

NeuralDB: Scaling Knowledge Editing in LLMs to 100,000 Facts with Neural KV Database

Weizhi Fei^{*} Hao Shi^{*} Jing Xu[†] Jingchen Peng^{*} Jiazheng Li[‡]
 Jingzhao Zhang^{† ¶} Bo Bai[§] Wei Han[§] Zhenyuan Chen[§] Xueyan Niu^{§ ¶}

Abstract

Efficiently editing knowledge stored in large language models (LLMs) enables model updates without large-scale training. One possible solution is Locate-and-Edit (L&E), allowing simultaneous modifications of a massive number of facts. However, such editing may compromise the general abilities of LLMs and even result in forgetting edited facts when scaling up to thousands of edits. In this paper, we model existing linear L&E methods as querying a Key-Value (KV) database. From this perspective, we then propose NeuralDB, an editing framework that explicitly represents the edited facts as a neural KV database equipped with a non-linear gated retrieval module, effectively preserving the general abilities of LLMs. Comprehensive experiments involving the editing of 10,000 facts were conducted on the ZsRE and CounterFacts datasets, using GPT2-XL, GPT-J (6B) and Llama-3 (8B). The results demonstrate that NeuralDB not only excels in editing efficacy, generalization, specificity, fluency, and consistency, but also preserves overall performance across six representative text understanding and generation tasks. Further experiments indicate that NeuralDB maintains its effectiveness even when scaled to 100,000 facts (**50x** more than in prior work).

1 Introduction

Updating the knowledge stored in the parameters of Large Language Models (LLMs) is crucial to refreshing outdated information [1] and integrating domain-specific knowledge to facilitate customization [2]. However, retraining LLMs from scratch is often impractical due to the substantial computational resources and time required. Fine-tuning, while a more feasible approach, can lead to catastrophic forgetting [3, 4]. To address these challenges, *knowledge editing* (KE) methods [5–7] have emerged as promising solutions that enable precise and cost-effective modifications of specific factual associations within LLMs.

Editing a large amount of knowledge without hurting LLMs [8] is an important but challenging task in KE. *Locate-and-edit* (L&E) methods [9–11] are the main solutions to achieve this goal, and these methods are the first to accomplish efficient and scalable editing of massive facts. The L&E paradigm aims to correct the activation of the models for new facts by introducing a linear perturbation Δ to the target parameter \mathbf{W} within the feedforward network (FFN) layers [12]. The new activations are learned from the new facts. Additionally, the activations of “preserved knowledge” that is unrelated to the edited content should be retained. Typically, the preserved knowledge is sampled from Wikipedia.

Preprint. Under review.

^{*}Department of Mathematical Sciences, Tsinghua University; {fwz22, shih22, pj22}@mails.tsinghua.edu.cn

[†]IIS, Tsinghua University; {xujing21, jingzhaoz}@mails.tsinghua.edu.cn

[‡]College of AI, Tsinghua University; foreverlasting1202@outlook.com

[§]Huawei Technologies Co., Ltd.; {baibo8, harvey.hanwei, chenchenyuan, niuxueyan3}@huawei.com

[¶]Corresponding authors.

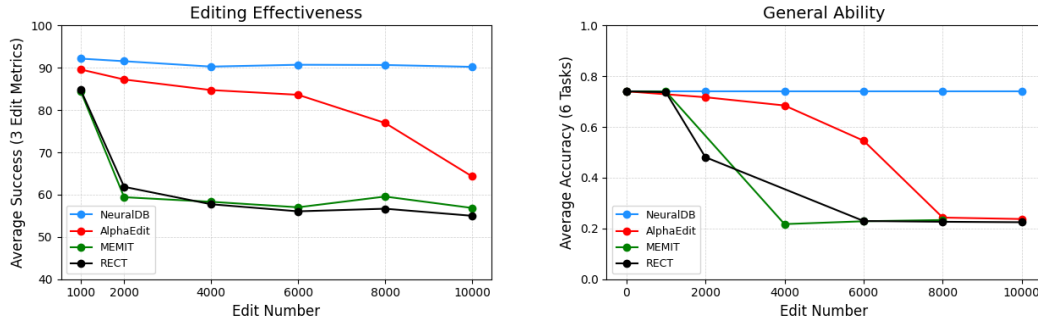


Figure 1: **The proposed NeuralDB scales the number of edited facts up to 10,000 with almost no performance loss.** *Left:* Average of efficacy, generalization, and specificity on the MCF dataset compared with AlphaEdit, MEMIT, and RECT. As the number of edits grows, NeuralDB experiences only a slight decline, whereas the other three methods undergo progressively larger decreases. *Right:* Average performance on six tasks (MMLU, SciQ, Commonsense QA, ARC Challenge, Lambada, WSC273) from the lm-evaluation-harness benchmark. Preserving these general abilities is crucial, as they are acquired through large-scale pre-training on extensive datasets. NeuralDB **perfectly preserves** these general abilities even as the number of edited facts increases, whereas the other three methods lose them almost immediately as the number of edits grows.

