Constraining Sequential Model Editing with Editing Anchor Compression

Hao-Xiang Xu[1*], **Jun-Yu Ma**[1*], **Zhen-Hua Ling**[1†], **Ningyu Zhang**[2], **Jia-Chen Gu**[3]
[1]National Engineering Research Center of Speech and Language Information Processing,
University of Science and Technology of China, Hefei, China
[2]Zhejiang University
[3]University of California, Los Angeles
{nh2001620,mjy1999}@mail.ustc.edu.cn, zhling@ustc.edu.cn,
zhangningyu@zju.edu.cn, gujc@ucla.edu

Abstract

Large language models (LLMs) struggle with hallucinations due to false or outdated knowledge. Given the high resource demands of retraining these models, there is an increasing focus on developing *model editing*. However, the general abilities of LLMs across downstream tasks are prone to significant degradation during sequential editing. This paper statistically observes that the parameter matrix after editing exhibits a significant deviation compared to its previous state as the number of edits increases. This serious deviation affects the original knowledge associations within LLMs and leads to the degradation of their general abilities. To this end, a framework termed **E**diting **A**nchor **C**ompression (EAC) is proposed to constrain the deviation of the parameter matrix during sequential editing. It compresses the editing information by selecting editing anchors that are important in encoding new relations without deviating too much from the original matrix, thereby preserving the general abilities. Experiments of applying EAC to two popular editing methods on three LLMs across four tasks are conducted. Evaluation results show that EAC effectively minimizes unreasonable deviations caused by model editing, preserving over 70% of the general abilities while better retaining the editing knowledge compared to the original counterpart methods[1].

1 Introduction

Despite the remarkable capabilities of large language models (LLMs) (Qin et al., 2023; Touvron et al., 2023), they often inevitably exhibit hallucinations due to incorrect or outdated knowledge embedded in their parameters (Zhang et al., 2023; Peng et al., 2023; Ji et al., 2023). Given the significant time and expense required to retrain

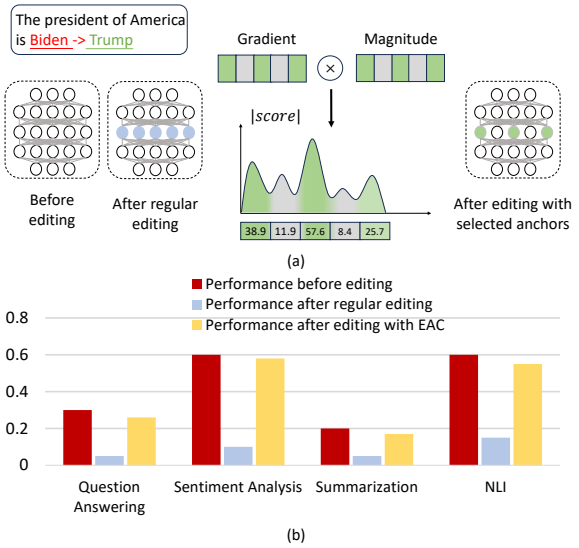


Figure 1: (a) Comparison of regular model editing and EAC. EAC compresses the editing information into the dimensions where the editing anchors are located. Here, we utilize the gradients generated during training and the magnitude of the updated knowledge vector to identify anchors. (b) Comparison of general downstream task performance before editing, after regular editing, and after constrained editing by EAC.

LLMs, there has been growing interest in *model editing* (a.k.a., *knowledge editing*) (Sinitsin et al., 2020; Zhu et al., 2020; Dai et al., 2022; Mitchell et al., 2022b; Meng et al., 2022, 2023; Yao et al., 2023; Zhong et al., 2023; Ma et al., 2024; Gu et al., 2024), which aims to update the knowledge of LLMs cost-effectively. Some existing methods of model editing achieve this by modifying model parameters, which can be generally divided into two categories (Wang et al., 2023; Yao et al., 2023). Specifically, one type is based on *Meta-Learning* (Cao et al., 2021; Dai et al., 2022), while the other is based on *Locate-then-Edit* (Dai et al., 2022; Meng et al., 2022, 2023). This paper primarily focuses on the latter.

Sequential model editing (Yao et al., 2023) can expedite the continual learning of LLMs where

[*]Equal contribution.
[†]Corresponding author.
[1]Code is available: https://github.com/famoustourist/EAC

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 5514–5530

a series of consecutive edits are conducted. This is very important in real-world scenarios because new knowledge continually appears, requiring the model to retain previous knowledge while conducting new edits. Some studies have experimentally revealed that in sequential editing, existing methods lead to a decrease in the general abilities of the model across downstream tasks (Gu et al., 2024; Gupta et al., 2024; Yang et al., 2024; Hu et al., 2024). Besides, Ma et al. (2025) have performed a theoretical analysis to elucidate the bottleneck of the general abilities during sequential editing. However, previous work has not introduced an effective method that maintains editing performance while preserving general abilities in sequential editing. This impacts model scalability and presents major challenges for continuous learning in LLMs.

In this paper, a statistical analysis is first conducted to help understand how the model is affected during sequential editing using two popular editing methods, including ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023). Matrix norms, particularly the L1 norm, have been shown to be effective indicators of matrix properties such as sparsity, stability, and conditioning, as evidenced by several theoretical works (Kahan, 2013). In our analysis of matrix norms, we observe significant deviations in the parameter matrix after sequential editing. Besides, the semantic differences between the facts before and after editing are also visualized, and we find that the differences become larger as the deviation of the parameter matrix after editing increases. Therefore, we assume that each edit during sequential editing not only updates the editing fact as expected but also unintentionally introduces non-trivial noise that can cause the edited model to deviate from its original semantics space. Furthermore, the accumulation of non-trivial noise can amplify the negative impact on the general abilities of LLMs.

Inspired by these findings, a framework termed **E**diting **A**nchor **C**ompression (EAC) is proposed to constrain the deviation of the parameter matrix during sequential editing by reducing the norm of the update matrix at each step. As shown in Figure 1, EAC first selects a subset of dimension with a high product of gradient and magnitude values, namely editing anchors, that are considered crucial for encoding the new relation through a weighted gradient saliency map. Retraining is then performed on the dimensions where these important editing anchors are located, effectively compressing the editing information. By compressing information only in certain dimensions and leaving other dimensions unmodified, the deviation of the parameter matrix after editing is constrained. To further regulate changes in the L1 norm of the edited matrix to constrain the deviation, we incorporate a scored elastic net (Zou and Hastie, 2005) into the retraining process, optimizing the previously selected editing anchors.

To validate the effectiveness of the proposed EAC, experiments of applying EAC to **two popular editing methods** including ROME and MEMIT are conducted. In addition, **three LLMs of varying sizes** including GPT2-XL (Radford et al., 2019), LLaMA-3 (8B) (Meta, 2024) and LLaMA-2 (13B) (Touvron et al., 2023) and **four representative tasks** including natural language inference (Dagan et al., 2005), summarization (Gliwa et al., 2019), open-domain question-answering (Kwiatkowski et al., 2019), and sentiment analysis (Socher et al., 2013) are selected to extensively demonstrate the impact of model editing on the general abilities of LLMs. Experimental results demonstrate that in sequential editing, EAC can effectively preserve over 70% of the general abilities of the model across downstream tasks and better retain the edited knowledge.

