Knowledge Decoupling via Orthogonal Projection for Lifelong Editing of Large Language Models

Haoyu Xu[1], Pengxiang Lan[1], Enneng Yang[1], Guibing Guo[1]*, Jianzhe Zhao[1], Linying Jiang[1], Xingwei Wang[2]
[1]Software College, Northeastern University, China,
[2]School of Computer Science and Engineering, Northeastern University, China
{haoyuxu, pengxianglan, ennengyang}@stumail.neu.edu.cn,
{guogb, zhaojz, jiangly}@swc.neu.edu.cn, wangxw@mail.neu.edu.cn

Abstract

As large language models (LLMs) require continuous knowledge updates and the mitigation of hallucination issues in generated content, lifelong model editing has become a prominent research area. A mainstream knowledge editing method usually freezes LLM's original parameters and adds extra trainable modules for new knowledge management, reducing interference with old knowledge. Although these approaches have achieved some success, our experiments show that, after extensive editing, the model's knowledge understanding and memory capacity significantly degrade, particularly concerning early edited knowledge. The root cause is that subsequent edits interfere with the previously edited knowledge, and we refer to this phenomenon as knowledge coupling. To address this issue, we propose the Knowledge Decoupling Editing (KDE) method. Specifically, KDE stores the basis vectors of the representation space of past edits in a knowledge cache. It projects the gradient of the current edit onto a space orthogonal to previous knowledge for updating. This method effectively alleviates the coupling between different pieces of knowledge. We also propose a two-stage training strategy to better balance the model's ability to edit new knowledge and distinguish whether a query is related to previous edits. This strategy gradually reduces the interference between new knowledge editing and query distinction, maintaining stable performance during long-term editing. We compared KDE with nine state-of-the-art editing methods across multiple mainstream LLMs. The results demonstrate that, regarding question-answering ability and hallucination mitigation, KDE achieves average improvements of 14% and 61%.

1 Introduction

Large language models (LLMs) (Roumeliotis and Tselikas, 2023; Touvron et al., 2023a; DeepSeek-AI et al., 2024) have demonstrated significant potential across various applications. However, they often face issues such as factual errors (Balachandran et al., 2022), hallucinations (Ji et al., 2023), and outdated information (Cao et al., 2021), which undermine their reliability. Updating model knowledge through full fine-tuning is costly and may compromise the model's general capabilities.

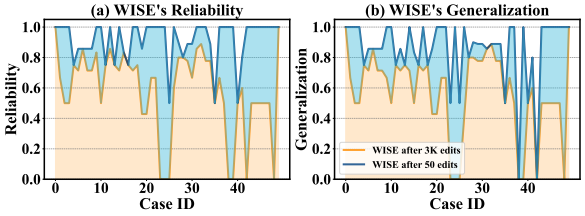

Figure 1: (a) and (b) show the changes in reliability and generalization for the first 50 samples in the WISE after 50 and 3,000 edits, respectively.

As a result, model editing techniques (Meng et al., 2022; Hartvigsen et al., 2023) have emerged as a more efficient alternative for modifying specific knowledge. Existing methods can be broadly categorized into two types (Zhang et al., 2024): (1) parameter-modifying methods (Meng et al., 2022; Gu et al., 2024a), which directly alter the model's parameters, yet they often change the model's output for queries that are unrelated to the editing. (2) parameter-preserving methods (Yu et al., 2024; Wang et al., 2024c), avoid this problem by freezing the parameters of LLMs and introducing additional parameters to edit knowledge. In this paper, we focus on the parameter-preserving method, because it can protect the general knowledge of LLM more effectively and absorb new knowledge.

However, most existing research has primarily focused on single or batch knowledge edits (Meng et al., 2022, 2023), which can no longer meet the demand for frequent knowledge updates in real-world applications. These methods often suffer from two critical limitations: (1) a significant de-

*Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13194–13213

cline in editing success rates during frequent editing and (2) unintended interference with knowledge unrelated to the edited content (Gu et al., 2024b; Gupta et al., 2024). To address the challenges of lifelong model editing, WISE (Wang et al., 2024c) has designed a knowledge sharding and merging mechanism, which can effectively alleviate the conflict issues caused by excessively high knowledge density (Allen-Zhu and Li, 2024). In addition, the method introduces a side memory to avoid interfering with the model's responses to knowledge unrelated to the edited content. Therefore, WISE performs excellently in scenarios that require frequent editing. Nevertheless, as the number of edits increases, the model's performance in retaining earlier knowledge significantly declines, as shown in Figure 1, with the reliability of the first 50 edit prompts decreasing by an average of 36% and generalization decreasing by 33%. The performance degradation highlights a critical challenge in lifelong model editing: the interference between newly edited and previously edited knowledge. Such interference often leads to the corruption or forgetting of earlier edits, a phenomenon we term as **knowledge coupling**. Figure 2 presents a specific instance of the knowledge coupling phenomenon in the ZsRE dataset (Levy et al., 2017). After applying the existing editing methods to the LLM for $t+1$ edits, we observed that the knowledge from the t-th edit was distorted. The original answer, "Swan River", was incorrectly replaced by a combination of the targets from the two edits, "Smithan River". This coupling phenomenon causes the model to mistakenly conflate the results of the two edits during inference. Solving this problem is crucial for the development of more powerful editing methods.

Based on these observations, this paper proposes a **K**nowledge **D**ecoupling **E**diting (KDE) method for lifelong editing of LLMs. KDE consists of four core components: (1) Editable Memory, (2) Switch Mechanism, (3) Knowledge Decoupling, and (4) Two-Stage Training. Specifically, *Editable Memory* is an additional parameter we introduce specifically for editing knowledge, designed to avoid the side effects of directly modifying the internal parameters of the LLM. It is a copy of the value matrix in a certain layer of the LLM's feedforward network (FFN) module. The *Switch Mechanism* determines whether the input is related to the editing content by assessing the activation differences between the input, Editable Memory, and the original value matrix, thereby preventing interference with the model's responses to knowledge unrelated to the edited content. *Knowledge Decoupling* includes a knowledge cache module that stores the representation space of the edited knowledge. During each edit, the gradients of the knowledge are constrained to optimize within the orthogonal space represented by the cache, thus reducing the coupling between different knowledge. However, when editing longer sequences, we observe a sudden performance drop. This occurs because Editable Memory must load new knowledge while maintaining differences from the original value matrix, resulting in increased interference between the two tasks during long sequence edits, with neither being effectively optimized. To address this issue, we propose a simple yet effective *Two-stage Training* strategy: during the early editing stages, both capabilities are trained simultaneously, after which the focus shifts to optimizing the edited knowledge. This approach allows our method to maintain stable and excellent performance even under ultra-long sequences (e.g., 5k edits).

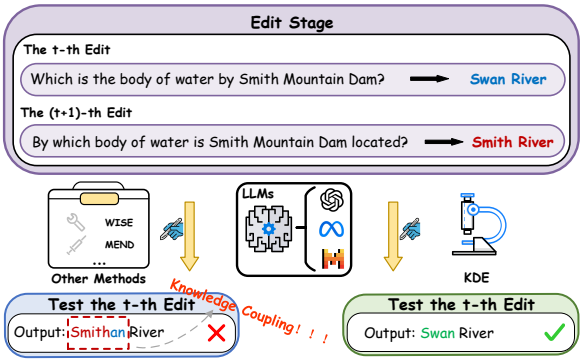

Figure 2: An example of knowledge coupling on ZsRE.

Our contributions can be summarized as follows:

- We identify one of the key challenges in lifelong editing as knowledge coupling, where new edits interfere with previously edited knowledge.
- We propose Knowledge Decoupling Editing (KDE), which reduces knowledge coupling by projecting new edit gradients into the orthogonal space of prior edited knowledge. We also introduce a two-stage training strategy to improve performance in long-sequence editing.
- Extensive experiments conducted on the ZsRE and SelfCheckGPT datasets, as well as with the GPT-J-6B, LLaMa2-7B, and Mistral-7B architectures, demonstrate the effectiveness of the proposed KDE method.

2 Background

2.1 Model Editing

In this section, we introduce the concepts of lifelong model editing and evaluation metrics.