The modified parameters Δ are then solved using least squares to ensure that the residuals required are generated for the new facts without affecting the samples to be preserved [9]. Notably, AlphaEdit [11] introduces null space projection to enhance the preservation of the sampled knowledge, effectively maintaining the general capabilities of LLMs. Inspired by their work, we propose a unified framework for these methods, which enables us to further enhance the editing capacity.

Despite the progress, existing methods can only edit hundreds of facts without compromising the models’ general abilities. When editing thousands of facts, the post-edited models usually suffer from a decline in the valuable general abilities developed from extensive training (see Fig. 1). The drop can be attributed to the sampled subset from Wikipedia being insufficient to represent general ability. Additionally, the previously updated information will fade as more facts are edited, because the linear system used by the editing methods is suboptimal in capacity.

In this paper, we demonstrate that most L&E methods, including MEMIT [8], D4S [13], and AlphaEdit [11], can be understood from the perspective of the Key-Value (KV) database. We conceptualize these methods as querying a KV database, wherein a certain hidden state serves as the query to retrieve the corresponding learned residual. Formally, the updated parameters of these methods can be interpreted as a weighted average of the residual matrix associated with the edited facts. We empirically investigate these weights across multiple post-edited models and show that they exhibit an extremely sparse form during inference. Specifically, when inferring on the edited fact, the weights closely resemble a one-hot vector, with only the weight corresponding to the edited fact being non-zero, thereby returning the associated residual. Conversely, when inference is made on unrelated content, the weights are in the form of a zero vector, thereby preventing interference.

In light of this new perspective, we propose a Neural KV Database (NeuralDB) editing framework, which integrates the target FFN layer with a gated non-linear retrieval module that replaces the original linear perturbation Δ . NeuralDB first constructs the neural KV database from the edited facts and then utilizes the retrieval module to integrate this database into LLMs. The gated module returns the most compatible learned residual only when the post-edited models involve the edited facts, therefore, it overcomes the limitations of linearity in the L&E methods and enjoys greater editing capacity. With the gated mechanism, our method can both protect the general ability, as well as reduce the computation and caching costs from sampling Wikipedia. Additionally, NeuralDB is easy to manage for supporting operations such as appending, modifying, and deleting.

To validate the capability of NeuralDB, we conducted comprehensive experiments on three models: GPT-2 XL [14], GPT-J (6B) [15], and Llama-3-Instruct (8B) [16]. These models were evaluated based on their performance for both edited facts and general capabilities. Our results demonstrate that NeuralDB achieves significantly better performance on edited facts, rephrased edited facts, and neighborhood facts, as well as preserves the fluency and consistency on the generation of post-edited models. Moreover, NeuralDB exhibits great scalability — the performances only drop 3% as the edited facts scale from 2000 to 10,000 entries. To further validate the effectiveness of NeuralDB, we

evaluated the post-edited models using widely adopted text understanding and generation tasks. The results indicate that NeuralDB can successfully edit tens of thousands of facts without compromising the quality of the generated text in Llama-3-8B. Extensive scaling experiments on 100,000 edited facts (45x more edited facts than AlphaEdit [11]) demonstrate our method’s ability to maintain high editing precision while fully preserving general language understanding and generation capabilities. This combination of large capacity without loss of generating quality underscores the potential of NeuralDB for trustworthy and customizable deployment of LLMs.

2 Background

Current KE methods typically focus on updating transformer-based LLMs with factual knowledge that can be represented as triples (s, r, o) , where s denotes the subject, r represents the relational predicate, and o is the object. For instance, the fact “The latest World Cup was held in Qatar.” can be represented as (“The latest World Cup”, “was located in”, “Qatar”). Conversely, triples can be transformed into natural language sentences, and we treat these two representations as interchangeable in the sequel. We denote the edited facts as a set of revised tuples $\mathcal{F}^* = \{(s_i, r_i, o_i \rightarrow \hat{o}_i)\}$, where \hat{o}_i represents the target new object that replaces the original o_i . Notably, KE should not compromise the general ability of the model [13], as these abilities are usually developed from extensive pre-training.