In summary, our contributions to this paper are three-fold: (1) This paper statistically elucidates how deviations in the parameter matrix after editing are responsible for the decreased general abilities of the model across downstream tasks after sequential editing. (2) A framework termed EAC is proposed, which ultimately aims to constrain the deviation of the parameter matrix after editing by compressing the editing information into editing anchors. (3) It is discovered that on models like GPT2-XL and LLaMA-3 (8B), EAC significantly preserves over 70% of the general abilities across downstream tasks and retains the edited knowledge better.

2 Related Work

Model Editing Many model editing methods focus on updating knowledge in LLMs by modifying parameters, including meta-learning and locate-then-edit approaches (Yao et al., 2023; Wang et al., 2023). Meta-learning methods train a hypernetwork to apply gradient-based updates to model parameters (Cao et al., 2021; Mitchell et al.,

2022a), as in Knowledge Editor (KE) and MEND. Locate-then-edit methods identify MLPs storing factual knowledge and edit them by injecting new key-value pairs (Meng et al., 2022, 2023), leveraging the observation that these layers can act as key-value memories (Geva et al., 2021, 2022). Additionally, Zhu et al. (2020) proposed constrained fine-tuning for modified facts. Recent works extend these approaches to domains such as model personality changes (Mao et al., 2023), multimodal models (Cheng et al., 2023), and user privacy protection (Wu et al., 2023). Studies also focus on sequential editing scenarios, showing that as edits increase, general abilities tend to degrade (Gu et al., 2024; Lin et al., 2024; Jiang et al., 2024; Fang et al., 2024). Additionally, Ma et al. (2025) made a theoretical analysis to elucidate that the bottleneck of the general abilities during sequential editing lies in the escalating value of the condition number of the edited matrix.

Saliency Analyses In Computer Vision (CV), extensive research on input saliency maps has contributed to explainable machine learning. Methods include pixel-space sensitivity maps (Smilkov et al., 2017) and class-discriminative localization (Selvaraju et al., 2020). Data-level saliency, also called data attribution, has been widely studied for model explanation (Grosse et al., 2023), efficient training (Xie et al., 2023), and improving generalization (Jain et al., 2023). Compared to input saliency and data attribution, model saliency is less explored. Weight sparsity (Frankle and Carbin, 2019), used in weight pruning, can be viewed as a weight saliency map that preserves model abilities. In NLP, model editing research (Dai et al., 2022; Meng et al., 2022, 2023) focuses on locating and modifying specific knowledge by targeting weights. This concept of an 'editable model region' aligns with weight saliency in NLP, where certain parameters are more influential and editable.

Compared with previous studies (Meng et al., 2022, 2023; Yao et al., 2023) that are the most relevant to our work, a main difference should be highlighted. These approaches target at designing editing algorithms or evaluation paradigms to improve or assess the performance of model editing. However, previous work has not provided an in-depth analysis or an effective solution to preserve these abilities based on that analysis. In contrast, our study seeks to analyze the factors for the degradation of the general abilities of the model in sequential editing and introduces the EAC framework to preserve these general abilities.

3 Preliminary

Model editing involves modifying the knowledge stored within LMs without requiring retraining, to better meet specific tasks or requirements. This process aims to refine various complex learned beliefs, including logical, spatial, and numerical knowledge. In this paper, we study editing factual knowledge in the form of (x_e, y_e)[2]. The language model $f_\theta \in \mathcal{F}$ can be defined as a function $f_\theta : \mathcal{X} \to \mathcal{Y}$, mapping input $x \in \mathcal{X}$ to its prediction $y \in \mathcal{Y}$. For an editing factual knowledge (x_e, y_e), where $f_\theta(x_e) \neq y_e$, the goal of the model editing is to edit the parameters $\theta \in \Theta$ of the model f_θ to obtain an edited model $f_{\theta'}$, such that $f_{\theta'}(x_e) = y_e$. In sequential editing, this process continues iteratively. Given a set of editing facts $\mathcal{E} = \{(x_{ei}, y_{ei}) \mid i = 1, \ldots, n\}$ and an initial model f_{θ_0}, each model editing step involves learning a function K that produces an edited language model f_{θ_i} such that $K(f_{\theta_{i-1}}, (x_{ei}, y_{ei})) = f_{\theta_i}$.

The model editing process typically affects the predictions for a broad set of inputs closely related to the edited factual knowledge. This collection of inputs is referred to as the *editing scope*. A successful edit should modify the model's behavior within the target scope while preserving performance on out-of-scope examples:

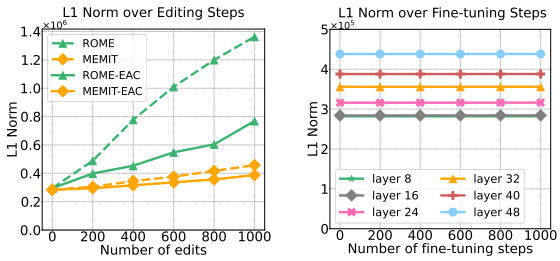
$$f_{\theta_i}(x_{ei}) = \begin{cases} y_{ei} & \text{if } x_{ei} \in I(x_{ei}, y_{ei}), \\ f_{\theta_{i-1}}(x_{ei}) & \text{if } x_{ei} \in O(x_{ei}, y_{ei}). \end{cases}$$

The *in-scope* $I(x_{ei}, y_{ei})$ typically includes x_{ei} and its equivalence neighborhood $N(x_{ei}, y_{ei})$, which encompasses related input/output pairs. In contrast, the *out-of-scope* $O(x_{ei}, y_{ei})$ comprises inputs unrelated to the edit example. To evaluate the effectiveness of various model editing methods, previous works focus on evaluation along three dimensions: *reliability*, *generalization* and *locality* (Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022, 2023; Yao et al., 2023).

4 Analysis of Ability Degradation

ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) are currently popular model editing methods. Given that MEMIT builds upon the

[2]Can be also represented as knowledge triple $t = (subject, relation, object)$.



(a) Editing-based methods (b) Fine-tuning approach

Figure 2: Illustration of the change of L1 norm (a) in sequential editing at the edited layer using editing-based methods and (b) in fine-tuning different batch steps for selected layers. Here we uniformly selected the layers of GPT2-XL for clarity when fine-tuning.

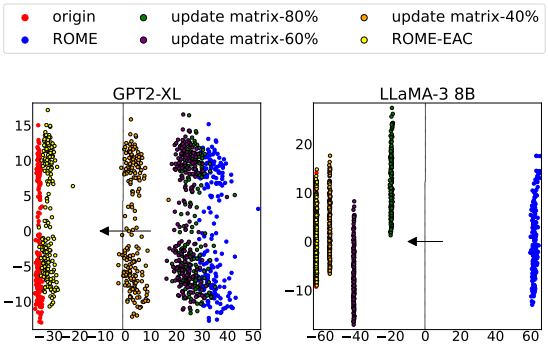


Figure 3: Visialization of six sets of facts recalled by LLMs using 2-dimensional PCA. Note that this hidden state is also projected by a language modeling head (linear mapping) for next-token prediction, implying the linear structure in the corresponding representation space (the PCA assumption).

foundations of ROME by implementing residual distribution across multiple layers, our analysis in the main text focuses primarily on ROME. This section presents a detailed analysis of how the model is affected during sequential editing using ROME. Statistical and visual analyses reveal that the degradation of general abilities is related to the unintentional introduction of the non-trivial noise that can make the parameter matrix after editing deviate from its original semantics space.

