, underscoring the need for more extensive investigations to refine our understanding of the intricate balance between knowledge editing and model integrity in LLMs.

Acknowledgments

This research is based upon work partially supported by the award No. 2238940 from the Faculty Early Career Development Program (CAREER) of the National Science Foundation (NSF) and the U.S. DARPA FoundSci Program #HR00112490370. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. Tydi qa: A benchmark for information-seeking question answering in ty pologically di verse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

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*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

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

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. *arXiv preprint arXiv:2210.03329*.

Ahmad Ghazal, Tilmann Rabl, Minqing Hu, Francois Raab, Meikel Poess, Alain Crolotte, and Hans-Arno Jacobsen. 2013. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pages 1197–1208.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors. *arXiv preprint arXiv:2211.11031*.

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*.

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*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

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*.

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*.

Yiming Ju and Zheng Zhang. 2023. Klob: a benchmark for assessing knowledge locating methods in language models. *arXiv preprint arXiv:2309.16535*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

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

Zichao Li, Ines Arous, Siva Reddy, and Jackie Chi Kit Cheung. 2023b. Evaluating dependencies in fact editing for language models: Specificity and implication awareness. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7623–7636.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*.

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

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

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

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

Yuval Pinter and Michael Elhadad. 2023. Emptying the ocean with a spoon: Should we edit models? *arXiv preprint arXiv:2310.11958*.

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.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. 2020. Investigating prior knowledge for challenging chinese machine reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:141–155.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro,

Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

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*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023a. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A Editing Methods

We conduct our experiments on four ME methods. The summary of each method is shown in Table 5. We introduce GRACE and ROME in detail in the following sections. The MEMIT method is not introduced as it is an improved version of ROME.

A.1 GRACE

GRACE (Hartvigsen et al., 2022) is a method designed for sequential memory editing without altering original model parameters. The GRACE adapter, which is wrapped into a chosen layer of an LLM, contains two components: (1) a codebook that consists of a set of keys, denoted as $\mathbb{K}$, and values, denoted as $\mathbb{V}$, and (2) deferral radii, denoted as $\mathcal{E}$, to decide whether the input information flow uses the codebook. Specifically, $\mathbb{K}$ is a set of cached activation h^{l-1} predicted by layer $l-1$. $\mathbb{V}$ is a set of values that are randomly initialized and updated using the LLMs' loss for edits. Each key is corresponding to a single value. The hyperparameter $\epsilon \in \mathcal{E}$ is a threshold for the similarity between the new input and previous edited knowledge. GRACE adapter is activated at layer l only if this similarity is smaller than the radius.

During editing, GRACE adds keys, corresponding values, and ϵ entries. In the inference process, at layer l, if the similarity of the activation at layer $l-1$ and keys are smaller than the corresponding radius ϵ, the activation of the next layer becomes the cached corresponding values. Formally, the activation of lth layer is formulated as follows:

$$h^l = \begin{cases} \text{GRACE}\left(h^{l-1}\right) & \text{if } \min_i \left(d\left(h^{l-1}, \mathbb{K}_i\right)\right) < \epsilon_{i_*}, \\ f^l\left(h^{l-1}\right) & \text{otherwise} \end{cases} \tag{1}$$

where $i_* = \text{argmin}_i \left(d\left(h^{l-1}\right), \mathbb{K}_i\right)$ and $f^l\left(h^{l-1}\right)$ denotes the l-th layer's activation of the unedited model. ϵ_i and $\mathbb{K}_i$ are the deferral radius and key i. GRACE(h^{l-1}) retrieves the corresponding value associated with the closest key. $d(.)$ is a distance function. Following Hartvigsen et al. (2022), we use Euclidean distance in our experiments.

As shown in our experiments in Section 5.2, the hyperparameter ϵ is a trade-off between *generalization* and maintaining the broader fundamental capabilities of LLMs.