As shown in Fig. 3, during inference, the hidden state \mathbf{h}^l of the prediction at the l -th FFN layer of a LLM is computed according to the following recursive form:

$$\mathbf{h}^l = \mathbf{h}^{l-1} + \mathbf{a}^l + \mathbf{m}^l, \quad \mathbf{m}^l = \mathbf{W}_{out}^l \sigma(\mathbf{W}_{in}^l (\mathcal{N}(\mathbf{h}^{l-1} + \mathbf{a}^l))), \quad (1)$$

where \mathbf{a}^l and \mathbf{m}^l are the outputs of the attention block and FFN layer, respectively, \mathbf{W}_{in}^l and \mathbf{W}_{out}^l represent the weight matrices of the l -th FFN layer, respectively, $\mathcal{N}(\cdot)$ represents the layer normalization, and $\sigma(\cdot)$ denotes the activation function. We follow previous research [9, 8] by modeling the FFN layer as operating linear key-value memories as follows:

$$\mathbf{k}^l = \sigma(\mathbf{W}_{in}^l (\mathcal{N}(\mathbf{h}^{l-1} + \mathbf{a}^l))), \quad \mathbf{v}^l = \mathbf{W}_{out}^l \mathbf{k}^l. \quad (2)$$

Then, the textual knowledge (s, r, o) can be linked to the parametric knowledge of the LLM through the activation derived from the inference process. In this context, the key vector \mathbf{k} can be interpreted as encoding the query (s, r) , while the object o is subsequently decoded by the model based on the value vector \mathbf{v} associated with the key [12].

L&E methods aim at adjusting the activation \mathbf{v} on new facts by modifying the parameter. In this paradigm, a learnable perturbation is added to the activation \mathbf{v}^l in the specified layer l using supervised learning, resulting in a new activation $\hat{\mathbf{v}}^l$ that allows the model to generate new answers \hat{o} . Then the perturbation matrix Δ^l should satisfy

$$(\mathbf{W}_{out}^l + \Delta^l) \mathbf{k}_i^l = \hat{\mathbf{v}}_i^l \quad \text{and} \quad (\mathbf{W}_{out}^l + \Delta^l) \mathbf{k}_j^l = \mathbf{v}_j^l \quad (3)$$

where $\hat{\mathbf{v}}_i^l$ corresponds to the new facts and $(\mathbf{k}_j^l, \mathbf{v}_j^l)$ to the sampled knowledge that shall be preserved.

For simplified notation, we denote the parameter to be updated $\mathbf{W}_{out}^l \in \mathbb{R}^{d_2 \times d_1}$ by \mathbf{W} , where d_1 and d_2 represent the dimensions of the intermediate and output layers of the FFN, respectively. To update a large batch of facts, we obtain the key and value matrix by stacking the vectors:

$$\mathbf{K}_1 = [\mathbf{k}_1, \mathbf{k}_2, \dots, \mathbf{k}_m] \in \mathbb{R}^{d_1 \times m}, \quad \hat{\mathbf{V}}_1 = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_m] \in \mathbb{R}^{d_2 \times m}, \quad (4)$$

where m is the number of edited facts, and we provide details on computing \mathbf{k}_i and $\hat{\mathbf{v}}_i$ from the target editing facts in Appendix F. Additionally, we define the residual matrix and residual vectors as

$$\mathbf{R}_1 = \hat{\mathbf{V}}_1 - \mathbf{W} \mathbf{K}_1 \quad \text{and} \quad \mathbf{r}_i = \mathbf{R}_1[:, i]. \quad (5)$$

To preserve the general abilities of the edited model, current methods [8, 13, 11], require the sampling of massive facts from Wikipedia⁶ to construct the matrix \mathbf{K}_0 , which typically consists of 100,000 stacked key vectors representing preserved general knowledge.