4.1 Comparison with Fine-tuning Approach

First, a statistical analysis is conducted by editing GPT2-XL (Radford et al., 2019) using the ZsRE (Levy et al., 2017) dataset. Considering the L1 norm effectively quantifies the absolute changes in parameter values pre- and post-editing, while providing insights into feature weight distributions within the matrix, it is used to represent the degree of change in the parameter matrix. As illustrated in Figure 2(a), when using editing-based methods such as ROME and MEMIT, the L1 norm of the matrix at the edited layer increases significantly with the number of edits. It can be seen that the norm increases by 317% (ROME) and 61% (MEMIT), respectively by the end of sequential editing. This result highlights a significant deviation from the unedited model, emphasizing the impact of sequential edits on stability.

A gradient-based fine-tuning approach can markedly enhance the performance of the model on specific tasks while preserving its general abilities across other downstream tasks (Moslem et al., 2023; Liu et al., 2023). As depicted in Figure 2(b), there are no significant changes in the norm of the parameter matrix for the given layers, with a maximum change of only 0.27%, even as the amount of fine-tuning knowledge increases. This stability in the parameter matrix norms suggests that the fine-tuning approach does not introduce significant non-trivial noise during the editing process. Thus, fine-tuning maintains the integrity and stability of the model's parameters, which is crucial for preserving its general abilities and preventing unintended non-trivial noise.

This comparison indicates that each editing method not only updates the intended fact as expected but also unintentionally introduces non-trivial noise into the model. This noise manifests as deviations in the parameter matrix during the sequential editing process. With each additional edit, the noise accumulates, progressively increasing the deviation in the parameter matrix. Consequently, as the number of edits grows, there is a significant deviation in the parameter matrix observed before and after the editing. This accumulated noise highlights the challenge of maintaining the stability and integrity of the parameter matrix through multiple edits, which can ultimately impact the general abilities of the model.

4.2 Visualization Analysis

Following the ROME, the second layer of MLP $W_{\text{proj}}^{(l)}$ is viewed as a linear associative memory (Anderson, 1972; Kohonen, 1972). This perspective observes that any linear operation W can operate as a key-value store for a set of vector keys $K = [\mathbf{k}_1|\mathbf{k}_2|\dots]$ and corresponding vector values $V = [\mathbf{v}_1|\mathbf{v}_2|\dots]$, by solving $WK \approx V$ (Meng et al., 2022). The key-value pair $(\mathbf{k}_i, \mathbf{v}_i)$ represents the representation of the input prompt, where $\mathbf{k}_i$ identifies patterns of the input and $\mathbf{v}_i$ is the fact

recalled by the model, which is considered to gather all the information about how the model understands the prompt and how it will respond. By stacking $\mathbf{k}_i$ and $\mathbf{v}_i$ separately for each prompt, matrix K and V are obtained. Based on this, 200 prompts of the same downstream task are collected to compute K and V.

On GPT2-XL and LLaMA-3 (8B), Principal Component Analysis is employed to visualize the hidden state of the facts of the downstream task recalled by the model. The first two principal components of six sets of facts, representing most features, are computed (Zheng et al., 2024). Two of these are derived from recalling the model before editing and the model after editing without any constraint, respectively. To explore the relationship between the deviation of the parameter matrix after editing and the resulting degradation of general abilities, four additional settings were tested by setting different percentages of the columns in the update matrix to zero, evenly distributed according to an arithmetic progression.

As illustrated in Figure 3, the principal components of facts recalled by the original model and the edited model without any constraint can be largely distinguished, whose boundaries (black dashed lines) can be easily fitted using logistic regression. This indicates a significant semantic discrepancy between the facts recalled by the unconstrained edited model and the original model, explaining the decline in general abilities is related to matrix deviation. Furthermore, when the deviation of the parameter matrix is constrained by reducing the norm of the update matrix, as shown by the black arrows, the principal components of the recalled facts by the edited model gradually align with those of the original model. This shows that by reducing the norm of the update matrix, the deviation of the parameter matrix after editing can be constrained, making the semantic distribution of the model before and after editing similar, thereby preserving the general abilities of the edited model.

5 EAC: Editing Anchor Compression

In Section 4, an in-depth analysis is provided on the factors that lead to the decrease in the general abilities of the model. Besides, it is found that the deviation of the edited parameter matrix could be constrained by reducing the norm of the update matrix at each edit. This helps maintain the semantic similarity of the facts recalled by the model before and after editing, ultimately preserving the general abilities of the edited model. As depicted in Figure 4, a framework called EAC is proposed to compress information only in certain dimensions, thereby reducing the norm of the edited matrix and further constraining its deviation.

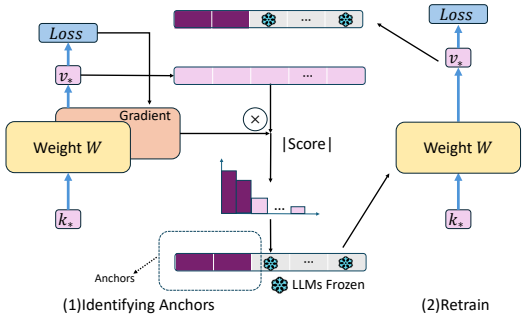


Figure 4: Proposed method: EAC. We first identify the key dimensions of the editing anchors using a weighted-gradient saliency map, followed by retraining on these dimensions to achieve the final optimization.

5.1 Definition of Editing Anchors

ROME uses an update matrix to insert a new knowledge triple $t = (subject, relation, object)$. As mentioned in Section 4.2, ROME calculates the update matrix by multiplying the pair $(\mathbf{k}_*, \mathbf{v}_*)$, where $\mathbf{k}_*$ identifies patterns of the input at the specified layer[3] and $\mathbf{v}_*$ is the fact recalled by the model. Readers can refer to Appendix B for the details of ROME. When injecting a new knowledge triple $t = (subject, relation, object^*)$ to replace an old one $t = (subject, relation, object)$, the specific part we aim to modify is the new relation $(relation, object^*)$, which is a property of the subject. It is believed that $\mathbf{v}_*$ gathers all the information about how the model understands the subject and how it will respond thus we think that the new relation is primarily encoded in $\mathbf{v}_*$. Thus, EAC chooses to reduce the norm of $\mathbf{v}_*$ for compression. Using a weighted gradient saliency map, EAC identifies high-scoring editing anchors crucial for encoding new relations. A scored elastic net is then applied to retrain and compress the editing information in key dimensions.

5.2 Weighted-gradient Saliency Map

To reduce the norm of $\mathbf{v}_*$ while preserving as much editing information as possible about the new relation, the goal is to compress this information over the smallest possible dimension. Drawing

[3]Found by causal tracing methods (Meng et al., 2022).