A.2 ROME and MEMIT

ROME (Meng et al., 2022a) applies a Locate-then-Edit strategy, which first utilizes the causal tracing

	Method	Additional Training	Edit Layer	Default Edit Parameter
Preserving Parameters	GRACE	NO	FFN	$30^{th} mlp_{proj}$
Modifying Parameters	MEND	YES	FFN	$Model_{hyper} + 29/30/31^{th} mlp$
	ROME	NO	FFN	$5^{th}\ mlp_{proj}$
	MEMIT	NO	FFN	$4/5/6/7/8^{th}\ mlp_{proj}$

Table 5: The details of memory editing methods. The edit parameter is in default for all checkpoints. We also conduct the ablation study on edited layers where we specify the exact layers we edit. In the table, mlp_{proj} means the down project layer of the MLP layer, while mlp means we edit the gate/up/down project layers of the MLP layer.

method to ensure that MLP layers in LLMs play a role in recalling factual knowledge, and then edits specific MLP layers to integrate new knowledge into LLMs. Following Meng et al. (2022a), we denote the first layer of the lth MLP layer as $W_{fc}^{(l)}$, and the second layer as $W_{proj}^{(l)}$. ROME treats $W_{proj}^{(l)}$ as a linear associative memory, which claims that any linear operation W can work as a key-value store for a set of Key-Value vectors denoted as $K = [k_1 | k_2 | \ldots]$ and $V = [v_1 | v_2 | \ldots]$, respectively. A new key-value pair (k_*, v_*) can be injected into W by solving the following equation:

$$\text{minimize } \|\hat{W}K - V\| \text{ such that } \hat{W}k_* = v_*. \quad (2)$$

This can be solved by setting $\hat{W} = W + \Lambda \left(C^{-1}k_*\right)^T$, where W is the original matrix, $C = KK^T$ is a pre-cached constant, and $\Lambda = (v_* - Wk_*) / \left(C^{-1}k_*\right)^T k_*$. In ROME's work, C works as a constraint method to avoid edited parameters forgetting other unrelated knowledge. It is computed using the hidden states k of 100,000 random samples from Wikipedia text. We evaluate whether the constraint method is beneficial to mitigating the damage of ROME to the general capabilities of LLMs in the Appendix D.1.

MEMIT (Meng et al., 2022b), which can edit multiple knowledge at a time (e.g. batch editing), is a following work of the ROME (Meng et al., 2022a).

B Evaluation Datasets

To rigorously assess the impact of ME methods on LLMs, we employ a diverse set of benchmarks encompassing essential capabilities, including Professional Knowledge, Common Sense Knowledge, Logical Reasoning, Reading Understanding, and Multilingual Proficiency. Our evaluation consists of eight benchmarks, the specifics of which are delineated in Table 6. We leverage the *opencompass* codebase (Contributors, 2023), a widely recognized open-source repository for LLMs evaluation. In alignment with their established protocols, we adopt the Perplexity (PPL) mode for the evaluation of the MMLU dataset. For instance, in the MMLU dataset, each item comprises a question and four possible answers. We concatenate the question with each answer option to create four distinct input sequences. Subsequently, we compute the Perplexity for each sequence using the edited LLMs under examination. A lower Perplexity score indicates higher model confidence in the corresponding sentence, thereby guiding our selection of the answer with the lowest score as the definitive prediction. Conversely, for the remaining benchmarks, we utilize the Generation (GEN) mode for evaluation. Specifically, for MATH, BBH, and TyDiQA, we ascertain the accuracy of the model's predictions against the ground truth following a post-processing procedure. Regarding the programming task MBPP, we employ Python's built-in *exec()* function to verify the error-free execution of the generated code.

C The Trade-off of the Threshold in GRACE

As shown in Table 7, the *generalization* increases rapidly when we increase the threshold from 1 to 20. However, the capabilities of multi-lingual understanding and code generation are completely damaged. One counter-intuitive finding is that the performance of the MMLU is not hugely influenced. We leave the explanation of this phenomenon as future work.