⁶<https://huggingface.co/datasets/wikimedia/wikipedia>

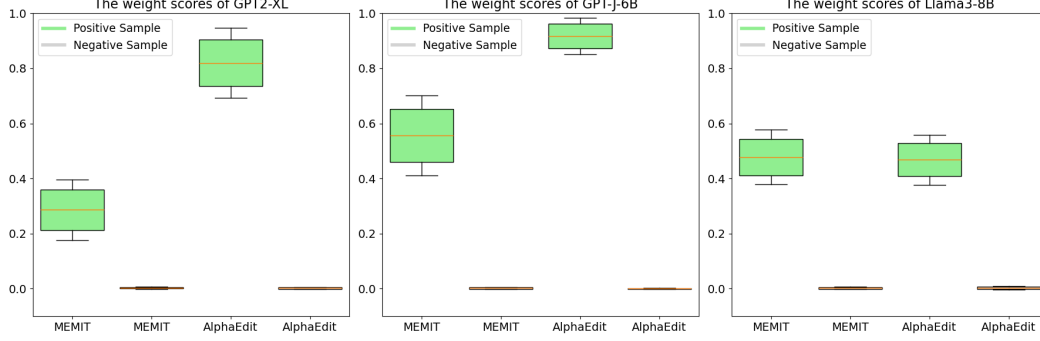


Figure 2: Visualization of weighted scores $\omega = \mathbf{K}_1^T \mathbf{S} \mathbf{k}$ using MEMIT and AlphaEdit for three models. The boxplots are generated from the mean and variance of weight scores, with the center line indicating the mean, boxes showing ± 1 standard deviation, and whiskers ± 1.5 . When inferring the i -th edited fact, ω_i serves as the positive sample, while the remaining elements of ω are negative samples. The high positive score ensures that $\mathbf{R}_1 \omega$ returns the right learned residual.

3 Rethinking locate-and-edit methods with querying Key-Value database

In this section, we demonstrate that most L&E methods can be treated as querying a Key-Value (KV) database. We investigate these editing methods, which support massive knowledge updates, through both theoretical derivation and experimental validation. Specifically, we update all target facts in a single step in executing MEMIT [8] and AlphaEdit [11], which has been shown to effectively mitigate the degradation of general capabilities by D4S [13]. For simplicity, we only discuss the parameter updating over a single FFN layer.

We conclude that the operational mechanism of updating the parameter with Δ_{upd} can be written as

$$(\mathbf{W} + \Delta_{\text{upd}}) \mathbf{k} = \mathbf{v} + \mathbf{R}_1 \omega, \quad \omega = \mathbf{K}_1^T \mathbf{S} \mathbf{k} \in \mathbb{R}^{m \times 1}, \quad (6)$$

where \mathbf{k} and \mathbf{v} are the key and value vectors from the original activation, \mathbf{K}_1^T is the transpose of key matrix computed using edited facts, and \mathbf{S} is the kernel matrix obtained from specific editing methods. $\mathbf{R}_1 \omega$ is the weighted average over \mathbf{R}_1 , with weighted scores being self-similarity $\omega = \mathbf{K}_1^T \mathbf{S} \mathbf{k}$. This means that \mathbf{k} serves as the query to retrieve information from the learned residuals \mathbf{R}_1 . Then we derive the solutions for MEMIT and AlphaEdit, and present their structures.

The objective of MEMIT is the following constrained least squares optimization:

$$\arg \min_{\Delta} \|(\mathbf{W} + \Delta) \mathbf{K}_1 - \hat{\mathbf{V}}_1\|_2^2 + \beta_1 \|\Delta \mathbf{K}_0\|_2^2, \quad (7)$$

where the term $\|\Delta \mathbf{K}_0\|_2^2$ ensures that the updated parameters maintain preserved knowledge. The closed-form solution for MEMIT can be derived as follows:

$$\Delta_{\text{upd}}^{\text{MEMIT}} = \mathbf{R}_1 \mathbf{K}_1^T (\mathbf{K}_1 \mathbf{K}_1^T + \beta_1 \mathbf{K}_0 \mathbf{K}_0^T)^{-1}. \quad (8)$$

Let $\mathbf{S}_1 = (\mathbf{K}_1 \mathbf{K}_1^T + \beta_1 \mathbf{K}_0 \mathbf{K}_0^T)^{-1}$, then the update can be expressed as $\Delta_{\text{upd}}^{\text{MEMIT}} = \mathbf{R}_1 \mathbf{K}_1^T \mathbf{S}_1$.