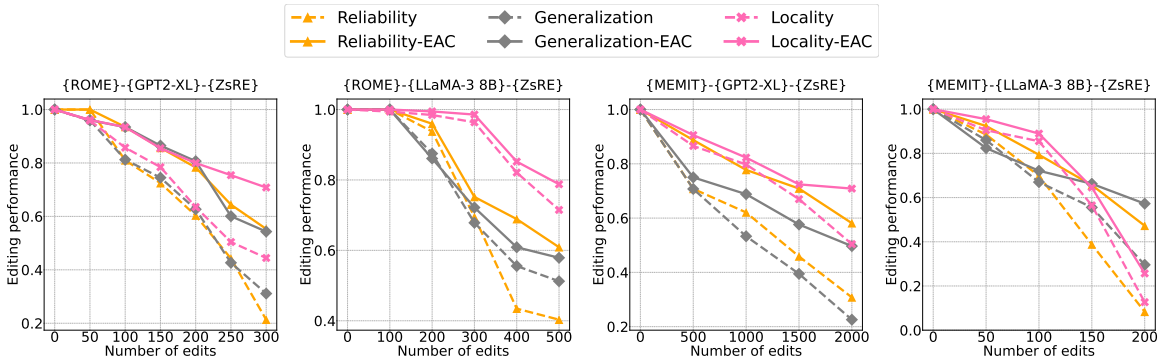


Figure 5: Edited on the ZsRE dataset, the sequential editing performance of ROME and MEMIT with GPT2-XL and LLaMA-3 (8B) before and after the introduction of EAC, as the number of edits increases.

inspiration from gradient-based input saliency maps (Smilkov et al., 2017; Adebayo et al., 2018), a question is posed that whether a *weight saliency map* can be constructed to aid compression. In previous work, ROME set $\mathbf{v}_* = \arg\min_z \mathcal{L}(\mathbf{z})$ (Meng et al., 2022). Similar to the input saliency map, the gradient of this loss function with respect to each feature is utilized, and the magnitude of the values of $\mathbf{v}_*$ is weighted accordingly to serve as the score for each feature:

$$\text{score} = \mathbf{v}_* \odot \nabla\mathcal{L}(\mathbf{z}), \tag{1}$$

where $\odot$ is element-wise product. For GPT2-XL, the vectors $\mathbf{v}_*$ and $\mathbf{z}$ have dimensions of 1600×1, whereas for LLaMA-3 (8B), the dimensions of these vectors are 4096×1. Based on Eq. (1), the desired weighted-gradient saliency map is obtained by applying a hard threshold:

$$\mathbf{m}_S = \mathbf{1}\left(|\text{score}| \geq \gamma\right), \tag{2}$$

where $\mathbf{1}(\mathbf{g} \geq \gamma)$ is an element-wise indicator function, which yields a value of 1 for the i-th element if $g_i \geq \gamma$ and 0 otherwise. $|\cdot|$ is an element-wise absolute value operation, and $\gamma > 0$ is a hard threshold. In practice, the value of γ is chosen according to different models and methods. Specifically, the value of γ is chosen to retain the editing performance of the model in single editing. For more details refer to Appendix A.

With the introduction of the weighted gradient saliency map, the dimension of $\mathbf{v}_*$ is split into two parts: one part represents the dimensions where the important editing anchors are located and it will be retrained to encode new relation, while the other part is set to 0, thereby reducing the norm of $\mathbf{v}_*$. Based on Eq. (2), the $\mathbf{v'}_*$, can be expressed as:

$$\mathbf{v'}_* \leftarrow \mathbf{m}_S \odot (\Delta\mathbf{v}_* + \mathbf{v}_*) + (\mathbf{1} - \mathbf{m}_S) \odot \mathbf{0}, \tag{3}$$

where $\mathbf{1}$ denotes an all-one vector and $\mathbf{0}$ denotes an all-zero vector. $\Delta\mathbf{v}_*$ is the part of $\mathbf{v}_*$ that requires updating during retraining. Eq. (3) demonstrates that during retraining, only the dimensions where these important anchors are located need to be retrained to compress the editing information.

5.3 Retraining Based on Scored Elastic Net

After choosing important anchors, retraining for $\mathbf{v}_*$ is performed in this section. To further compress the editing information, inspired by Zou and Hastie (2005), a score-based elastic net is also introduced during retraining:

$$\mathcal{L}_0(\mathbf{z}) = \lambda\|\mathbf{z}\|_{1,\alpha} + \mu\|\mathbf{z}\|_2^2, \tag{4}$$

where λ and μ are the hyper-parameters that control the strength of regularization and $\mathbf{z}$ is the vector that causes the network to predict the target object in response to the factual prompt. Detailed hyper-parameters can be referred to in Appendix C.5. Considering that the score computed in Eq. (1) represents the importance of the anchors for encoding the new relation, a weighted L1 norm is utilized when computing the L1 norm:

$$\|\mathbf{z}\|_{1,\alpha} = \sum_{i=1}^{n} \alpha_i |z_i|. \tag{5}$$

In practice, we set $\alpha = \frac{1}{\text{score}+\epsilon}$, a small positive number ϵ is introduced to prevent the score from being zero. Applying an elastic network, we ultimately derived the loss function during the retraining process to get the $\mathbf{v'}_*$ in Eq. (3):

$$\mathcal{L}_r(\mathbf{z}) = \mathcal{L}(\mathbf{z}) + \mathcal{L}_0(\mathbf{z}). \tag{6}$$

It is worth noting that when we make optimization here, only the dimensions where the editing anchors identified in section 5.2 are modified.

By introducing the elastic net, L1 regularization enables refining the selection of the editing anchors identified through the weighted-gradient saliency map during the retraining process. Meanwhile, L2 regularization effectively prevents model overfitting and improves the model's stability. Finally, we complete the optimization. For specific optimization details, we recommend interested readers to refer to Appendix B. Furthermore, the scored elastic net can also be applied to the FT. Readers can refer to Appendix E for more details.

6 Experiments

6.1 Experimental Setup

Experiments were conducted on three LLMs, GPT2-XL (Radford et al., 2019), LLaMA-3 (8B) (Meta, 2024) and LLaMA-2 (13B) (Touvron et al., 2023), using ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) as baseline editing methods. The editing performance was evaluated on two datasets: ZsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022), using reliability, generalization, and locality metrics (Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022, 2023; Yao et al., 2023). Four downstream tasks were selected to measure the general abilities of models before and after editing: **Natural language inference (NLI)**, **Open-domain QA**, **Summarization** and **Sentiment analysis**. Readers can refer to Appendix C for more details.

6.2 Main Results

This section illustrates the editing performance and downstream task performance of edited models with GPT2-XL and LLaMA-3 (8B) on the ZsRE dataset. Due to page limitation, results of other LLMs and datasets were put in Appendix D

Editing Performance In previous work, the evaluation of editing performance has primarily focused on single editing scenarios, meaning that only the success of editing a single fact is assessed. However, in sequential editing, the goal is for the model to retain all prior knowledge. To evaluate this, a set of sequential edits was applied, and the final model's reliability, generalization, and locality were assessed. Applying ZsRE as the editing dataset, Figure 5 shows the sequential editing performance of ROME and MEMIT on GPT2-XL and LLaMA-3 (8B) before and after the introduction of EAC. The dashed line represents the ROME or MEMIT, while the solid line represents the ROME or MEMIT applying the EAC. As sequential edits increase, models using ROME or MEMIT methods show a significant decline in reliability, generalization, and locality, retaining only partial knowledge from the latest edits while forgetting earlier information. Besides, as shown in Figure 5, the introduction of EAC brings improvements in the editing performance in sequential editing scenarios. By reducing the non-trivial noise introduced with each edit, EAC helps the model to retain the knowledge more effectively compared to previous methods.