D Additional Impact of Different Editing Settings

D.1 Efficacy of Constraint Method in ROME

In our examination of ROME's constraint methodologies, which incorporate 100,000 Wikidata entries to limit the influence of edits on unrelated

Capability	Task	Datasets	#. Items	Metrics	Language	Mode	#. Shots
Professional Knowledge	High School / University Professional Examination	MMLU	15691	Acc.	English	PPL	5
Logical Reasoning	Mathematical Reasoning	MATH	5000	Acc.	English	GEN	4
	Comprehensive Reasoning	BBH	6511	Acc.	English	GEN	3
	Textual Entailment	AX-b	1104	Acc.	English	GEN	0
Common Sense Knowledge	Knowledge Question Answering	ComQA	1221	Acc.	English	GEN	8
Reading Understanding	Reading Understanding	C3	1825	Acc.	Chinese	GEN	0
Multilingual Proficiency	Multi-Language Question Answering	TyDiQA	6322	F1	13 languages	GEN	0
Code Generation	Code Generation	MBPP	500	Pass.	Code	GEN	3

Table 6: The details of downstream evaluation benchmarks.

information, we analyze a variant of ROME without constraints (ROME w/o C). Figure 3 illustrates that applying constraints significantly enhances the model's performance in all datasets, validating the effectiveness of this strategy. In the absence of constraints, a marked deterioration in performance is observed, notably in benchmarks like TyDiQA, CommonsenseQA, and MBPP. This finding indicates that unconstrained parameter modification can severely impair the model's efficacy, while the application of constraints attenuates this negative impact. However, it's noteworthy that the effectiveness of these constraints begins to wane after approximately 20 edits. This observation highlights an emerging need for innovative constraint methodologies in parameter modification, particularly in the context of sequential memory editing. Developing more robust constraint mechanisms could be vital to maintaining model performance and integrity over a broader range of edits.

D.2 Explanation of Instruction Tuning

To preliminary explain why instruction tuning help to safeguard model capabilities against the negative effects of sequential editing, we conduct the following experiments on LLaMA-2-7B and LLaMA-2-7B-Chat. Both of them are sequentially edited 100 times at the 5th layer, which is consistent with the experimental setting in Figure 3. As described in Section 7.1, parameter changes are one of the possible reasons for the performance decline in downstream tasks. Therefore, we compare the parameter changes of the two edited models. We report the similarity score of edited parameters and original parameters in Table 8. It is shown that the parameter changes of the LLaMA-2-7B-Chat model are always smaller than that of the LLaMA-2-7B model, which indicates the potential for less damage on the instruction tuning model compared to the original model. Besides, as mentioned in Section 6, CommonsenseQA and MMLU are highly different from MBPP (a code generation task), and TyDiQA (a multi-lingual understanding task). The instruction-tuned model is mainly tuned for dialogue in English, which is largely different from code generation and multi-lingual texts but relatively similar to the English knowledge questions, which might be a potential reason for the robustness of the instruction-tuned model on CommonsenseQA and MMLU after being edited.

D.3 Explanation of Editing Deeper Layers

In this section, we explain the phenomenon when we edit the 30th layer of LLaMA-2-7B using ROME. As shown in Figure 4, after 14 edits to the 30th layer, the model's performance intriguingly plummeted to zero. However, a notable recovery occurred after 20 edits, with performance gradually increasing to approximately 45% following 50 edits. This unusual pattern can be attributed to the methodology used in our evaluation, where we considered the first token of the output generated by the edited model as the final prediction. Initially, after 14 edits, the model's language modeling capability appeared to be completely compromised. Yet, after 20 edits, the model consistently predicted the first token as one of the candidates - 'A, B, C, D, or E' - although it still failed to generate a coherent sequence beyond this. This indicates that while the model retained the capacity to predict the first token accurately,

Layer	ϵ	C3	ComQA	MBPP	AX-b	MMLU	Rel.	Gen.
20	1	27.07	44.1	15.8	46.56	46.8	99	30.2
	5	27.07	44.1	15.2	45.92	46.8	98	45.0
	10	27.01	39.72	15.2	41.58	46.8	99	52.1
	20	10.36	12.19	0	14.49	46.2	98	97.3
30	1	32.1	49.6	18.2	45.9	46.8	99	27.2
	5	32.1	49.6	18.2	45.9	46.8	99	28.1
	10	28.55	46.36	17.4	45.02	46.8	99	41.5
	20	24.25	42.06	16.4	43.97	46.8	99	51.8

Table 7: The evaluation results across different thresholds of GRACE. We edit the 20th and 30th layers in this experiment, while in Table 1 where we only edit the 30th layer. We denote ϵ as the threshold. Rel. and Gen. are *reliability* and *generalization* respectively, which is evaluated on the editing dataset.