Based on prior research (Meng et al., 2022, 2023), model editing refers to modifying a language model's parameters to change its behavior for specific instances while maintaining consistent behavior for other instances. Given a language model f_θ and an editing instance (x_e, y_e), the initial model may fail to produce the correct output y_e for the input x_e, i.e., $f_\theta(x_e) \neq y_e$. Model editing generates an edited model f'_θ by adjusting existing parameters or adding new ones, which satisfies $f'_\theta(x_e) = y_e$. In the lifelong editing (Huang et al., 2023; Hartvigsen et al., 2023), the model requires continuous updates. Starting from an initial language model f^0_θ, the editor E sequentially updates the model according to the editing requirements:

$$f^t_\theta = E(f^{t-1}_\theta, x_e, y_e), \quad t = 1, 2, 3, \ldots \quad (1)$$

In such scenarios, an ideal model editing method must exhibit three key characteristics: reliability, generalization, and locality (Wang et al., 2024c; Cai and Cao, 2024). Reliability measures the model's ability to accurately retain all edited instances; generalization evaluates the model's ability to make correct predictions for examples related to the edited instances; and locality ensures that the model's behavior remains unchanged when processing inputs unrelated to the edited instances, thereby preserving its original performance. The specific information about the evaluation metrics can be found in Appendix A.1.

2.2 Input and Gradient Spaces

Our method leverages the characteristic of stochastic gradient descent updates confined to the space spanned by the input data (Zhang et al., 2017; Saha et al., 2021). We will illustrate this property specifically for fully connected layers. Let $x \in \mathbb{R}^m$ be the input vector, $y \in \mathbb{R}^n$ be the corresponding label vector from the dataset, and $W \in \mathbb{R}^{m\times n}$ be the trainable weight matrix. The loss function is defined as the mean squared error loss $L = \frac{1}{2}\|Wx - y\|_2^2$.

The gradient of the loss function with respect to the weights can be expressed as:

$$\nabla_W L = (Wx - y)x^\top = \delta x^\top, \quad (2)$$

where $\delta \in \mathbb{R}^n$ is the error vector. Consequently, the gradient update occurs within the space spanned by the input x (a detailed proof is provided in Appendix B.1). This formulation is also valid for networks with other loss functions, such as cross-entropy, except that the computation of the error vector δ will differ.

3 Method

This section introduces our proposed method, Knowledge Decoupling Editing (KDE). The overall architecture is shown in Figure 3. It consists of four main components: (1) Editable memory, (2) Switch Mechanism, (3) Knowledge Decoupling, and (4) Two-Stage Training.

3.1 Editable Memory

To mitigate issues such as knowledge forgetting arising from directly editing the internal parameters of LLMs, we draw inspiration from the design in WISE(Wang et al., 2024c) and introduce additional parameters into the FFN of a specific transformer block in the LLMs to learn new knowledge. Recent research(Geva et al., 2021) has shown that the structure of an FFN is similar to a key-value memory, where it can be viewed as consisting of a key matrix W_k and a value matrix W_v. We refer to the newly added parameters as the Editable Memory W_e, which is directly copied from the FFN's value matrix W_v, and thus we term W_v as the Original Memory. Given an input x, the forward propagation of the Editable Memory is as follows:

$$FFN(x) = h \cdot W_e, \quad (3)$$

where $h = \sigma(x \cdot W_k^\top)$ represents the output of the key matrix W_k, and $\sigma(\cdot)$ is the activation function.

In the subsequent editing process, we update only the Editable Memory (i.e., W_e) while freezing the parameters of the original LLM. This approach aims to preserve the model's general capabilities while updating outdated or erroneous knowledge.

3.2 Switch Mechanism

After introducing Editable Memory, another important issue is determining whether a query x should enter Editable Memory or Original Memory during inference. Similar to retrieval-augmented methods (Huang et al., 2023; Wang et al., 2024c), our Switch Mechanism routes queries based on the activation difference between Editable Memory and Original

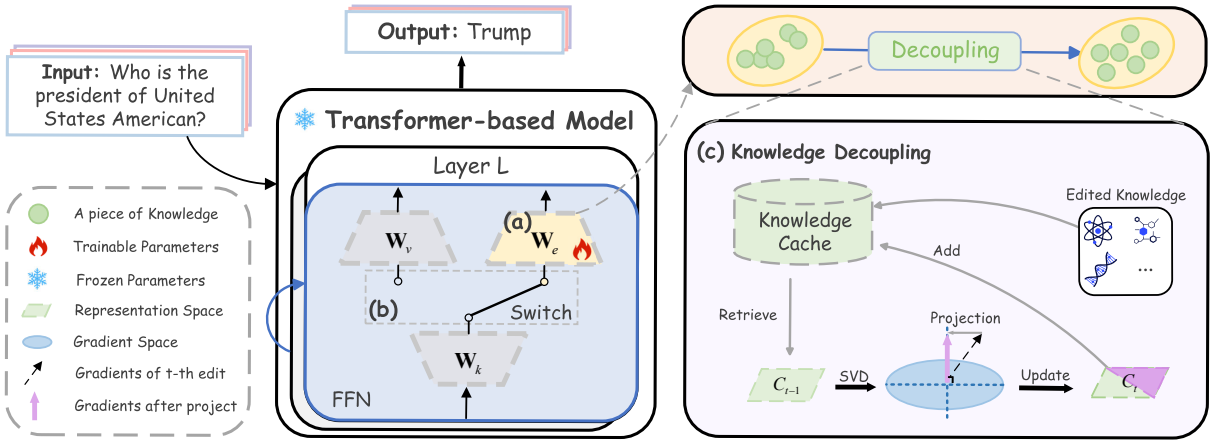

Figure 3: The overview of our proposed KDE. (a) represents the Editable Memory. (b) denotes the Switch Mechanism, which routes queries based on the activation differences between the editable and original memory. (c) refers to Knowledge Decoupling, which employs a knowledge cache to store the representation space of past edits and updates the current edit by projecting its gradient onto the orthogonal space of these historical representations.

Memory for the query x. Specifically, for a given query x, we first compute its activation difference:

$$A(x) = \|h \cdot (W_e - W_v)\|_2, \tag{4}$$

where h is the output by the key matrix W_k, and $\|\cdot\|_2$ denotes the L2 norm. The activation difference $A(x)$ reflects the disparity in responses between Editable Memory and Original Memory for the query. A larger $A(x)$ indicates that the query should enter Editable Memory. To accurately distinguish which memory the query should enter, we define a hinge loss function (Gentile and Warmuth, 1998) L_s:

$$L_s = \min_{W_e} \{\max(0, A(x_i) - \alpha) + \max(0, \beta - A(x_e))\}, \tag{5}$$

where x_i is the input unrelated to editing, and x_e is the input related to editing. The goal of this loss function is to maximize the activation difference for queries unrelated to the edit x_i, ensuring it is below α, while minimizing the activation difference for queries related to the edit x_e, ensuring it is at least β. By default, α and β are set to 5 and 20, respectively.

Finally, the Switch Mechanism dynamically decides which memory the query should enter using a Gate Function $\delta(\cdot)$, defined as:

$$\delta(a) = \begin{cases} 0 & , \text{ if } a \geq \epsilon \\ 1 & , \text{ otherwise} \end{cases}, \tag{6}$$

where $a = A(x)$ is the activation difference, and ϵ is the minimum activation difference across all edit examples, i.e., $A(x_e)_{\min}$. Therefore, the final output of the FFN layer is:

$$FFN(x) = h \cdot (\delta(a) \cdot W_e + (1 - \delta(a)) \cdot W_v), \tag{7}$$

where h is the output by the key matrix W_k.

3.3 Knowledge Decoupling

Figure 5 (a) illustrates the performance variation of the WISE method across 100 to 3,000 edits. It can be observed that after 3,000 edits, the model's average performance drops by 24%, and its proficiency in the edited knowledge significantly decreases. Furthermore, as shown in Figure 1, with increasing edits, the model's ability to recall and comprehend earlier edits deteriorates notably, with reliability decreasing by 36% and generalization ability dropping by 33%. We believe that the underlying cause of this phenomenon lies in the coupling of knowledge, wherein the knowledge from later edits interferes with the knowledge from earlier ones. Inspired by works in continual learning (Zeng et al., 2019; Saha et al., 2021), we propose a knowledge decoupling method to mitigate the issue of knowledge coupling by adjusting the direction of the gradient updates during editing.

First Edit. No restrictions are applied for the first editing example (x_e^1, y_e^1). In other words, the Editable Memory is directly edited. The update process is as follows:

$$W_e \leftarrow W_e - \eta \frac{\partial L}{\partial W_e}, \tag{8}$$

where the loss function is defined as $L = -\log P_{W_e}(y_e|x_e) + L_s$, and η represents the learn-

ing rate. We refer to the Editable Memory after the first edit as W_1 to simplify notation.