General Abilities Applying ZsRE as the editing dataset, Figure 6 shows the performance on general tasks of ROME and MEMIT on GPT2-XL and LLaMA-3 (8B) before and after the introduction of EAC. The dashed line represents the ROME or MEMIT, while the solid line represents the ROME or MEMIT applying the EAC. It can be seen that, when using the ROME or the MEMIT for sequential editing, the performance of the edited models on various tasks fluctuates significantly and shows a downward trend as the number of edits increases. After applying EAC, the general abilities of the model on downstream tasks are well preserved. However, the performance of the model on downstream tasks inevitably declined when the number of sequential edits was high. This indicates that some non-trivial noise is still introduced with each edit even when applying EAC. As these non-trivial noises accumulate, the general abilities of the model are compromised.

6.3 Ablation Study

To validate the effectiveness of EAC, we performed ablation tests by removing either the weighted-gradient saliency map or the scored elastic net from EAC. As depicted in Figure 7, we find that both the weighted-gradient saliency maps and the score-based elastic net retain a certain level of the general abilities of the model. It can also be seen that removing either the weighted-gradient saliency map or the scored elastic net results in a decreased ability to preserve the general abilities of the model compared to EAC, illustrating the effectiveness of EAC. In addition, we observe a more significant decrease in the general abilities of the model when the weighted-gradient saliency map is removed. This suggests that the weighted-gradient saliency map plays a crucial role in EAC by effectively reducing the norm of the update matrix, which

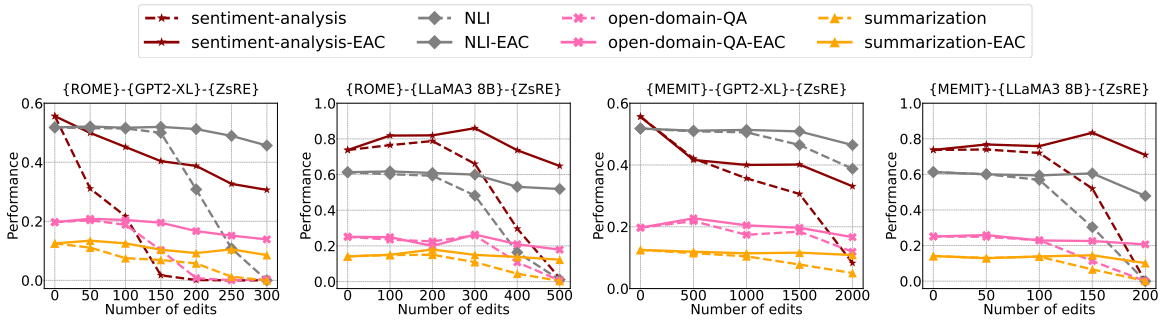


Figure 6: Edited on the ZsRE dataset, the general task performance of ROME and MEMIT with GPT2-XL and LLaMA-3 (8B) before and after the introduction of EAC, as the number of edits increases.

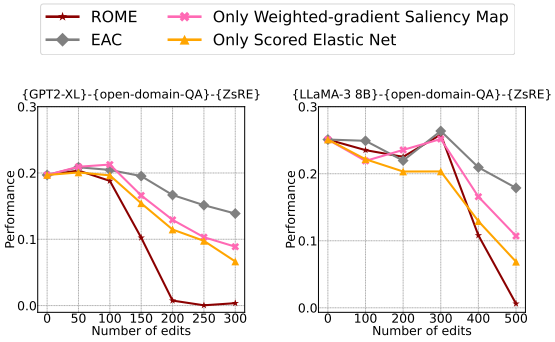


Figure 7: Ablation analysis of performance on open-domain-QA for EAC. Results were conducted with GPT2-XL and LLaMA-3 (8B) on the ZsRE dataset.

	GPT2-XL	LLaMA3-8B
ROME	6.70s	27.53s
ROME-EAC	6.21s	23.74s
MEMIT	5.23s	11.73s
MEMIT-EAC	5.92s	12.66s

Table 1: Comparison of editing time between the EAC framework and original methods

achieves this by setting some dimensions of $\mathbf{v}_*$ to zero where the editing anchors are located, thereby constraining the deviation of the edited matrix and finally preserving the general abilities of the model.

6.4 Time Analysis

In the EAC framework, the original training process for updating new knowledge is divided into two stages: first, selecting the important anchors, and second, retraining to complete the knowledge update. Using GPT2-XL as an example, the original ROME method applies 20 optimization steps to update knowledge. In contrast, the EAC framework allocates 10 optimization steps to identify important editing anchors, followed by another 10 steps of retraining to finalize the knowledge update. Additionally, since we reuse the

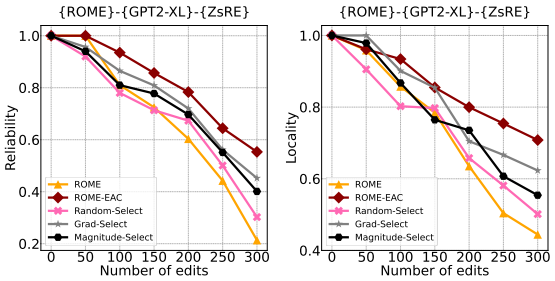


Figure 8: Edited on the ZsRE dataset, the editing performance of different methods with GPT2-XL as the number of edits increases.

gradient information generated during the selection of editing anchors, the EAC framework remains largely consistent with the previous method in terms of editing time and computational resource consumption, without incurring any extra computational overhead. This result is presented in Table 1.

6.5 Editing Anchors Selecting Analysis

In section 5, we employed a weighted-gradient saliency map to identify editing anchors, demonstrating the efficacy of this approach in subsequent experiments. Here, we delve deeper into the rationale behind using a weighted-gradient saliency map for determining these important anchors. The weighted-gradient saliency map combines gradient sensitivity and vector magnitude to prioritize dimensions that significantly influence the knowledge vector, ensuring precise edits by focusing on areas both sensitive to changes and substantively important. We validated this approach by comparing it to methods using random selection, gradients alone, or absolute values. Figure 8 demonstrates that the weighted-gradient saliency map consistently outperforms other methods when applied to GPT2-XL on the ZsRE dataset. The superior performance of the weighted-gradient saliency map emphasizes its enhanced capability

in precisely identifying key editing anchors within the model, ensuring greater accuracy and efficiency in the editing process.

7 Conclusion

This paper focuses on sequential model editing. Statistical and visual analyses reveal that each edit not only updates the desired fact but also introduces non-trivial noise, causing the model to deviate from its original semantic space. The accumulation of this noise negatively impacts the general abilities of LLMs. To address this issue, a framework termed **E**diting **A**nchor **C**ompression (EAC) is proposed, which constrains edited matrix deviation by reducing the update matrix norm. Experimental results show that EAC can effectively preserve the general abilities of edited models and the accuracy of editing facts in sequential editing. For future work, we aim to investigate the impact of complex edits and integrate EAC with other editing methods.