	Number of Edits					
Model	1	10	20	50	75	100
7B	0.997	0.946	0.753	0.605	0.501	0.348
7B-Chat	0.999	0.949	0.775	0.625	0.525	0.361

Table 8: The similarity scores between edited models and original models. 7B refers to LLaMA-2-7B and 7B-Chat refers to LLaMA-2-7B-Chat respectivately.

	Number of Edits			
Edit Layer	100	200	500	1000
20	46.2	25.7	12.8	0
25	44.6	38.8	1.15	0
30	42.3	47.0	23.1	0

Table 9: The evaluation results on CommonsenseQA across different editing layers of ROME.

its broader language modeling capabilities were significantly diminished. We delve into a more in-depth analysis and explanation of this phenomenon in Section 7, exploring this observation's underlying mechanisms and implications.

D.4 Long-term Effects of ROME Method

We extend our experiments in Figure 4 by sequentially editing the 20th, 25th, and 30th layers of LLaMA-2-7B 1000 times, and evaluating the edited model on CommonsenseQA. As is shown in Table 9, after 500 edits, the model's performance drastically declines, with almost complete degradation observed after 1000 edits. Specifically, for layer 20, the performance drops from 46.2% after 100 edits to 0 after 1000 edits. Similar trends are noted for the 25th and 30th layers, with performances plummeting to zero after extensive editing. These results illustrate the long-term effects of sequential memory editing on LLMs, revealing a critical threshold beyond which the model fails to maintain its capabilities, essentially 'entirely forgetting' its knowledge. This degradation not only confirms the substantial impact of extensive ME but also highlights the necessity of developing more sustainable editing approaches that can preserve the model's integrity over time.

D.5 Layers to Edit in GRACE Method

We also conduct experiments to edit different layers of LLaMA-2-7B using the GRACE method. According to Table 10, with the same threshold, editing the shallower layer results in more damage to LLMs. This is because, in the shallow layer, the activations are not much different for different inputs because of the less calculation compared to deeper layers. We claim that editing deeper layers in GRACE is a better choice than that of shallower layers.

Layer	MMLU	ComQA	TyDiQA	MBPP
10	23.1	8.7	0.1	0
20	46.8	39.7	22.8	16.4
30	46.8	46.4	23.42	17.4

Table 10: The evaluation results across different editing layers of GRACE. The threshold is set by 10.

D.6 Different Editing Datasets

To ensure the robustness of our findings, we conduct further experiments using four different random seeds to select distinct sets of 100 samples for editing. These experiments are designed to assess the variability in the impact of memory editing across different subsets of the data. In these extended experiments, we apply the ROME method to sequentially edit the 5th layer of the LLaMA-2-7B model, with the evaluation conducted on Common-

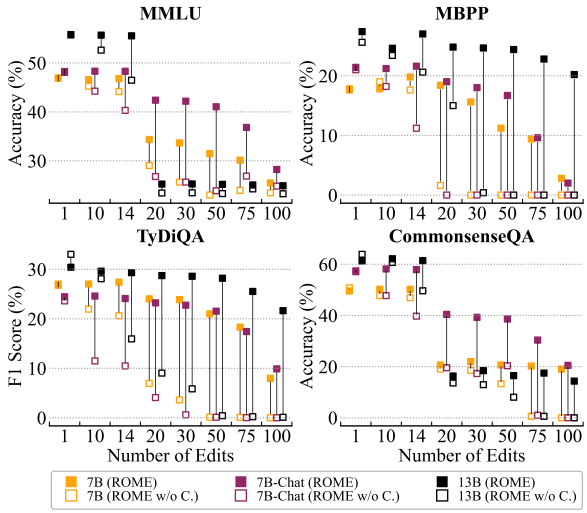

Figure 7: Evaluation Performance across three different checkpoints of LLaMA-2-7B. We denote the ROME method without constraint strategy using 100,000 Wikipedia text as ROME w/o C.

senseQA. As is shown in Table 11, despite the sample variance, there is a consistent trend of performance degradation post-editing. Specifically, after 100 sequential edits, all versions of the edited LLM exhibit a significant performance decline, indicating a general trend of damage across different sample sets. However, it was also observed that the rate of performance degradation varied among the samples within the first 20 edits, suggesting that some samples might induce faster degradation than others.