After the first edit, we propagate the input x_e^1 through the network to obtain the intermediate variable $h_1 = f(x_e^1 \cdot W_k^\top)$, which represents the input to the Editable Memory W_e. The gradient of the linear layer (represented by each column) lies within the subspace spanned by the input. The detailed proof can be found in Appendix B.1. To further explore the relationship between the input and the gradient, we apply singular value decomposition (SVD) (Businger and Golub, 1969) to h_1:

$$h_1 = U_1 \Sigma_1 V_1^\top, \tag{9}$$

where $U_1 \in \mathbb{R}^{m \times m}$ and $V_1 \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\Sigma_1 \in \mathbb{R}^{m \times n}$ contains the singular values arranged along the main diagonal. To improve computational efficiency, we apply the Eckart-Young Theorem (see Appendix B.2) to approximate h_1 with rank k. Precisely, we control the number of singular values retained, k, by setting a predefined threshold γ, such that the following condition is met:

$$\frac{\|(\Sigma)_k\|_F^2}{\|\Sigma\|_F^2} \geq \gamma, \tag{10}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and Σ_k contains the top k singular values. Subsequently, we define the representation space for the first edit, S_1, as the span of the first k vectors of U_1:

$$S_1 = \text{span}\{u_{1,1}, u_{1,2}, ..., u_{1,k}\}, \tag{11}$$

where the basis vectors $u_{1,i}$ represent the i-th column vector of U_1. When we update the model along the direction of S_1, it has the maximal impact on this knowledge, while updating in the orthogonal subspace of S_1 minimizes the effect, which allows us to decouple the knowledge.

To facilitate the retrieval of the representation space from prior edits in subsequent edits, we design a knowledge cache C, where the basis vectors of the representation space S_1 are stored: $C_1 = \{u_{1,1}, u_{1,2}, ..., u_{1,k}\}$.

Subsequent Edits. For editing the t-th ($t \geq 2$) knowledge (x_e^t, y_e^t), the process begins by retrieving the representation space C_{t-1} from the knowledge cache C, which contains the representations of the first $t-1$ knowledge edits. Next, we project the gradient $\nabla_{W_t} L_t$ of the t-th edit onto the orthogonal space of C_{t-1} before performing the update:

$$\nabla_{W_t} L_t = \nabla_{W_t} L_t - (\nabla_{W_t} L_t) C_{t-1} C_{t-1}^\top. \tag{12}$$

	#Edits	Rel.	Gen.	Loc.	Avg.
w/ L_s	10	1.00	0.96	1.00	0.99
	500	0.98	0.88	0.97	0.94
w/o L_s	2000	**0.93**	**0.81**	**0.89**	**0.88**
w/ L_s	2000	0.70	0.63	0.79	0.71

Table 1: Impact of L_s on Model Performance Across Edit Counts On ZsRE.

After completing the t-th edit, we use its representation space to update the knowledge cache. As with the first edit, we propagate x_e^t through the network to obtain the intermediate variable $h_t = f(x_e^t \cdot W_k^\top)$. To ensure the uniqueness of the basis vectors in the knowledge cache, we need to remove the overlapping components between h_t and the existing content in the cache:

$$\hat{h}_t = h_t - (h_t) C_{t-1} C_{t-1}^\top. \tag{13}$$

Subsequently, we perform SVD on $\hat{h}_t$, yielding $\hat{R}_t = U_t \Sigma_t V_t^\top$, and select the number of singular values k to retain based on the threshold γ:

$$\frac{\|(\hat{h}_t)_k\|_F^2 + \|(h_t) C_{t-1} C_{t-1}^\top\|_F^2}{\|h_t\|_F^2} \geq \gamma. \tag{14}$$

Finally, we add the first k vectors from the unitary matrix U_t to the knowledge cache, yielding the updated cache $C_t = \{C_{t-1}, u_{t,1}, ..., u_{t,k}\}$. For the specific update process, please refer to the pseudocode in Appendix A.2.

3.4 Two-Stage Training

As shown in Table 1, after 500 edits, the model's locality remains high. If the loss function L_s, which optimizes query distinction ability, is removed, the locality remains stable, staying around 90% even after 2000 edits. However, when L_s is retained, we observe a significant drop in model performance once the number of edits reaches 2000 or more. We attribute this phenomenon to the increasing interference between learning new knowledge and improving query distinction ability as the number of edits grows, preventing both tasks from being fully optimized.

To address this issue, we propose a simple yet effective two-stage training strategy. The core idea of this strategy is to gradually reduce the interference between new knowledge editing and query distinction ability enhancement through staged training. This strategy helps maintain stable model performance during long sequence editing and avoids

Method	ZsRE																SelfCheckGPT							
	$N=1$				$N=10$				$N=100$				$N=1000$				$N=1$		$N=10$		$N=100$		$N=600$	
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	PPL.	Loc.	PPL.	Loc.	PPL.	Loc.	PPL.	Loc.
LLaMa2-7B																								
FT-L	0.57	0.52	0.96	0.68	0.48	0.48	0.76	0.57	0.30	0.27	0.23	0.27	0.19	0.16	0.03	0.13	4.41	0.96	12.57	0.71	33.06	0.41	69.22	0.26
FT-EWC	0.96	0.95	0.02	0.64	0.82	0.76	0.01	0.53	0.83	0.74	0.08	0.55	0.76	0.69	0.08	0.51	2.56	0.24	3.63	0.09	**2.10**	0.16	4.56	0.24
MEND	0.95	0.93	0.98	0.95	0.26	0.28	0.28	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.65	0.87	11.01	0.86	10.04	0.88	-	-
ROME	0.85	0.80	0.99	0.88	0.64	0.62	0.75	0.67	0.23	0.22	0.04	0.16	0.01	0.01	0.00	0.01	1.68	0.99	2.04	0.94	94.15	0.05	103.94	0.02
MEMIT	0.84	0.81	0.99	0.88	0.58	0.58	0.85	0.67	0.02	0.02	0.02	0.02	0.04	0.04	0.02	0.03	1.66	1.00	2.36	0.97	76.65	0.05	107.61	0.02
MASS	0.84	0.81	0.99	0.88	0.75	0.72	0.97	0.81	0.76	0.68	0.85	0.76	0.69	0.65	0.62	0.65	1.66	1.00	1.61	0.99	7.18	0.96	13.47	0.94
DEFER	0.68	0.58	0.56	0.61	0.65	0.47	0.36	0.49	0.20	0.12	0.27	0.20	0.03	0.03	0.74	0.27	1.29	0.23	3.64	0.28	8.91	0.19	19.16	0.12
GRACE	0.98	0.08	**1.00**	0.69	0.96	0.00	**1.00**	0.65	0.96	0.00	**1.00**	0.65	**0.97**	0.08	**1.00**	0.68	2.59	**1.00**	9.62	**1.00**	9.44	**1.00**	9.34	**1.00**
WISE	0.98	0.92	**1.00**	0.97	0.94	0.88	**1.00**	0.94	0.90	0.81	**1.00**	0.90	0.77	0.72	**1.00**	0.83	1.91	**1.00**	1.04	**1.00**	1.14	**1.00**	3.12	0.99
KDE	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.96**	**1.00**	**0.99**	**0.99**	**0.92**	**1.00**	**0.97**	0.95	**0.86**	0.94	**0.92**	**1.00**	**1.00**	**1.01**	**1.00**	**1.05**	0.94	**1.44**	0.91
GPT-J-6B																								
FT-L	0.57	0.52	0.96	0.48	0.48	0.76	0.27	0.23	0.55	0.19	0.16	0.03	0.15	0.12	0.05	0.10	6.14	0.92	14.58	0.72	39.25	0.44	81.77	0.31
FT-EWC	0.98	0.97	0.01	0.65	0.58	0.56	0.01	0.38	0.59	0.56	0.01	0.39	0.52	0.46	0.01	0.33	2.31	0.16	3.71	0.10	4.24	0.18	4.89	0.22
MEND	0.96	0.95	0.99	0.97	0.02	0.02	0.02	0.02	0.01	0.01	0.00	0.01	0.00	0.00	0.00	0.00	6.65	0.88	16.82	0.81	21.37	0.74	-	-
ROME	0.99	0.96	0.87	0.94	0.48	0.46	0.12	0.35	0.03	0.02	0.04	0.03	0.03	0.02	0.02	0.02	1.81	0.99	4.62	0.92	99.13	0.04	136.47	0.01
MEMIT	0.99	0.88	0.98	0.95	0.58	0.56	0.20	0.45	0.02	0.02	0.04	0.03	0.04	0.02	0.02	0.03	1.62	**1.00**	7.14	0.91	79.96	0.04	118.43	0.02
MASS	0.99	0.88	0.98	0.95	0.96	0.81	0.97	0.91	0.78	0.72	0.92	0.81	0.81	0.75	0.70	0.75	1.62	**1.00**	2.01	0.99	5.54	0.96	9.73	0.93
DEFER	0.99	0.99	0.39	0.79	0.67	0.51	0.37	0.52	0.01	0.00	0.92	0.31	0.32	0.28	0.25	0.28	3.32	0.38	5.16	0.33	8.43	0.28	17.82	0.17
GRACE	0.99	0.01	**1.00**	0.67	0.97	0.27	**1.00**	0.75	0.98	0.18	**1.00**	0.72	**0.98**	0.09	**1.00**	0.69	2.31	**1.00**	5.64	**1.00**	8.77	**1.00**	8.96	**1.00**
WISE	**1.00**	0.98	**1.00**	0.99	0.92	0.91	**1.00**	0.94	0.86	0.75	**1.00**	0.87	0.70	0.64	0.99	0.78	1.01	**1.00**	1.05	0.99	1.17	**1.00**	4.16	**1.00**
KDE	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.96**	**1.00**	**0.99**	**0.99**	**0.87**	**1.00**	**0.95**	**0.98**	**0.79**	**1.00**	**0.92**	**1.00**	**1.00**	**1.00**	**1.00**	**1.01**	0.99	**1.31**	0.94