Limitations

Despite the effectiveness of EAC, our current studies still have limitations. Firstly, similar to previous model editing research, we focus on factual knowledge assessment. However, it is also important to investigate whether editing other types of knowledge will affect general abilities and whether EAC is effective in this situation. Secondly, our work focuses on sequential editing, where one fact is edited at a time. However, in real-world applications, we may need to change multiple facts in a single edit. Therefore, to enhance the scalability of model editing, batch-sequential settings should be emphasized in future studies. Finally, to more fully validate the effectiveness of EAC, experiments should be conducted on larger-size models and across more downstream tasks.

Acknowledgements

This work is funded by the National Science and Technology Major Project (No.2023ZD0121103). We would like to express gratitude to the anonymous reviewers for their kind comments.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9525–9536.

James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6491–6506. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. 2023. Can we edit multimodal large language models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13877–13888. Association for Computational Linguistics.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 8493–8502. Association for Computational Linguistics.

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

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts

in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 30–45. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5484–5495. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China. Association for Computational Linguistics.

Roger B. Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamile Lukosiute, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. 2023. Studying large language model generalization with influence functions. *CoRR*, abs/2308.03296.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 15202–15232. Association for Computational Linguistics.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3476–3503. Association for Computational Linguistics.

Saachi Jain, Hadi Salman, Alaa Khaddaj, Eric Wong, Sung Min Park, and Aleksander Madry. 2023. A data-based perspective on transfer learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 3613–3622. IEEE.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38.

Houcheng Jiang, Junfeng Fang, Tianyu Zhang, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. 2024. Neuron-level sequential editing for large language models. *arXiv preprint arXiv:2410.04045*.

W Kahan. 2013. A tutorial overview of vector and matrix norms. *University of California, Berkeley, CA, USA, Lecture notes*, page 19.

Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE Trans. Computers*, 21(4):353–359.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096. Association for Computational Linguistics.

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), Vancouver, Canada, August 3-4, 2017*, pages 333–342. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zihao Lin, Mohammad Beigi, Hongxuan Li, Yufan Zhou, Yuxiang Zhang, Qifan Wang, Wenpeng Yin, and Lifu Huang. 2024. Navigating the dual facets: A comprehensive evaluation of sequential memory editing in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13755–13772. Association for Computational Linguistics.

Yixin Liu, Avi Singh, C. Daniel Freeman, John D. Co-Reyes, and Peter J. Liu. 2023. Improving large language model fine-tuning for solving math problems. *CoRR*, abs/2310.10047.

Jun-Yu Ma, Zhen-Hua Ling, Ningyu Zhang, and Jia-Chen Gu. 2024. Neighboring perturbations of

knowledge editing on large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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

Shengyu Mao, Ningyu Zhang, Xiaohan Wang, Mengru Wang, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2023. Editing personality for llms. *CoRR*, abs/2310.02168.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *NeurIPS*.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Yasmin Moslem, Rejwanul Haque, and Andy Way. 2023. Fine-tuning large language models for adaptive machine translation. *CoRR*, abs/2312.12740.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *CoRR*, abs/2302.12813.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 1339–1384. Association for Computational Linguistics.

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.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2020. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry V. Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. 2017. Smooth-grad: removing noise by adding noise. *CoRR*, abs/1706.03825.

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, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, and Huajun Chen. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *CoRR*, abs/2308.07269.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: detecting and editing privacy neurons in pretrained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 2875–2886. Association for Computational Linguistics.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. 2023. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. 2024. The butterfly effect of model editing: Few edits can trigger large language models collapse. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 5419–5437. Association for Computational Linguistics.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 10222–10240. Association for Computational Linguistics.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219.

Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. 2024. On prompt-driven safeguarding for large language models. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 15686–15702. Association for Computational Linguistics.

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

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320.

A Hard Threshold of EAC

In constructing a weighted-gradient saliency map, the value of γ determines the number of the dimensions we select where important feature anchors are located. As the value of γ increases, the number of selected dimensions decreases, requiring the editing information to be compressed into a smaller space during the compression process. During compression, it is desired for the compression space to be as small as possible to preserve the general abilities of the model. However, reducing the compression space inevitably increases the loss of editing information, which reduces the editing performance of the model. Therefore, to ensure editing performance in a single editing scenario, different values of γ are determined for various models, methods, and datasets. Fifty pieces of knowledge were randomly selected from the dataset, and reliability, generalization, and locality were measured after editing. The averages of these metrics were then taken as a measure of the editing performance of the model. Table 2 presents the details of γ, while Table 3 illustrates the corresponding editing performance before and after the introduction of EAC. P_x denotes the value below which x% of the values in the dataset.

Table 2: The value of γ.

Datasets	**Model**	**ROME**	**MEMIT**
ZSRE	GPT-2 XL	P_{80}	P_{80}
	LLaMA-3 (8B)	P_{90}	P_{95}
COUNTERFACT	GPT-2 XL	P_{85}	P_{85}
	LLaMA-3 (8B)	P_{95}	P_{95}

Table 3: The value of γ.

Dataset	Method	**GPT-2 XL**			**LLaMA-3 (8B)**		
		Reliability	Generalization	Locality	Reliability	Generalization	Locality
ZsRE	ROME	1.0000	0.9112	0.9661	1.0000	0.9883	0.9600
	ROME-EAC	1.0000	0.8923	0.9560	0.9933	0.9733	0.9742
	MEMIT	0.6928	0.5208	1.0000	0.9507	0.9333	0.9688
	MEMIT-EAC	0.6614	0.4968	0.9971	0.9503	0.9390	0.9767
CounterFact	ROME	1.0000	0.4200	0.9600	1.0000	0.3600	0.7800
	ROME-EAC	0.9800	0.3800	0.9600	1.0000	0.3200	0.8800
	MEMIT	0.9000	0.2200	1.0000	1.0000	0.3800	0.9500
	MEMIT-EAC	0.8000	0.1800	1.0000	1.0000	0.3200	0.9800

B Optimization Details

ROME derives a closed-form solution to achieve the optimization:

$$\text{minimize } \|\widehat{W}K - V\| \text{ such that } \widehat{W}\mathbf{k}_* = \mathbf{v}_* \text{ by setting } \widehat{W} = W + \Lambda(C^{-1}\mathbf{k}_*)^T. \tag{7}$$

Here W is the original matrix, $C = KK^T$ is a constant that is pre-cached by estimating the uncentered covariance of $\mathbf{k}$ from a sample of Wikipedia text, and $\Lambda = (\mathbf{v}_* - W\mathbf{k}_*)/((C^{-1}\mathbf{k}_*)^T\mathbf{k}_*)$ is a vector proportional to the residual error of the new key-value pair on the original memory matrix.

In ROME, $\mathbf{k}_*$ is derived from the following equation:

$$\mathbf{k}_* = \frac{1}{N}\sum_{j=1}^{N}\mathbf{k}(x_j + s), \quad \text{where} \quad \mathbf{k}(x) = \sigma\left(W_{f_c}^{(l^*)}\gamma\left(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)}\right)\right). \tag{8}$$

ROME set $\mathbf{v}_* = \arg\min_z \mathcal{L}(z)$, where the objective $\mathcal{L}(z)$ is:

$$\frac{1}{N}\sum_{j=1}^{N} -\log \mathbb{P}_{G(m_i^{l^*}:=z))}\left[o^* \mid x_j + p\right] + D_{KL}\left(\mathbb{P}_{G(m_i^{l^*}:=z)}\left[x \mid p'\right] \parallel \mathbb{P}_G\left[x \mid p'\right]\right). \tag{9}$$

C Experimental Setup

C.1 Editing Methods

In our experiments, Two popular editing methods including ROME and MEMIT were selected as baselines.