Similarly, we discuss AlphaEdit, which leverages the null space projection to convert soft constraints for preserved knowledge into hard constraints. The null space projection matrix \mathbf{P} such that $\mathbf{P} \mathbf{K}_0 = \mathbf{0}$ is computed using SVD decomposition over $\mathbf{K}_0^T \mathbf{K}_0$. Then, AlphaEdit constructs $\Delta = \delta \mathbf{P}$ such that $\|\Delta \mathbf{K}_0\|$ approaches zeros and solves the following least squares problem with L2 Norm:

$$\arg \min_{\delta} \|(\mathbf{W} + \delta \mathbf{P}) \mathbf{K}_1 - \hat{\mathbf{V}}_1\|_2^2 + \beta_2 \|\delta \mathbf{P}\|_2^2. \quad (9)$$

The closed-form solution for AlphaEdit is

$$\Delta_{\text{upd}}^{\text{AlphaEdit}} = \mathbf{R}_1 \mathbf{K}_1^T \mathbf{P}^T (\mathbf{P} \mathbf{K}_1 \mathbf{K}_1^T \mathbf{P}^T + \beta_2 \mathbf{I})^{-1} \mathbf{P}. \quad (10)$$

Let $\mathbf{S}_2 = \mathbf{P}^T (\mathbf{P} \mathbf{K}_1 \mathbf{K}_1^T \mathbf{P}^T + \beta_2 \mathbf{I})^{-1} \mathbf{P}$, this can be rewritten as $\Delta_{\text{upd}}^{\text{AlphaEdit}} = \mathbf{R}_1 \mathbf{K}_1^T \mathbf{S}_2$, which is also the weighted average over \mathbf{R}_1 , with $\mathbf{K}_1^T \mathbf{S}_2 \mathbf{k}$ representing the weighted scores ω . In particular,

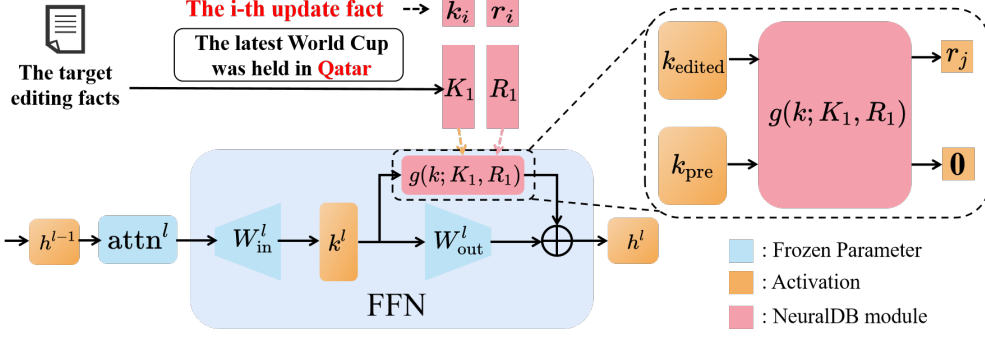


Figure 3: **Overview of NeuralDB editing framework.** We replace the linear system in L&E with a non-linear gated retrieval module in the FFN layer. The key \mathbf{K}_1 and residual matrix \mathbf{R}_1 , defined in Eq. 4 and Eq. 5, are computed to construct the neural KV database. During inference, our non-linear gated function $g(\cdot; \mathbf{K}_1, \mathbf{R}_1)$ only returns the most matched residual \mathbf{r}_j when post-edited models infer one edited fact and involve key vectors $\mathbf{k}_{\text{edited}}$. The function $g(\cdot; \mathbf{K}_1, \mathbf{R}_1)$ reverts zero vector $\mathbf{0}$ when involving the key vector \mathbf{k}_{pre} of preserved knowledge.

AlphaEdit returns $\mathbf{0}$ when \mathbf{k} is from the null space of preserved knowledge \mathbf{K}_0 . Consequently, $\mathbf{S}\mathbf{k} = \mathbf{P}^T(\mathbf{P}\mathbf{K}_1\mathbf{K}_1^T\mathbf{P}^T + \beta_2\mathbf{I})^{-1}(\mathbf{P}\mathbf{k}) = \mathbf{0}$, which effectively preserves the general abilities.

We empirically visualize the weighted scores $\omega = \mathbf{K}_1^T\mathbf{S}\mathbf{k}$ for three post-edited models using MEMIT and AlphaEdit during their inference on new facts. Specifically, we perform KE on 1,000 facts from the Counterfactuals dataset and evaluate the post-edited models on these edited facts. When testing the i -th edited knowledge, we refer to the i -th component of ω_i as the positive sample, while the remaining components are considered negative samples. As shown in Fig. 2, the weighted scores labeled as negative samples are close to 0, while the weighted scores of positive samples are significantly higher, with those for AlphaEdit approaching 1. These empirical results indicate that only the residuals corresponding to the edited facts are activated, while other residual vectors unrelated to the target facts are effectively suppressed. Additional results provided in the Appendix I.2 demonstrate that current methods typically yield $\omega = 0$ when conducting inference on unmodified facts.