Table 2: The overall results for the QA (ZsRE dataset) and Hallucination (SelfCheckGPT dataset) settings are presented. Here, N denotes the number of edits, and "Rel.", "Gen.", "Loc." and "PPL." represent the reliability, generality, locality, and perplexity, respectively. Note that for PPL, smaller values are better, while higher values are preferred for other metrics. Due to space limitations, we use "-" to indicate cases where PPL exceeds 1,000, and use "MASS" stands for "MEMIT-MASS".

performance degradation due to the interference between the two tasks. In the first n edits, we use a composite loss function that includes L_s, allowing the model to improve its query distinction ability while learning new knowledge. In subsequent edits, we remove L_s and focus on enhancing the model's knowledge editing ability:

$$L = \begin{cases} -\log P_{W_e}(y_e|x_e) + L_s & \text{, if } t \leq n \\ -\log P_{W_e}(y_e|x_e) & \text{, otherwise} \end{cases}, \tag{15}$$

where t represents the current number of edits. This strategy leverages the stability of locality during the editing process and stages the training of new knowledge editing ability and query distinction ability. As a result, the model's performance remains stable during long-term editing, avoiding performance degradation caused by interference between the two tasks.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. We utilize two widely used lifelong model editing datasets, ZsRE (Levy et al., 2017) and SelfCheckGPT (Manakul et al., 2023), to train and evaluate the performance of KDE. Specifically, we follow the evaluation methodology of (Wang et al., 2024c) and test KDE's performance in answering context-independent questions (ZsRE) and its ability to mitigate hallucination issues (SelfCheckGPT). Details of these two datasets can be found in Appendix A.3. We evaluate the model's performance on the ZsRE using the metrics of Reliability, Generalization, and Locality defined in Section 2, along with their average values. For the SelfCheckGPT, we follow the evaluation settings from prior research (Wang et al., 2024c), using perplexity (PPL) as the metric for evaluating reliability, with locality assessed in the same manner as for the ZsRE and excluding generalization. [1]

4.2 Experiment Results

Main Results. As shown in Table 2, we observe the following phenomena: (1) Our KDE method outperforms all baseline methods in terms of average performance, achieving the best results. For instance, after 1,000 edits on the LLaMA2 in the zero-shot question-answering task (ZsRE), our method improves upon the state-of-the-art WISE by 11%. Notably, our method demonstrates significant improvements in reliability and generalization, with

[1]The detailed description of baselines and implementation are presented in Appendix A.4 and Appendix A.5.

Method	$N = 2000$				$N = 3000$				$N = 4000$				$N = 5000$			
	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.	Rel.	Gen.	Loc.	Avg.
GRACE	**0.96**	0.03	**1.00**	0.66	**0.96**	0.03	**1.00**	**0.66**	**0.95**	0.02	**1.00**	0.66	**0.95**	0.01	**1.00**	0.65
WISE	0.66	0.63	**1.00**	0.76	0.58	0.56	**1.00**	0.71	0.56	0.55	**1.00**	0.70	0.54	0.53	**1.00**	0.69
KDE	0.93	**0.81**	0.89	**0.88**	0.89	**0.77**	0.87	**0.84**	0.86	**0.74**	0.86	**0.82**	0.81	**0.71**	0.85	**0.79**

Table 3: Scaling to **5K edits** on ZsRE (LLaMA2).

increases of 23% and 19%, respectively. These improvements are attributed to the design of our knowledge decoupling module, which stores the basis vectors of the historical editing representation space in a knowledge cache and constrains the gradient of each edit to its orthogonal space. This alleviates the knowledge coupling issue and significantly enhances the model's reliability and generalization when responding to edited prompts. Additionally, our method maintains locality above 90%, highlighting its ability to distinguish inputs unrelated to the edit prompt.

(2) Among the baselines, GRACE maintains high levels of reliability and locality, but it suffers from a near-complete loss of generalization capability. This is due to the non-parametric nature of its codebook representation, which simply memorizes knowledge without understanding. WISE shows a more balanced performance, but its performance declines significantly as the number of edits increases. This is because its knowledge sharding and merging mechanism merely reduces knowledge density in fixed parameters without addressing the underlying issue of knowledge coupling.

(3) As the number of edits increases, we observe a gradual decline in the average performance of nearly all methods. For instance, the strongest baseline method, WISE, drops from 99% performance after a single edit to 83% after several thousand edits, a decrease of nearly 16%, while our method only experiences an 8% drop. Such a result aligns with our expectations, as an increasing number of edits exacerbates the coupling of knowledge, leading to a significant performance decline. This further highlights the advantages of KDE in lifelong editing scenarios.

(4) KDE's advantages in hallucination mitigation are even more remarkable. KDE maintains perplexities of 1.44 and 1.31, respectively. KDE shows 54% and 68% improvements compared to WISE, respectively. Meanwhile, the locality always remains above 90%.

Scaling to Longer Sequences: 5,000 Edits. As shown in Table 3, KDE consistently outperforms all other methods, with a performance improvement of approximately 15% over the best baseline method, WISE. Notably, as the number of edits increases, the gap between our model and the best baseline method in terms of reliability and generalization continues to widen. After 5,000 edits, reliability and generalization are improved by 50% and 34%, respectively, compared to WISE. This underscores the advantages and potential of KDE in handling extremely long sequence edits.

	Rel.	Gen.	Loc.	Avg.
KDE	0.95	0.86	0.94	0.92
- Knowledge Decoupling	0.75	0.71	0.99	0.82
- Switch Mechanism	0.97	0.88	0.76	0.87

Table 4: Ablation study of Knowledge Decoupling and Switch Mechanism with **1k** edits on ZsRE (LLaMA2).

	Rel.	Gen.	Loc.	Avg.
KDE	0.89	0.77	0.87	0.84
- Two Stage Training	0.70	0.63	0.79	0.71

Table 5: Ablation study of Two Stage Training with **3k edits** on ZsRE (LLaMA2).

5 Analysis

5.1 Ablation Study

We designed a series of ablation experiments to evaluate the impact of each component in the model on performance, including the Switch Mechanism, Knowledge Decoupling, and Two-stage Training. The results are presented in Tables 4 and 5.

First, after removing the switch mechanism, we observed a decrease of approximately 19% in the model's locality, making it more difficult for the model to accurately distinguish whether a query should enter editable memory or original memory. However, the model's accuracy and generalization ability improved, suggesting that removing the mechanism allowed the model to focus more on knowledge-learning tasks without frequent

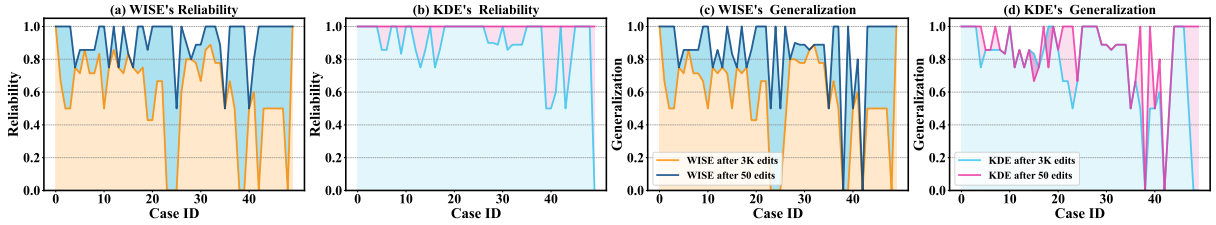

Figure 4: a) and c), as well as b) and d), show the changes in reliability and generalization for the first 50 samples in the WISE and KDE after 50 and 3,000 edits, respectively.

memory switching. Overall, after removing the Switch Mechanism, the model's average performance dropped from 90% to 85%, highlighting the balance that the Switch Mechanism provides between accuracy and locality.

After removing the knowledge decoupling module, the model's average performance decreased by approximately 12%, with reliability and generalization dropping by 27% and 17%, respectively. Without any constraints on the editing knowledge, the coupling between knowledge became severe, leading to interference with early-edited knowledge, thereby affecting the model's memory and understanding of knowledge. This result underscores the critical role of the Knowledge Decoupling module in maintaining knowledge independence and enhancing model stability.