ROME (Meng et al., 2022): it first localized the factual knowledge at a specific layer in the transformer MLP modules, and then updated the knowledge by directly writing new key-value pairs in the MLP module.

MEMIT (Meng et al., 2023): it extended ROME to edit a large set of facts and updated a set of MLP layers to update knowledge.

The ability of these methods was assessed based on EasyEdit (Wang et al., 2023), an easy-to-use knowledge editing framework which integrates the released codes and hyperparameters from previous methods.

C.2 Editing Datasets

In our experiment, two popular model editing datasets ZSRE (Levy et al., 2017) and COUNTERFACT (Meng et al., 2022) were adopted.

ZSRE is a QA dataset using question rephrasings generated by back-translation as the equivalence neighborhood. Each input is a question about an entity, and plausible alternative edit labels are sampled from the top-ranked predictions of a BART-base model trained on ZSRE.

COUNTERFACT accounts for counterfacts that start with low scores in comparison to correct facts. It constructs out-of-scope data by substituting the subject entity for a proximate subject entity sharing a predicate. This alteration enables us to differentiate between superficial wording changes and more significant modifications that correspond to a meaningful shift in a fact.

C.3 Metrics for Evaluating Editing Performance

Reliability means that given an editing factual knowledge, the edited model should produce the expected predictions. The reliability is measured as the average accuracy on the edit case:

$$\mathbb{E}_{(x'_{ei},y'_{ei})\sim\{(x_{ei},y_{ei})\}}\mathbf{1}\left\{\arg\max_{y} f_{\theta_i}\left(y \mid x'_{ei}\right) = y'_{ei}\right\}. \tag{10}$$

Generalization means that edited models should be able to recall the updated knowledge when prompted within the editing scope. The generalization is assessed by the average accuracy of the model on examples uniformly sampled from the equivalence neighborhood:

$$\mathbb{E}_{(x'_{ei},y'_{ei})\sim N(x_{ei},y_{ei})}\mathbf{1}\left\{\arg\max_{y} f_{\theta_i}\left(y \mid x'_{ei}\right) = y'_{ei}\right\}. \tag{11}$$

Locality means that the edited model should remain unchanged in response to prompts that are irrelevant or the out-of-scope. The locality is evaluated by the rate at which the edited model's predictions remain unchanged compared to the pre-edit model.

$$\mathbb{E}_{(x'_{ei},y'_{ei})\sim O(x_{ei},y_{ei})}\mathbf{1}\left\{f_{\theta_i}\left(y \mid x'_{ei}\right) = f_{\theta_{i-1}}\left(y \mid x'_{ei}\right)\right\}. \tag{12}$$

C.4 Downstream Tasks

Four downstream tasks were selected to measure the general abilities of models before and after editing: **Natural language inference (NLI)** on the RTE (Dagan et al., 2005), and the results were measured by accuracy of two-way classification. **Open-domain QA** on the Natural Question (Kwiatkowski et al., 2019), and the results were measured by exact match (EM) with the reference answer after minor normalization as in Chen et al. (2017) and Lee et al. (2019). **Summarization** on the SAMSum (Gliwa et al., 2019), and the results were measured by the average of ROUGE-1, ROUGE-2 and ROUGE-L as in Lin (2004). **Sentiment analysis** on the SST2 (Socher et al., 2013), and the results were measured by accuracy of two-way classification.

The prompts for each task were illustrated in Table 4.

NLI: {SENTENCE1} entails the {SENTENCE2}. True or False? answer:
Open-domain QA: Refer to the passage below and answer the following question. Passage: {DOCUMENT} Question: {QUESTION}
Summarization: {DIALOGUE} TL;DR:
Sentiment analysis: For each snippet of text, label the sentiment of the text as positive or negative. The answer should be exact 'positive' or 'negative'. text: {TEXT} answer:

Table 4: The prompts to LLMs for evaluating their zero-shot performance on these general tasks.

C.5 Hyper-parameters for Elastic Net

In our experiment, we set $\lambda = 5 \times 10^{-7}$, $\mu = 5 \times 10^{-1}$ for GPT2-XL(Radford et al., 2019) and $\lambda = 5 \times 10^{-7}$, $\mu = 1 \times 10^{-3}$ for LLaMA-3 (8B)(Meta, 2024).

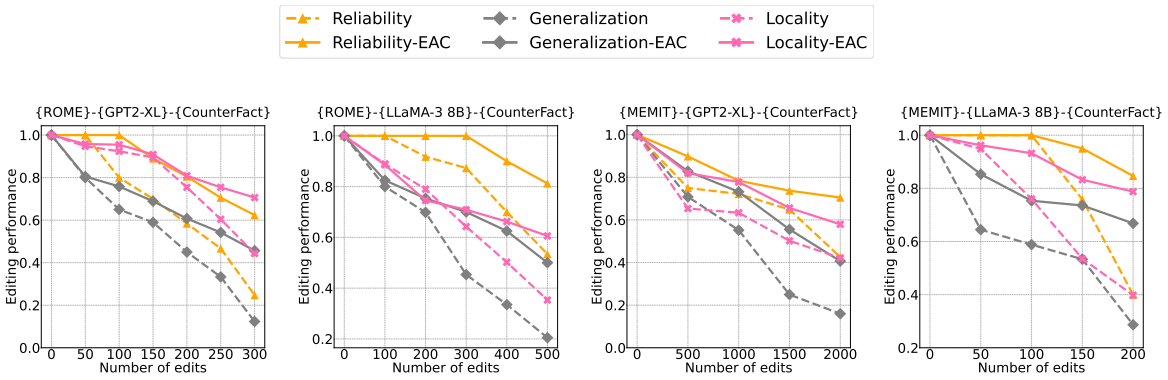


Figure 9: Edited on CounterFact, editing performance of edited models using the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) on GPT2-XL (Radford et al., 2019) and LLaMA-3 (8B) (Meta, 2024), as the number of edits increases before and after the introduction of EAC.

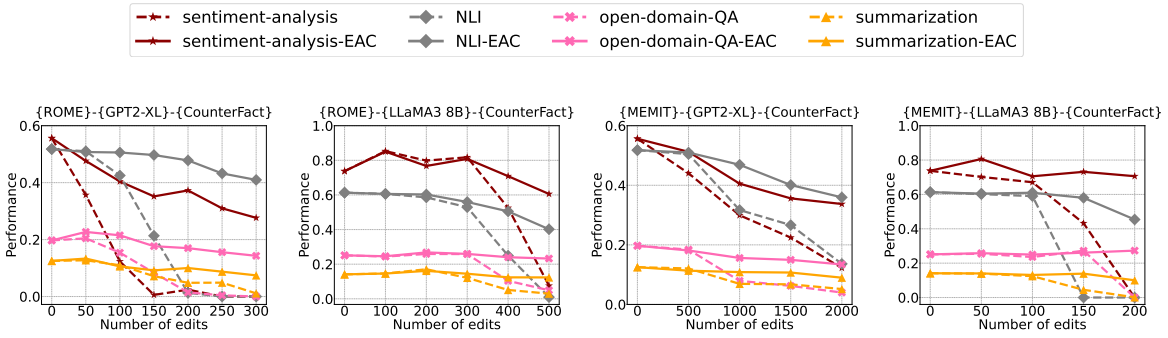


Figure 10: Edited on CounterFact, performance on general tasks using the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) on GPT2-XL (Radford et al., 2019) and LLaMA-3 (8B) (Meta, 2024), as the number of edits increases before and after the introduction of EAC.