4 Method

From the perspective of querying KV database, we can improve the editing capacity and maintain the general abilities of LLMs by explicitly constructing the KV database. To this end, we propose a neural KV database (NeuralDB) editing framework, which serves as a plug-and-play module by replacing the linear perturbation with a non-linear gated retrieval module. Specifically, NeuralDB first computes the key vectors and residual vectors for the target editing facts and constructs a neural KV database. This constructed KV database is then equipped with the non-linear retrieval function for querying the appropriate residual vector during inference on the edited facts.

4.1 Neural KV database

Given the target facts $\mathcal{F}^* = \{(s_i, r_i, o_i \rightarrow \hat{o}_i)\}$, we first compute their key and residual matrix to obtain \mathbf{K}_1 and \mathbf{R}_1 . Below, we formally define them as a neural KV database.

Definition 1 (Neural Key-Value Database). *Given \mathcal{F}^* , the constructed neural KV database can be represented as $(\mathbf{K}_1, \mathbf{R}_1)$, where $\mathbf{K}_1 \in \mathbb{R}^{d_1 \times m}$ and $\mathbf{R}_1 \in \mathbb{R}^{d_2 \times m}$ denote the key matrix and residual matrix of the edited facts as defined in Eq. (4) and Eq. (5). \mathbf{K}_1 and \mathbf{R}_1 serve as keys and values within the database, with $\mathbf{k}_i = \mathbf{K}_1[:, i]$ being associated with $\mathbf{r}_i = \mathbf{R}_1[:, i]$.*

We propose the following sufficient conditions for effective editing.

Editing Rules. *Given a neural KV database $(\mathbf{K}_1, \mathbf{R}_1)$ for target facts \mathcal{F}^* . Let \mathbf{k} denote the key vector obtained through inference on a sequence \mathbf{x} , KE should then adhere to the following rules:*

1. *If the sequence \mathbf{x} is related to the i -th edited fact, then \mathbf{k} matches with \mathbf{k}_j within \mathbf{K}_1 , and the corresponding residual vector \mathbf{r}_j is returned;*

- II. If the sequence \mathbf{x} is not related to any edited facts, then \mathbf{k} does not match any keys within \mathbf{K}_1 , and the zero vector $\mathbf{0}$ is returned.

Rule I ensures the effective updating of target facts, while Rule II ensures the preservation of knowledge unrelated to the edited facts. To more strictly realize these rules, we propose the following non-linear retrieval function.

4.2 Nonlinear gated retrieval module

Given a neural KV database $(\mathbf{K}_1, \mathbf{R}_1)$ constructed from target edited facts, we propose the following function, which returns the residual with maximal similarity to involved the target edited facts:

$$g(\mathbf{k}; \mathbf{K}_1, \mathbf{R}_1) = \mathbf{r}_j * \overbrace{\mathbf{1}_{\cos(\mathbf{k}, \mathbf{k}_j) > \gamma}}^{\text{Gate}}, j = \arg \max \cos(\mathbf{k}, \mathbf{k}_i), \quad (11)$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity, $\mathbf{1}$ represents the indicator function, and γ is the parameter controlling the gating mechanism. We employ cosine similarity because it provides good interpretability, facilitating the easy setting of γ .

As illustrated in Fig. 3, the nonlinear function is integrated into the target FFN layer as follows:

$$\mathbf{v}^l = \mathbf{W}^l \mathbf{k}^l + g(\mathbf{k}^l; \mathbf{K}_1, \mathbf{R}_1). \quad (12)$$

When post-edited models utilize preserved knowledge, the involved key vectors \mathbf{k} typically exhibit low similarity with all key vectors \mathbf{k}_i in the matrix \mathbf{K}_1 . Therefore, this additional module will not provide the perturbation, ensuring that the original model’s workflow remains same. Therefore, the general ability of pre-edited LLMs can successfully be preserved. In contrast, when post-edited models encounter revised facts, our retrieval function can identify and recall the most closely related residual vector, because of the high similarity between the keys. This precise retrieval mechanism effectively updates the specified parametric knowledge.

Another significant advantage of our proposed framework is its ease of deployment. Current L&E methods typically update parameters through optimization, which poses challenges for incremental updates or reverting modified facts. In contrast, our framework maintains the neural KV database that corresponds specifically to the target facts. This design facilitates addition, deletion, and modification of edited facts within the model, thereby improving the flexibility of the editing process.

Single layer versus multi-layer editing We just implement our module in one single FFN layer. Although previous L&E methods typically employ multi-layer strategies, we find this strategy offers limited performance gains. We provide a related discussion in Appendix H.