Without two-stage training, the model's performance in long-sequence editing decreased by about 15%. This result demonstrates that two-stage training effectively mitigates the interference between learning new knowledge and query distinction tasks by optimizing them in stages.

In summary, the three core components are crucial to KDE's performance. The switch mechanism ensures that the model can accurately route queries to the appropriate memory. Knowledge decoupling effectively prevents interference between different pieces of knowledge, and two-stage training ensures stability during long-term editing through a phased training strategy.

5.2 Forgetting Resistance Evaluation

As shown in Figure 1, after multiple edits, WISE severely forgets earlier edited knowledge. In the test of the first 50 samples, its reliability decreased by 36%, and its generalization ability decreased by 33%. This issue arises because subsequent edits interfere with earlier ones, leading to knowledge coupling. To address this, KDE extracts the representation space of the edited knowledge and stores it in a knowledge cache. Then, the gradient of the new knowledge is projected into the orthogonal space of this cache for updates, achieving knowledge decoupling. Figure 4 shows the changes in reliability and generalization ability of the first 50 examples after 50 and 3,000 edits, respectively. KDE outperforms WISE after 50 edits, and after 3,000 edits, its reliability and generalization ability only decreased by 11% and 12%, respectively, which is an improvement of 59% and 38% over WISE. The results demonstrate that our approach effectively alleviates the knowledge coupling problem, significantly improving the model's editing performance.

5.3 Memory and Computational Efficiency Scaling with Increasing Edits

On Memory Usage with Increasing Edit Iterations: KDE does not introduce new trainable parameters as the number of edits increases. However, we need to store the basis vectors of previously edited knowledge in the knowledge cache, which incurs some additional memory overhead. Based on our experiments with the LLaMA2-7B model on the ZsRE dataset (1,000 edits), the average number of selected singular values per edit is $k = 9$. Given that each parameter is in Float32 precision, the average memory increase per edit is calculated as $9 \times 11008 \times 32/(8 \times 1024 \times 1024) \approx 0.378$MB (where 11008 is the hidden layer dimension of LLaMA2-7B). After 1,000 edits, the total additional memory usage amounts to $1000 \times 0.378 = 378\,\text{MB} \approx 0.37$GB. For an A100 GPU with 80GB of memory, this overhead is entirely manageable.

On Computational Efficiency with Increasing Edit Iterations: We conducted 1,000 edit experiments using the LLaMA2-7B model on the ZsRE dataset and analyzed the time consumed per 100 edits to roughly evaluate the computational efficiency of KDE as the number of edits grows. As shown in the table 6, overall, the time taken by KDE for

Time for 0-99 edits (min)	100-199	200-299	300-399	400-499	500-599	600-699	700-799	800-899	900-999
29.89	30.17	38.34	40.74	37.23	40.96	43.16	43.01	44.44	43.58

Table 6: The time consumed for every 100 edits using LLaMa2-7B on the ZsRe dataset.

every 100 edits remains relatively stable. For instance, after 600 edits, the duration for each set of 100 edits stabilizes at approximately 43 minutes.

5.4 Why Knowledge Decoupling Works

Suppose the editing of the $(t-1)$-th piece of knowledge x_{t-1} has been completed, and the editable memory W_e is denoted as W_{t-1} at this stage. The basis vectors stored in the knowledge cache collectively span a knowledge space S_{t-1}, and as stated in the appendix B.1, x_{t-1} is contained within this space S_{t-1}. When editing the t-th piece of knowledge x_t, we project its gradient $\nabla_{W_e} L_t$ onto the orthogonal direction of S_{t-1}, enabling W_e to update along the direction orthogonal to S_{t-1}. Let ΔW_{t-1} represent the amount of change in W_e after editing the t-th piece of knowledge, where this change satisfies $\Delta W_{t-1} x_{t-1} = 0$. The editable memory after the t-th edit can be expressed as:

$$W_t = W_{t-1} + \Delta W_{t-1}. \tag{16}$$

For the $(t-1)$-th piece of knowledge, the following derivation holds:

$$\begin{aligned} W_t x_{t-1} &= (W_{t-1} + \Delta W_{t-1}) x_{t-1} \\ &= W_{t-1} x_{t-1} + \Delta W_{t-1} x_{t-1} = W_{t-1} x_{t-1}. \end{aligned} \tag{17}$$

The above derivation shows that after completing the editing of the t-th piece of knowledge, the $(t-1)$-th piece of knowledge remains unaffected, thereby effectively alleviating the problem of knowledge coupling.

6 Related work

Model editing methods can be categorized into several types: constraint-based fine-tuning, locate & edit, meta-learning, and retrieval-based. The locate & edit approach, such as ROME (Meng et al., 2022), uses causal mediation analysis to identify parameters to be modified, followed by direct modification. MEMIT (Meng et al., 2023) extends ROME by enabling batch editing, while AlphaEdit (Fang et al., 2024) introduces continual learning techniques to MEMIT for more stable continuous editing. T-patcher (Huang et al., 2023) performs editing by adding additional new neurons to the FFN. Retrieval-based methods, such as GRACE (Hartvigsen et al., 2023) and WISE (Wang et al., 2024c), introduce extra modules to edit knowledge.

For detailed related works and additional discussions, please refer to Appendix D.

7 Conclusion

In this paper, we emphasize that existing lifelong editing methods often result in significant performance degradation after extensive editing, particularly for the early edited knowledge. We identify that this phenomenon is caused by knowledge coupling. To address this issue, we propose KDE, which mitigates the coupling between knowledge by adjusting the gradient direction during editing and maintains stability during long-sequence editing through a two-stage training strategy. Extensive experiments demonstrate that KDE achieves promising results across multiple datasets and large language models.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China under Grant No. 62032013, No. 62102074, the Science and technology projects in Liaoning Province (No. 2023JH3/10200005), and Natural Science Foundation of Liaoning Province (No. 2024-MSBA-49).

Limitations

Several limitations need to be addressed in future research. First, when the sequence is extended to tens or even hundreds of thousands of edits, a single value matrix may no longer fully comprehend such vast knowledge (leading to excessive knowledge capacity (Allen-Zhu and Li, 2024)). Therefore, exploring the introduction of MoE (Jacobs et al., 1991) mechanisms to accommodate more knowledge could be an interesting research direction. Second, our current focus has primarily been on improving the model's accuracy in answering questions, with less attention given to other capabilities, such as knowledge reasoning. Lastly, we conducted experiments only on decoder-only architecture-based LLMs due to computational

resource constraints while overlooking encoder-decoder architectures. Future research could extend our findings by experimenting with larger-scale models of different architectures.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.3, knowledge capacity scaling laws. *CoRR*, abs/2404.05405.

Vidhisha Balachandran, Hannaneh Hajishirzi, William W. Cohen, and Yulia Tsvetkov. 2022. Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. 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 9818–9830. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Peter A. Businger and Gene H. Golub. 1969. Algorithm 358: singular value decomposition of a complex matrix [f1, 4, 5]. *Commun. ACM*, 12(10):564–565.

Yuchen Cai and Ding Cao. 2024. O-edit: Orthogonal subspace editing for language model sequential editing. *CoRR*, abs/2410.11469.

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.

Arslan Chaudhry, Naeemullah Khan, Puneet K. Dokania, and Philip H. S. Torr. 2020. Continual learning in low-rank orthogonal subspaces. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Qizhou Chen, Taolin Zhang, Dongyang Li, Longtao Huang, Hui Xue, Chengyu Wang, and Xiaofeng He. 2024a. Lifelong knowledge editing for llms with retrieval-augmented continuous prompt learning. *arXiv preprint arXiv:2405.03279*.

Ruizhe Chen, Yichen Li, Zikai Xiao, and Zuozhu Liu. 2024b. Large language model bias mitigation from the perspective of knowledge editing. *CoRR*, abs/2405.09341.

Yingfa Chen, Zhengyan Zhang, Xu Han, Chaojun Xiao, Zhiyuan Liu, Chen Chen, Kuai Li, Tao Yang, and Maosong Sun. 2024c. Robust and scalable model editing for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pages 14157–14172. ELRA and ICCL.

Siyuan Cheng, Ningyu Zhang, Bozhong Tian, Xi Chen, Qingbin Liu, and Huajun Chen. 2024. Editing language model-based knowledge graph embeddings. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17835–17843. AAAI Press.

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.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, and Wangding Zeng. 2024. Deepseek-v3 technical report. *CoRR*, abs/2412.19437.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual

knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5937–5947. 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. *CoRR*, abs/2410.02355.