D Experimental Results

D.1 Results of Editing Performance

Applying CounterFact as the editing dataset, Figure 9 presents the editing performance of the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) methods on GPT2-XL (Radford et al.,

2019) and LLaMA-3 (8B) (Meta, 2024), respectively, as the number of edits increases before and after the introduction of EAC. The dashed line represents the ROME or MEMIT, while the solid line represents the ROME or MEMIT applying the EAC.

D.2 Results of General Abilities

Applying CounterFact as the editing dataset, Figure 10 presents the performance on general tasks of edited models using the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) methods on GPT2-XL (Radford et al., 2019) and LLaMA-3 (8B) (Meta, 2024), respectively, as the number of edits increases before and after the introduction of EAC. The dashed line represents the ROME or MEMIT, while the solid line represents the ROME or MEMIT applying the EAC.

D.3 Results of Larger Model

To better demonstrate the scalability and efficiency of our approach, we conducted experiments using the LLaMA-2 (13B) (Touvron et al., 2023). Figure 11 presents the editing performance of the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) methods on LLaMA-2 (13B) (Touvron et al., 2023), as the number of edits increases before and after the introduction of EAC. Figure 12 presents the performance on general tasks of edited models using the ROME and MEMIT methods on LLaMA-2 (13B), as the number of edits increases before and after the introduction of EAC. The dashed line represents the ROME or MEMIT, while the solid line represents the ROME or MEMIT applying the EAC.

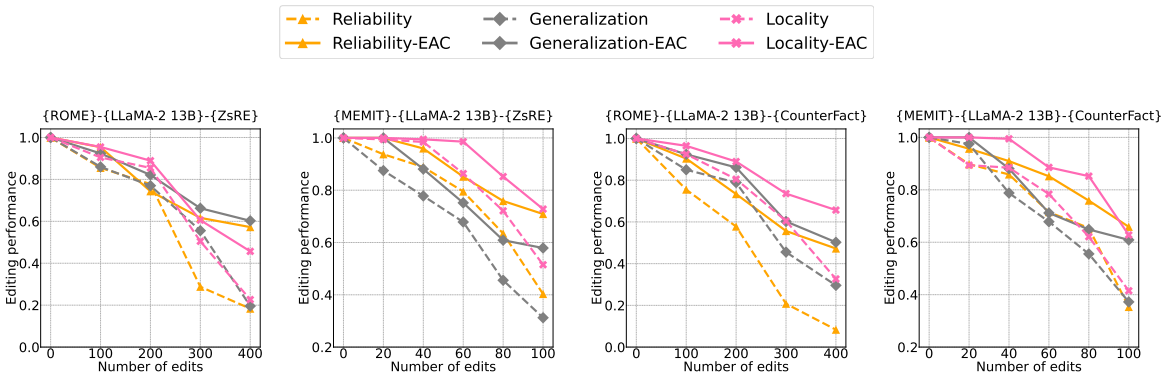


Figure 11: Editing performance of edited models using the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) on LLaMA-2 (13B) (Touvron et al., 2023), as the number of edits increases before and after the introduction of EAC.

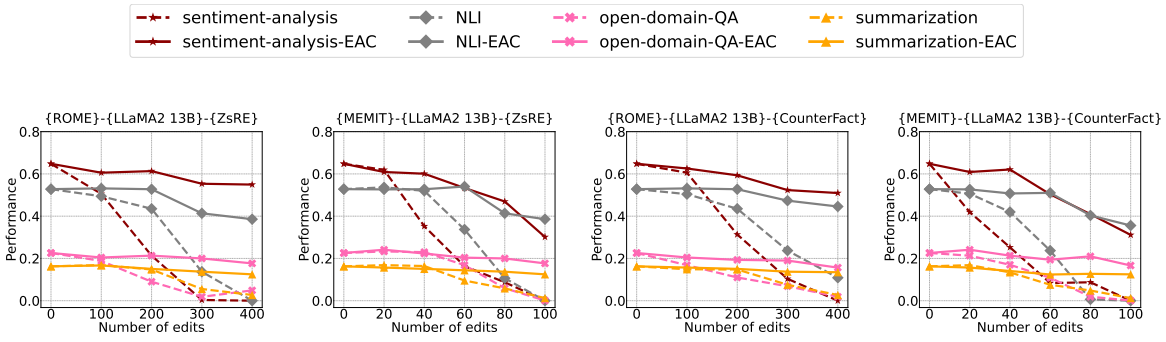


Figure 12: Performance on general tasks using the ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) on LLaMA-2 (13B) (Touvron et al., 2023), as the number of edits increases before and after the introduction of EAC.

E Analysis of Elastic Net

It is worth noting that the elastic net introduced in EAC can be applied to methods beyond ROME and MEMIT, such as FT (Cao et al., 2021), to preserve the general abilities of the model. Unlike the previously mentioned fine-tuning, FT is a model editing approach. It utilized the gradient to gather information about

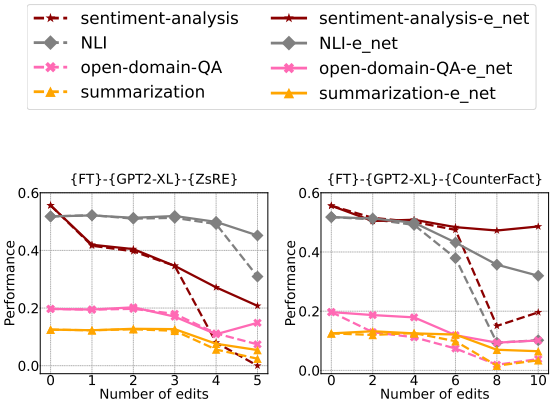


Figure 13: Edited on the ZsRE or CounterFact datasets, the sequential editing performance of FT (Cao et al., 2021) and FT with elastic net on GPT2-XL before and after the introduction of elastic net.

the knowledge to be updated and applied this information directly to the model parameters for updates. Similar to the approaches of ROME and MEMIT, which involve locating parameters and modifying them, the FT method utilizes gradient information to directly update the model parameters for editing. Therefore, we incorporate an elastic net during the training process to constrain the deviation of the edited matrix. Figure 13 shows the sequential editing performance of FT on GPT2-XL and LLaMA-3 (8B) before and after the introduction of elastic net. The dashed line represents the FT, while the solid line represents the FT applying the elastic net. The experimental results indicate that when using the FT method to edit the model, the direct use of gradient information to modify the parameters destroys the general ability of the model. By constraining the deviation of the edited matrix, the incorporation of the elastic net effectively preserves the general abilities of the model.