Additional memory usage and computation time The introduction of additional parameters in the NeuralDB module incurs increased computation and storage requirements. Specifically, the space complexity is $O((d_1 + d_2) \times m)$ for the storage of matrices $\mathbf{K}_1 \in \mathbb{R}^{d_1 \times m}$ and $\mathbf{R}_1 \in \mathbb{R}^{d_2 \times m}$, where m denotes the number of facts. When editing 10,000 facts with Llama 3 8B (Instruct), the additional parameter size amounts to 150M, which constitutes approximately 2.2% of the original model’s size. The evaluation time for 10,000 facts in Counterfact dataset increases only by 1.5% compared to the original model. Detailed information regarding additional memory usage and computation time across various models is provided in Appendix G.

5 Experiments

In this section, we present a comprehensive evaluation of our method for mass KE. For fair comparison, we mostly follow the experimental setups in AlphaEdit [11] to benchmark our method. Specifically, we examine whether the edited models can effectively balance mastery of the edited facts with retention of their general capabilities. Detailed implementation of the NeuralDB editing framework is provided in Appendix E. Additionally, ablation studies are included in Appendix H, where we explore the impact of layer selection, the setting of the parameter γ , and the multi-layer configurations.

Table 1: Comparison of NeuralDB with existing methods on the massive KE task. We evaluated the performance of editing 2,000 facts, with results for Pre-edited, FT, MEND, InstructEdit, MELO, and ROME sourced from Fang et al. [11]. For 10,000 facts, the results for MEMIT, RECT, AlphaEdit, and NeuralDB are denoted using the arrow (\rightarrow) notation. The best results for both 2,000 and 10,000 facts are highlighted in bold, respectively.