Claudio Gentile and Manfred K. Warmuth. 1998. Linear hinge loss and average margin. In *Advances in Neural Information Processing Systems 11, [NIPS Conference, Denver, Colorado, USA, November 30 - December 5, 1998]*, pages 225–231. The MIT Press.

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.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024a. 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, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16801–16819. Association for Computational Linguistics.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024b. 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, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 16801–16819. 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.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. 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*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87.

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.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Pengxiang Lan, Haoyu Xu, Enneng Yang, Yuliang Liang, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. 2025a. Efficient and effective prompt tuning via prompt decomposition and compressed outer product. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4406–4421.

Pengxiang Lan, Enneng Yang, Yuting Liu, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. 2025b. Ept: Efficient prompt tuning by multi-space projection and prompt fusion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24366–24374.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. PMET: precise model editing in a transformer. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18564–18572. AAAI Press.

Yan-Shuo Liang and Wu-Jun Li. 2023. Adaptive plasticity improvement for continual learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 7816–7825. IEEE.

Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. 2022. TRGP: trust region gradient projection for continual learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 9004–9017. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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.

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.

Yasumasa Onoe, Michael J. Q. Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can lms learn new entities from descriptions? challenges in propagating injected knowledge. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5469–5485. Association for Computational Linguistics.

Jingyang Qiao, Xin Tan, Chengwei Chen, Yanyun Qu, Yong Peng, Yuan Xie, et al. 2024. Prompt gradient projection for continual learning. In *The Twelfth International Conference on Learning Representations*.

Konstantinos I. Roumeliotis and Nikolaos D. Tselikas. 2023. Chatgpt and open-ai models: A preliminary review. *Future Internet*, 15(6):192.

Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

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, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan,

Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024a. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024b. Detoxifying large language models via knowledge editing. 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 3093–3118. Association for Computational Linguistics.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024c. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 53764–53797. Curran Associates, Inc.

Renzhi Wang and Piji Li. 2024. Lemoe: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 2551–2575. Association for Computational Linguistics.

Weixuan Wang, Barry Haddow, and Alexandra Birch. 2024d. Retrieval-augmented multilingual knowledge editing. 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 335–354. Association for Computational Linguistics.

Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024e. Deepedit: Knowledge editing as decoding with constraints. *CoRR*, abs/2401.10471.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian J. McAuley. 2024f. MEMORYLLM: towards self-updatable large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Zhenyi Wang, Enneng Yang, Li Shen, and Heng Huang. 2024g. A comprehensive survey of forgetting in deep learning beyond continual learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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

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.

Jiakuan Xie, Pengfei Cao, Yuheng Chen, Yubo Chen, Kang Liu, and Jun Zhao. 2024. MEMLA: enhancing multilingual knowledge editing with neuron-masked low-rank adaptation. *CoRR*, abs/2406.11566.

Enneng Yang, Li Shen, Zhenyi Wang, Shiwei Liu, Guibing Guo, and Xingwei Wang. 2023. Data augmented flatness-aware gradient projection for continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5630–5639.

Enneng Yang, Li Shen, Zhenyi Wang, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2025. Revisiting flatness-aware optimization in continual learning with orthogonal gradient projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

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*, pages 10222–10240.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. MELO: enhancing model editing with neuron-indexed dynamic lora. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19449–19457. AAAI Press.

Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. 2019. Continual learning of context-dependent processing in neural networks. *Nat. Mach. Intell.*, 1(8):364–372.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM*, 64(3):107–115.

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

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4862–4876. Association for Computational Linguistics.

Appendix

In the appendix, we provide additional experimental details, theoretical analysis, more comprehensive experimental results and analysis, as well as a more detailed discussion of related work.

- Appendix A: Experimental details.
- Appendix B: Theoretical analysis.
- Appendix C: More results and analyses.
- Appendix D: Additional discussions and more related works.

A Experimental Details

A.1 Evaluation Metrics

Given a dataset $D_e = \{(x_e, y_e)\}$ containing N editing instances, where $x_{e'}$ denotes synonyms of x_e (used to evaluate generalization) and x_{loc} denotes statements unrelated to x_e (used to evaluate locality). The final model after N edits is denoted as f_θ^N. The evaluation metrics of our experiments are as follows:

- **Reliability** measures the model's ability to accurately retain all edited instances:

$$\text{Rel.} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}(f_\theta^N(x_e^t) = y_e^t). \quad (18)$$

- **Generalization** assesses the model's capability to correctly predict examples related to the edited instance x'_e:

$$\text{Gen.} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}(f_\theta^N(x_{e'}^t) = y_e^t). \quad (19)$$

- **Locality** ensures that the model's behavior on inputs unrelated to the edited instances remains unchanged, preserving its original capabilities:

$$\text{Loc.} = \frac{1}{N}\sum_{t=1}^{N} \mathbb{1}(f_\theta^N(x_{loc}^t) = f_\theta^0(x_e^t)). \quad (20)$$

Here, $\mathbb{1}(\cdot)$ denotes the indicator function.

A.2 Algorithms of Knowledge Decoupling

For the pseudocode of knowledge decoupling, please refer to Algorithm 1.

Algorithm 1 Knowledge Decoupling

Input: The initial LLM model f_θ^0, the editable memory W_e, the edit dataset $\mathcal{D}_{\text{edit}} = \{x_e^i, y_e^i\}_{i=1}^N$, the loss function L.
Output: The final LLM model f_θ^N after N edits.
for each edit $(x_e^i, y_e^i) \in \mathcal{D}_{\text{edit}}$, where $i \in (0, N)$ **do**
  # Edit Stage
  $\nabla_w L_i \leftarrow SGD((x_e^i, y_e^i), L, W_e)$
  if M is not empty **then**
    $\nabla_{W_i} L_i \leftarrow project(\nabla_{W_i} L_i, M)$ using Eq. 12
  end if
  $W_e \leftarrow W_e - \eta \nabla_{W_i} L_i$ using Eq. 8
  $\epsilon \leftarrow \min(\epsilon, A(x_i))$
  # Update Knowledge cache
  $h_i \leftarrow forward(x_i, W_e)$
  if M is not empty **then**
    $\hat{h}_i \leftarrow Deduplication(R_i, M)$ using Eq. 13
    $U_i \leftarrow SVD(\hat{h}_i)$
    $k \leftarrow criteria(\hat{h}_i, h_i, \alpha)$ using Eq. 14
  else
    $k \leftarrow criteria(h_i, \alpha)$ using Eq. 10
  end if
  $C_i \leftarrow [C_{i-1}, U_i[0:k]]$
end for
return the post-edit LLM model f_θ^N.

A.3 Model Editing Datasets

ZsRE Dataset. ZsRE (Levy et al., 2017) is a context-free question-answering (QA) dataset generated by BART (Lewis et al., 2020) with human quality control, and it has been widely studied in the field of knowledge editing. Each sample in this dataset consists of an edit example, a synonymous example, and an unrelated example, which are used to evaluate the model's Reliability, Generalization, and Locality, respectively. We partition the dataset following the approach of (Mitchell et al., 2022a), resulting in 163,196 training samples and 19,086 test samples. Notably, among all baseline models, only MEND (Mitchell et al., 2022a) trains the hypernetwork on the training set, while other methods perform editing and evaluation directly on the test set.

SelfCheckGPT Dataset. SelfCheckGPT (Manakul et al., 2023) is a dataset designed to evaluate the effectiveness of model editing methods in mitigating hallucinations in LLMs. It contains highly inaccurate sentences generated by GPT-3 (Brown et al., 2020), with the goal of replacing them with accurate statements from Wikipedia. Compared to ZsRE, SelfCheckGPT presents a greater challenge due to the significantly larger number of tokens that require editing. We preprocess the dataset following the WISE approach, retaining 906 edit examples. To ensure a fair comparison with MEND (Mitchell et al., 2022a), we divide the dataset into

Dataset	N	Pre-Edit (LLaMA/GPT-J/Mistral)
ZsRE	1000	0.36/0.22/0.39 ACC
SelfCheckGPT	600	27.4/84.68/19.4 PPL

Table 7: Dataset Statistics for main results. N is the number of samples. *Pre-edit* is the unedited model's performance on each dataset.

306 training samples and 600 test samples, with the training set exclusively used for MEND. At the same time, all methods are used to edit and evaluate the test set.

The dataset statistics for the main results are provided in Table 7, while specific examples can be found in Table 8.

A.4 Baseline methods

We compare KDE with nine strong baseline methods, which can be categorized into four groups:

- **Fine-tuning based methods**: FT-L (Meng et al., 2022) and FT-EWC (Kirkpatrick et al., 2016). FT-L edits the model by directly fine-tuning the FFN of a specific layer. At the same time, FT-EWC applies regularization to important parameters during continued fine-tuning to mitigate catastrophic forgetting.