	Number of Edits				
Sample Seed	1	10	20	50	100
a	49.6	49.1	20.6	20.1	19.0
b	49.6	49.2	48.5	31.6	22.1
c	49.6	49.1	46.7	25.4	21.2
d	49.6	49.1	24.7	20.2	19.5

Table 11: The evaluation results on CommonsenseQA across different editing samples.

E Additional Analysis of the Damage to LLMs by ME Methods

E.1 The Language Modeling Capability

E.1.1 Adjusted Perplexity

As described in Section 7.2, we proposed adjusted perplexity as a measurement for the language modeling capability of post-edited LLMs to avoid the influence of the generated repetitive sequences. We employ Vicuna-7b-v1.5 (Zheng et al., 2023b) to measure the Perplexity of the output sequences generated by post-edited models edited by ROME to answer questions in the CommonsenseQA dataset. Specifically, denote a generated sequence with n tokens as $Y = (y_1, y_2, ..., y_n)$, we calculate the perplexity using the following equation:

$$\text{PPL}(Y) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta\left(y_i \mid y_{<i}\right)\right\} \quad (3)$$

where $p_\theta(y_i|y_{<i})$ is the log-likelihood of the ith token conditioned on the previous tokens $y_{<i}$. However, such a naive approach is not applicable in our situation because post-edited models tend to generate repetitive tokens, which leads to relatively low perplexity. Therefore, we calculate the n-gram repetitive ratio for each sequence. We first slice the sequence into several n-gram fragments, then we set the ratio of the number of unique fragments over the total number of fragments as the repetitive ratio ρ. Finally, we calculate the adjusted PPL is calculated by:

$$\text{Adj\_PPL}(Y) = \text{PPL}(Y) \times e^{1-\rho} \quad (4)$$

E.1.2 Additional Evaluation on the Language Modeling Capability

In addition to applying pre-trained LLM to calculate the perplexity of sequences generated by the edited model, as described in Section 7.2 and E.1.1, we also use the edited model to calculate the perplexity of normal texts as another evaluation metrics of the language modeling capability. Specifically, we randomly select 1000 sequences from the WikiText-103 dataset (Merity et al., 2016) and feed them into edited LLaMA-2-7B to calculate the perplexity scores. As is shown in Table 12, editing shallow layers (especially for layers 10 and 15) damages the model rapidly and severely. However, the model retains more language modeling capabilities when edited in deeper layers. This result is consistent with the Table 4.

E.2 The In-Context Learning Capability

In-context learning, which concatenates several demonstration-label pairs and the demonstration to be predicted as input context, is one of the most important capabilities of LLMs. Wang et al. (2023) explain the success of LLMs in in-context learning, that in the shallow layers (near to input), the model aggregates information from demonstrations to label words, while in deep layers, the model

	Number of Edits					
Edit Layer	0	1	10	14	50	100
5	10.6	12.71	12.72	13.01	13.23	18.31
10	10.6	18.33	18.99	50.15	417.64	1593.14
15	10.6	12.71	12.80	19.68	23.5	18 373.90
20	10.6	18.75	18.98	19.37	20.14	29.30
25	10.6	12.70	12.75	12.96	20.62	35.55
30	10.6	15.56	30.25	60.24	40.99	53.12

Table 12: Perplexity scores of standard texts, calculated by the edited LLaMA-2-7B, when different layers are sequentially edited.

extracts and uses this information from previous label words to form the final prediction. In this section, we utilize the same way proposed by Wang et al. (2023) to analyze whether the in-context learning capability has been influenced after sequential edits. Specifically, we calculate the saliency score (Simonyan et al., 2013) for each attention matrix:

$$I_l = \left| \sum_h A_{h,l} \odot \frac{\partial \mathcal{L}(x)}{\partial A_{h,l}} \right| \tag{5}$$

where $\mathcal{L}(x)$ is the loss function of the task, $A_{h,l}$ represents the value of attention matrix of the h-th attention head in the l-layer and x represents the input. $I_l(i,j)$ is the significance of the information flow from the i-th token to j-th token. We denote p_i as the i-th label words such as "True" or "False", q as the target position in which the model predicts labels, and w as the words in demonstrations. C represents the number of label words. We have three metrics as shown below:

S_{wp}: the saliency score of information flow from text part w to label words p:

$$\begin{aligned} S_{wp} &= \frac{\sum_{(i,j)\in C_{wp}} I_l(i,j)}{|C_{wp}|}, \\ C_{wp} &= \{(p_k, j) : k \in [1, C], j < p_k\}. \end{aligned} \tag{6}$$

S_{pq}: the saliency score of information flow from label words p to target position q:

$$\begin{aligned} S_{pq} &= \frac{\sum_{(i,j)\in C_{pq}} I_l(i,j)}{|C_{pq}|}, \\ C_{pq} &= \{(q, p_k) : k \in [1, C]\}. \end{aligned} \tag{7}$$

S_{pq}: the saliency of information flow except S_{wp} and S_{pq}:

$$\begin{aligned} S_{ww} &= \frac{\sum_{(i,j)\in C_{ww}} I_l(i,j)}{|C_{ww}|} \\ C_{ww} &= \{(i,j) : j < i\} - C_{wp} - C_{pq} \end{aligned} \tag{8}$$

We utilize SST-2 (Socher et al., 2013) as the experimental datasets and one-shot setting. According to Figure 8, the original Llama-2-7B model proves the claim proposed by Wang et al. (2023). Specifically, in the shallow layer (from layer 0 to layer 5), the line of S_{wp} dominates, which shows that the information is aggregating from text to labels. While in the deep layer (from layer 6 to the last layer), the line of S_{pq} dominates, indicating that the label information is aggregating to the target position. For the ROME method, editing layer 5 has a slight influence on layers 6 to 10, which promotes the information aggregating to label words process. Because the change is not very obvious, the model can still maintain an average score of 18.3% accuracy according to Table 1. While if we edit layer 15, due to the damage stored in layer 15, in the deeper layer, there are some fluctuate between S_{wp} and S_{pq}, which shows unstable attention across those layers, resulting in much worse performance on CommonsenseQA as shown in Figure 4. The same thing happens when we edit layers from 4th to 8th using the MEMIT method. It is shown that in the deeper layer, the information fails to aggregate form label words to target position, which explains a worse average score of 3.8% according to Table 1. Finally, editing the 30th layer does not have much influence on such an attention mechanism for information flow. This means that the perplexity capability is much different from the in-context learning capability. Besides, this also partly explains why editing the 30th layer using ROME gives a high performance after 100 edits.