- **Locate & edit methods**: ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and MEMIT-MASS (a batch-editing version of MEMIT). These methods treat the FFN as the primary module for storing knowledge and use causal analysis to locate and modify the knowledge stored within the FFN.

- **Meta-learning methods**: MEND (Mitchell et al., 2022a). MEND trains a meta-network on extra data, transforming low-rank gradients from fine-tuning into updated gradients to modify the target layer's parameters.

- **Memory-based methods**: DEFER (Mitchell et al., 2022b), GRACE(Hartvigsen et al., 2023), and WISE (Wang et al., 2024c). DEFER uses a reimplementation of SERAC (Mitchell et al., 2022b) to store editing examples in an external cache; GRACE stores new knowledge by maintaining a codebook; WISE stores new knowledge by copying the value matrix from a specific layer of the FFN and reduces interference between different knowledge through a knowledge sharding and merging mechanism.

In line with prior research(Wang et al., 2024c; Hartvigsen et al., 2023), we chose three widely adopted base models: LLaMa2-7B(Touvron et al., 2023b), GPT-J-6B(Roumeliotis and Tselikas, 2023), and Mistral-7B(Jiang et al., 2023).

A.5 Implementation Details

The hyperparameters for KDE are consistent across different scenarios (question answering and hallucination handling). We use the SGD optimizer with a learning rate η set to 0.5. Following the configurations in GRACE (Hartvigsen et al., 2023) and WISE (Wang et al., 2024c), we set the batch size to 1. The model design includes hyperparameters such as the two boundary parameters $\alpha = 5$ and $\beta = 20$ in the transformation mechanism, as well as the ratio of retained singular values $\gamma = 0.95$ in knowledge decoupling.

For up to 1,000 edits, a two-phase training process is not required, meaning the boundary point c is set to the number of edits. For longer edits ($N \geq 2,000$), c is set to 400 to ensure the effectiveness and stability of long-sequence editing. For LLaMA2-7B, GPT-J-6B, and Mistral-7B, KDE performs editing on the 27th, 21st, and 27th layers. KDE is implemented using the Python libraries PyTorch 2.0.1 [2] and Huggingface Transformers 4.46.3 [3]. All of our experiments were conducted with 8 GPUs, each with a memory of 48 GB.

For all baseline models, we follow the same evaluation and training settings described in (Wang et al., 2024c).

B Theoretical Analysis

B.1 Relationship Between Inputs and Gradients

Based on existing research(Saha et al., 2021; Zhang et al., 2021), the following conclusion can be drawn: the gradient updates of a linear layer always lie within the space spanned by its input.

The forward propagation of a linear layer can be expressed as:

$$h_l = \sigma_l(W_l^\top h_{l-1} + b_l), \tag{21}$$

where the output vector $h_l \in \mathbb{R}^{d_o}$, the input vector $h_{l-1} \in \mathbb{R}^{d_i}$, the weight matrix $W_l \in \mathbb{R}^{d_i \times d_o}$, and σ_l denotes the activation function. The dimensions d_i and d_o correspond to the input and output dimensions. Let the loss function be denoted as L. By

[2] https://pytorch.org/

[3] https://github.com/huggingface/transformers

Dataset	Type	Text
ZsRE	x_e, y_e	What state is Methley located? **Essex**
	x'_e, y'_e	What state has Methley? **Essex**
	x_{loc}, y_{loc}	Who did the voiceover in michael jackson's thriller? **Vincent Price**
SelfCheckGPT	x_e, y_e	This is a Wikipedia passage about william j. flanagan , jr.. William J. Flanagan, Jr. (born October 28, 1945) is an American politician and lawyer who served as the Mayor of Cranston, Rhode Island, from 2003 to 2019. He was first elected in 2003 and was re-elected in 2006, 2010, and 2014. **The Flanagan household consisted of eight children: Patricia Mary, William John Jr., Kathleen, John J., Peter A., Mary Margaret, Anne, and Joseph M. William Flanagan, Sr. was a member of the Massachusetts National Guard.**
	x_{loc}, y_{loc}	The actress was one of 27 arrested yesterday for trespassing during a protest of the DAPL. Actress Shailene Woodley, best known for her starring role **in the Divergent series, was arrested yesterday while protesting the looming implementation of a new oil pipeline project in North Dakota. Woodley is**

Table 8: An editing dataset example from ZsRE and SelfCheckGPT datasets.

applying the chain rule, the gradient of W_l can be derived as:

$$\frac{\partial L}{\partial W_l} = \frac{\partial L}{\partial h_l}\frac{\partial h_l}{\partial W_l} = \left(\frac{\partial L}{\partial h_l} \odot \sigma_l\right) h_{l-1}^T, \quad (22)$$

where $\odot$ represents element-wise multiplication. Furthermore, since $\left(\frac{\partial L}{\partial h_l} \odot \sigma_l\right)$ is equivalent to a vector consisting of real values, denoted as $[b_1, b_2, \ldots, b_{d_o}]^\top$, the gradient of W_l can also be written as:

$$\frac{\partial L}{\partial W_l} = [b_1 h_{l-1}, b_2 h_{l-1}, \ldots, b_{d_o} h_{l-1}], \quad (23)$$

It is evident from the above equation that each column of $\frac{\partial L}{\partial W_l}$ can be represented as the input vector h_{l-1} multiplied by a scalar coefficient b_j ($1 \leq j \leq d_o$). Therefore, in a linear layer, each column of the gradient $\frac{\partial L}{\partial W_l}$ lies within the space spanned by the input vector h_{l-1}.

B.2 Eckart-Young Theorem

For any matrix $A \in \mathbb{R}^{m\times n}$, its optimal low-rank approximation can be obtained by truncating its SVD (Businger and Golub, 1969) to retain the top r singular values along with their corresponding singular vectors. The SVD of matrix A is given by:

$$A = U\Sigma V^\top, \quad (24)$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices consisting of the left and right singular vectors, respectively, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix containing the singular values. By preserving the top r largest singular values and their associated singular vectors, the best rank-r approximation matrix A_r can be constructed as:

$$A \approx A_r = U_r \Sigma_r V_r^\top. \quad (25)$$

The Eckart-Young theorem guarantees that among all rank-r matrices, A_r is the optimal solution that minimizes the approximation error measured by the Frobenius norm. The corresponding error is given by:

$$\|A - A_r\|_F = \sigma_{r+1}, \quad (26)$$

where σ_{r+1} represents the $(r+1)$-th singular value of matrix A.

The Eckart-Young theorem provides an optimal low-rank approximation method based on SVD, effectively ensuring the minimal approximation error in the Frobenius norm.

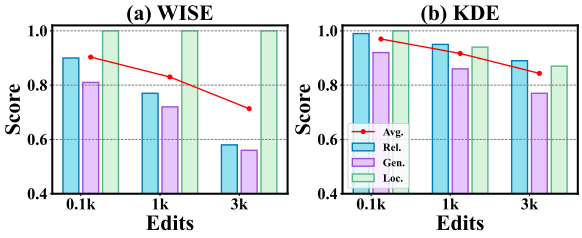

Figure 5: (a) and (b) show the changes in various metrics for both WISE and KDE after edits ranging from 100 to 3,000 on ZsRE, respectively.

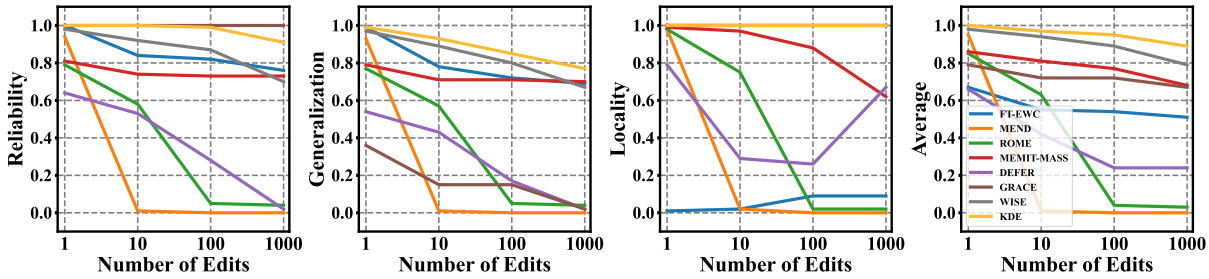

Figure 6: Performance of KDE and baselines on ZsRE using Mistral-7B.

	Case ID	Prompt	Edit Target	Post-Edit Output
I	15	Who is the architect of Toodyay Fire Station?	Kohn Pedersen Fox	Wohn Pedersen Fox ✓✗
		Who was behind the establishment of Toodyay Fire Station?	Kohn Pedersen Fox	Woodohn Pedersen Fox ✓✗
	357	When was USA-64's launch date?	2 March 1992	3 March 1992 ✓✗
		What date was USA-64 launched?	2 March 1992	3 March 1992 ✓✗
II	996	What is the language of Sarah Kazemy?	English	Genman ✗
		What's Sarah Kazemy's language?	English	Genman ✗
	570	What language is Politika written in?	Russian	Polish ✗
		In which language does the monthly magazine Politika report?	Russian	Polish ✗
III	758	What river does La Crosse Rail Bridge cross?	Ohio River	Ohio River ✓
		Over what river does La Crosse Rail Bridge cross?	Ohio River	River ✓✗
	272	Which war was Frank Lucien Hale a part of?	Korean War	Korean War ✓
		In which war did Frank Lucien Hale fight?	Korean War	World war ✓✗
IV	417	The artwork The Forest Fire was by who?	William Etty	University Etty ✓✗
		The work of art The Forest Fire was from whom?	William Etty	William Etty ✓
	977	In which year was the service entry date for JS 7.62?	1963	1961 ✓✗
		In which year did JS 7.62 enter the service?	1963	1963 ✓

Table 9: Failure cases of KDE after 1,000 edits on ZsRE. ✓ and ✗ represent examples of correct and incorrect answers, respectively, while ✓✗ indicates that some tokens are correct. Each case ID corresponds to two prompts, which are used to test the reliability and generalization ability of the model, respectively.

C More Results and Analyses

C.1 The performance of Mistral on ZsRE

The performance of Mistral-7B on the ZsRE dataset is shown in Figure 6. Analyzing the figure, it can be seen that KDE consistently maintains a leading position in comprehensive performance throughout the process from single editing to 1,000 edits. Specifically, its Reliablity and Locality indicators always remain at high levels (both exceeding 90%), and its Generalization performs more stably compared to other Baseline methods. These experimental results are consistent with the test results on the LLaMa2-7b and GPT-J-6B models.

C.2 Case Study

As shown in Table 9, we present the failure cases encountered by KDE after 1,000 edits on the ZsRE dataset. Analyzing the characteristics of these cases is crucial for the future development of lifelong editing technologies. Based on our observations, the error cases can be broadly classified into the following four categories:

I) consists of errors occurring in only a subset of tokens, indicating that the editing operation has not fully converged. Future improvements may involve adjusting the number of iterations and learning rate to alleviate the problem better and enhance the edits' accuracy and consistency.

II) involves errors in the entire output, although these errors are less frequent. We found that these failure cases often involve editing targets that are single words. When a token is predicted incorrectly, it may lead to the entire output deviating from the editing objective.

III) involves cases where the model produces an answer that matches the edit prompt but fails to generalize effectively. For instance, in case ID 758, the model returns "river", which does not fully align with the editing goal. This type of error is relatively common, suggesting substantial room for improving the generalization capability of the current editing methods.

IV) is the opposite of the third, where the model makes an error when responding to the edit prompt but correctly answers the paraphrased prompt. These errors are rare, and we observed that they typically occur when querying dates or names. This situation is due to the limited number of samples involving dates and names, which may also involve repetitions, allowing the model to occasionally answer correctly.

By classifying and analyzing these error cases, we gain a deeper understanding of KDE's limitations in lifelong editing tasks and provide valuable insights for future improvements.

D Detailed Related Works

D.1 Model Editing of LLMs

Model editing is a promising research area, with existing studies exploring various methods for editing large language models (LLMs). These methods can generally be classified into two categories based on whether or not the internal parameters of the model are modified(Zhang et al., 2024; Yao et al., 2023):

Parameter-modifying methods. Methods in this category primarily focus on identifying model parameters associated with specific knowledge and achieving knowledge editing by adjusting these parameters (Meng et al., 2022; Wu et al., 2023; Wang et al., 2024b; Chen et al., 2024b). For example, KN (Dai et al., 2022) attributes knowledge to specific neurons and updates these neurons to edit the model's knowledge. ROME (Meng et al., 2022) utilizes causal mediation analysis to locate the regions of parameters that need modification and updates these regions to perform the model edit. To address the limitation of ROME being capable of only single-step edits, MEMIT (Meng et al., 2023) extends the framework to allow the model to edit multiple knowledge points simultaneously. PMET (Li et al., 2024) further enhances MEMIT by incorporating attention mechanisms to improve the model's editing capabilities.

Recently, O-edit (Cai and Cao, 2024) and AlphaEdit (Fang et al., 2024) methods have explored the application of continual learning (Chaudhry et al., 2020) techniques based on MEMIT to enhance model performance in lifelong editing tasks. However, these approaches face notable limitations. For instance, O-edit requires acquiring a large amount of additional corpora (100,000 pieces of knowledge unrelated to the edited prompts (Cai and Cao, 2024)) to mitigate the negative impact on the general capabilities of the base model. This process is not only time-consuming but also computationally expensive. AlphaEdit ensures knowledge stability during editing by mapping perturbation terms to the null space of knowledge. Although both methods have demonstrated significant results, they still risk disrupting knowledge unrelated to the edits. In contrast, our proposed method maintains editing precision while minimizing the impact on unrelated knowledge, achieving more stable and reliable editing outcomes.

Parameter-preserving methods. These methods avoid directly modifying the model's original parameters, instead employing additional mechanisms to edit or expand the model's knowledge. Based on the specific method analogies, they can be classified into three categories (Zhang et al., 2024): (1) Retrieval-based methods (Chen et al., 2024c; Wang et al., 2024e; Onoe et al., 2023; Wang et al., 2024f,d): These techniques supplement the model with external knowledge repositories, enabling the LLMs to retrieve knowledge from the external database when prompted effectively. For example, IKE(Zheng et al., 2023) edits the model through context learning, adjusting the model's output based on similarity matching without requiring gradient-based adjustments. (2) Meta-learning methods (Mitchell et al., 2022a; Cheng et al., 2024; Tan et al., 2023): These approaches train a meta-network to assist with editing. For instance, MEND (Mitchell et al., 2022a) introduces a meta-network to decouple fine-tuning gradients, enhancing the model's generalization ability. Building on MEND, MALMEN (Tan et al., 2023) uses the meta-network to generate offsets for editing the model and formulates the aggregation of these offsets as a least squares problem, mitigating the offset issues in MEND. (3) Methods of adding extra parameters (Hartvigsen et al., 2023; Wang et al., 2024c; Huang et al., 2023; Wang and Li, 2024; Xie et al., 2024): These approaches freeze the model's parameters and introduce additional trainable parameters to modify the model's output. CaLiNet (Dong et al., 2022) and T-Patcher (Huang et al., 2023) introduce

specific neurons or patches in the model's feed-forward network (FFN) to store and process new knowledge. GRACE (Hartvigsen et al., 2023) maintains a discrete codebook to dynamically store and update knowledge, enabling more stable sequential editing. MELO (Yu et al., 2024) replaces the codebook with Lora (Hu et al., 2021) blocks and explores the use of vector databases for knowledge retrieval. WISE (Wang et al., 2024c) introduces a side memory in one of the FFN layers of LLMs to store knowledge and designs knowledge sharding and merging mechanisms to alleviate knowledge conflicts caused by high knowledge density (Allen-Zhu and Li, 2024) in lifelong editing. Recently, Recipe (Chen et al., 2024a) have shown promising results. However, due to the editing domain's unpredictable nature, this approach faces significant feasibility challenges when deployed in real-world applications.

D.2 Continual learning

Continual learning aims to enable models to continuously acquire new knowledge while retaining previously learned information (Wang et al., 2024a; Wu et al., 2024; Wang et al., 2024g). Among traditional continual learning methods, projection-based strategies (Chaudhry et al., 2020; Saha et al., 2021; Yang et al., 2023, 2025) are most relevant to our work. Specifically, OWM (Zeng et al., 2019) constrains parameter updates to move in directions orthogonal to the gradient space of previous data, thereby reducing interference. Orthog-Subspace (Chaudhry et al., 2020) maps different pieces of knowledge into distinct subspaces, minimizing mutual interference between them. GPM (Saha et al., 2021) analyzes network activations to identify the basis of task subspaces and uses gradient projections to reduce interference between new and old tasks. Building on GPM, DualGPM (Liang and Li, 2023) is an improved version of GPM, which introduces a dual projection mechanism to balance learning between new and old tasks better while mitigating catastrophic forgetting. TRGP (Lin et al., 2022), on the other hand, dynamically and selectively projects gradients in directions orthogonal to old tasks by analyzing inter-task correlations. PGP (Qiao et al., 2024) combines prompt-tuning (Lester et al., 2021; Lan et al., 2025b,a) with gradient projection to ensure the orthogonality of prompt gradients, thus effectively alleviating forgetting. DFGP (Yang et al., 2023, 2025) reinterprets the problem of catastrophic forgetting in orthogonal projection from the perspective of loss landscape flatness, and accordingly improves network flatness to balance plasticity and stability.