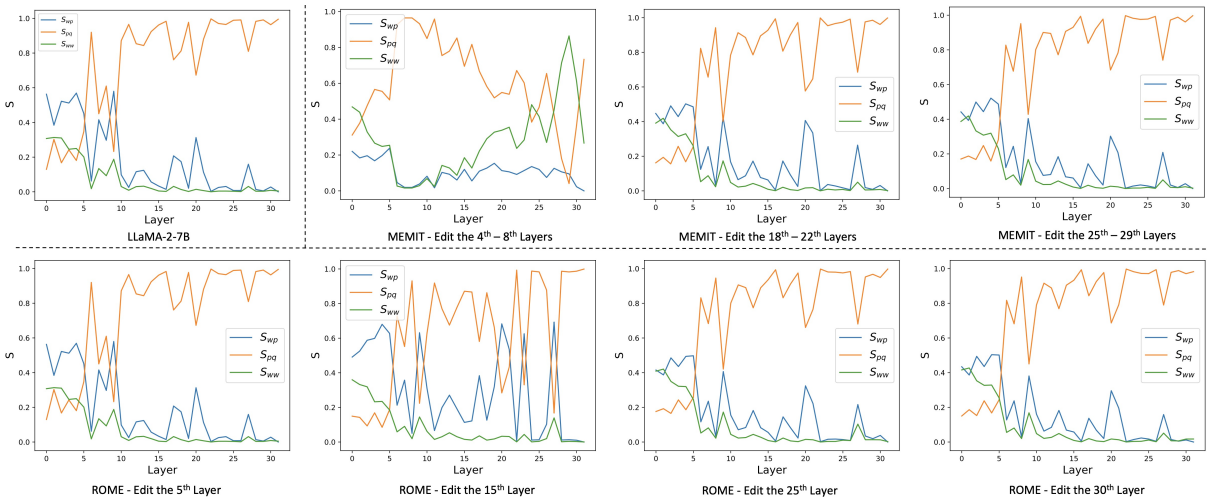

Figure 8: In-context learning saliency score

F Experiments on Other Models

In order to prove the robustness of our findings, we conduct similar experiments on two different models: Mistral-7B (Jiang et al., 2023) and GPT2-XL (Radford et al., 2019).

F.1 Mistral-7B

We apply ROME to sequentially edit 1000 knowledge on Mistral-7B and evaluate the edited model on CommonsenseQA. As is shown in Figure 9, sequential memory editing completely damages LLM after around 200 edits.

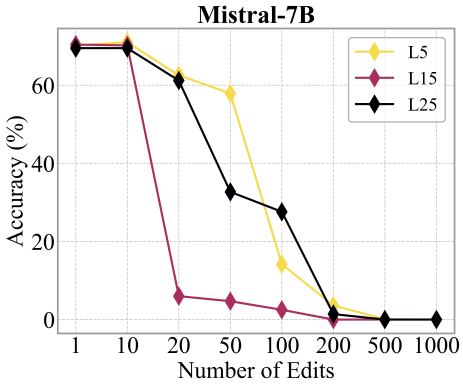

Figure 9: The performance of edited Mistral-7B on CommonsenseQA, utilizing ROME as the editing method with different editing layers.

F.2 GPT2-XL

We further conduct the same experiments on GPT2-XL. We use ROME to edit 1000 knowledge and evaluate the edited model on CommonsenseQA and WikiText. As is shown in Figure 10 and Table 13, after sequentially editing GPT2-XL, although the edited model can still successfully answer some questions of CommonsenseQA, it fails to maintain language modeling capabilities. One possible explanation is that GPT-2-XL maintains high in-context learning capabilities after sequential editing. As explained in Section 7.3, maintaining the in-context learning capability is helpful for the tasks CommonsenseQA. To prove this, following Appendix E.2, we calculate the score of S_{wp}, S_{qp} and S_{ww}. As is shown in Table 14, after 1000 edits, the GPT2-XL still maintains the in-context learning capability, which explains the reason why it maintains similar results on CommonsenseQA after sequential editing.

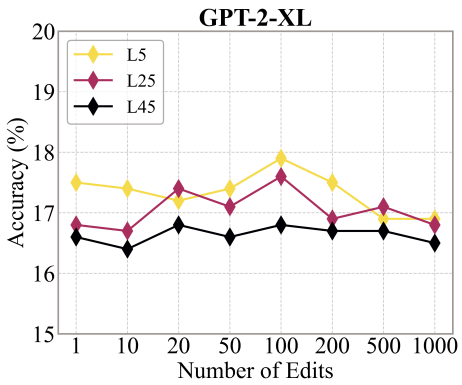

Figure 10: The performance of edited GPT2-XL on CommonsenseQA, utilizing ROME as the editing method with different editing layers.

	Number of Edits						
Layer	0	1	10	50	100	500	1000
5	41.1	41.2	41.4	41.9	470.9	910.7	1250.3
25	41.1	100.7	472.1	476.3	900.2	1025.7	1798.6
45	41.1	125.6	127.3	370.6	490.2	881.1	1697.9

Table 13: Perplexity scores of standard texts, calculated by the edited GPT2-XL, when different layers are sequentially edited.

	Layer									
Score	0	5	10	15	20	25	30	35	40	45
S_{wp}	0.68	0.61	0.3	0.17	0.13	0.01	0.02	0.01	0.01	0.01
S_{qp}	0.16	0.24	0.6	0.75	0.8	0.94	0.95	0.98	0.98	0.98
S_{ww}	0.16	0.15	0.1	0.08	0.07	0.05	0.03	0.01	0.01	0.01

Table 14: The in-context learning salience score of each layer of the edited GPT2-XL.