Method	Model	Counterfact					ZsRE		
		Efficacy \uparrow	Generalization \uparrow	Specificity \uparrow	Fluency \uparrow	Consistency \uparrow	Efficacy \uparrow	Generalization \uparrow	Specificity \uparrow
Pre-edited		7.9	10.6	89.5	635.2	24.1	37.0	36.3	31.9
FT	LLaMA3	83.3	67.8	46.6	233.7	8.8	30.5	30.2	15.5
MEND		63.2	61.2	45.4	372.2	4.2	0.9	1.1	0.5
InstructEdit		66.6	64.2	47.1	443.9	7.3	1.6	1.4	1.0
MELO		65.3	58.6	63.4	609.0	32.4	25.2	24.1	30.4
ROME		64.4	61.4	49.4	449.1	3.3	2.0	1.8	0.7
MEMIT		63.5 \rightarrow 63.4	62.8 \rightarrow 56.6	52.0 \rightarrow 50.55	466.6 \rightarrow 460.4	6.5 \rightarrow 6.5	36.7 \rightarrow 0.1	32.9 \rightarrow 0.1	19.1 \rightarrow 1.5
RECT		64.2 \rightarrow 60.0	62.5 \rightarrow 53.9	58.9 \rightarrow 51.2	502.8 \rightarrow 399.1	12.9 \rightarrow 1.6	86.8 \rightarrow 0.0	82.3 \rightarrow 0.0	31.9 \rightarrow 0.0
AlphaEdit		99.1 \rightarrow 75.8	94.0 \rightarrow 63.1	68.6 \rightarrow 54.0	622.7 \rightarrow 417.8	32.8 \rightarrow 7.0	94.4 \rightarrow 90.5	91.3 \rightarrow 85.9	32.6 \rightarrow 30.3
NeuralDB		99.9 \rightarrow 99.2	86.6 \rightarrow 85.9	88.2 \rightarrow 85.6	632.7 \rightarrow 631.02	32.9 \rightarrow 32.6	96.3 \rightarrow 95.9	92.0 \rightarrow 91.0	31.9 \rightarrow 31.8
Pre-edited		16.2	18.6	83.1	621.8	29.7	26.3	25.8	27.4
FT	GPT-J	92.2	72.4	43.4	297.9	6.7	72.4	68.9	19.7
MEND		46.2	46.2	53.9	242.4	3.9	0.7	0.7	0.5
InstructEdit		50.6	51.7	56.3	245.9	4.2	0.9	0.9	0.7
MELO		78.3	60.5	66.8	610.8	24.3	82.2	32.9	26.7
ROME		57.5	54.2	52.1	589.4	3.2	56.4	54.7	9.9
MEMIT		98.6 \rightarrow 48.8	95.4 \rightarrow 49.3	66.1 \rightarrow 51.9	557.8 \rightarrow 281.5	36.5 \rightarrow 5.1	90.5 \rightarrow 0.2	84.7 \rightarrow 0.1	30.9 \rightarrow 0.2
RECT		98.8 \rightarrow 76.3	86.3 \rightarrow 70.6	74.4 \rightarrow 54.9	618.1 \rightarrow 517.3	41.2 \rightarrow 25.4	96.6 \rightarrow 53.5	91.5 \rightarrow 49.6	29.0 \rightarrow 21.9
AlphaEdit		99.8 \rightarrow 91.6	96.3 \rightarrow 79.6	76.2 \rightarrow 60.3	618.5 \rightarrow 517.8	41.9 \rightarrow 6.9	99.7 \rightarrow 94.2	95.9 \rightarrow 86.1	28.8 \rightarrow 22.5
NeuralDB		99.7 \rightarrow 99.1	94.6 \rightarrow 93.2	80.0 \rightarrow 75.7	619.8 \rightarrow 620.0	41.4 \rightarrow 41.3	99.2 \rightarrow 98.2	95.9 \rightarrow 95.0	27.5 \rightarrow 27.0
Pre-edited		22.2	24.3	78.5	626.6	31.9	22.2	31.3	24.2
FT	GPT2-XL	63.6	42.2	57.1	519.4	10.6	37.1	33.3	10.4
MEND		50.8	50.8	49.2	407.2	1.0	0.0	0.0	0.0
InstructEdit		55.3	53.6	53.3	412.6	1.1	3.5	4.3	3.2
ROME		54.6	51.2	52.7	366.1	0.7	47.5	43.6	14.3
MELO		72.6	53.6	63.3	588.6	23.6	93.5	45.3	23.5
MEMIT		93.0 \rightarrow 58.5	83.3 \rightarrow 55.8	58.9 \rightarrow 56.1	481.8 \rightarrow 496.2	23.2 \rightarrow 8.1	74.4 \rightarrow 3.5	66.9 \rightarrow 2.8	25.87 \rightarrow 2.07
RECT		91.8 \rightarrow 86.9	79.5 \rightarrow 69.5	64.0 \rightarrow 55.0	482.1 \rightarrow 517.8	20.3 \rightarrow 10.9	82.6 \rightarrow 27.5	74.7 \rightarrow 25.1	24.6 \rightarrow 13.1
AlphaEdit		99.4 \rightarrow 92.2	93.8 \rightarrow 76.5	65.6 \rightarrow 56.5	584.0 \rightarrow 580.9	37.9 \rightarrow 29.6	93.2 \rightarrow 57.1	83.5 \rightarrow 47.5	25.3 \rightarrow 13.5
NeuralDB		99.8 \rightarrow 99.1	97.2 \rightarrow 95.7	74.1 \rightarrow 70.9	621.5 \rightarrow 619.9	42.2 \rightarrow 41.7	96.3 \rightarrow 94.6	92.8 \rightarrow 91.0	25.0 \rightarrow 24.2

Models and methods We select three representative LLMs, including GPT-2 XL [14], GPT-J (6B) [15], and Llama3 (8B) [16]. We compare our method with the following KE methods, including Fine-Tune (FT) [8], MEND [6], ROME [9], MEMIT [8], MELO [17], RECT [18], and AlphaEdit [11]. We provide a detailed introduction on these baselines and models in Appendix C.

Datasets for knowledge editing To evaluate KE methods, we utilize two widely recognized benchmarks: the Counterfact dataset [9] and the ZsRE dataset [19]. Consistent with previous research [9, 11], we employ the following evaluation metrics: *efficacy* (success of edited facts), *generalization* (success of paraphrased facts), *specificity* (success of neighboring facts), *fluency* (generation entropy), and *consistency* (reference score). Detailed explanations of the datasets and metrics are provided in Appendix D.

Datasets for general ability We assess the general capabilities of the edited LLMs using the following typical datasets, SciQ [20] (science question answering), MMLU [21] (massive multitask language understanding), Commonsense QA [22] (commonsense knowledge understanding), ARC Challenge [23] (challenge task requiring reasoning), WSC273 [24] (coreference resolution), Lambada [25] (predict the endings of text passages). Datasets such as SciQ, MMLU, and Commonsense QA primarily evaluate knowledge-based question answering, focusing on the models’ ability to understand and retain factual information. In contrast, the ARC Challenge, WSC273, and Lambada are designed to assess capabilities beyond mere knowledge memory, such as reasoning and text generation. Detailed information regarding these datasets is provided in Appendix D.

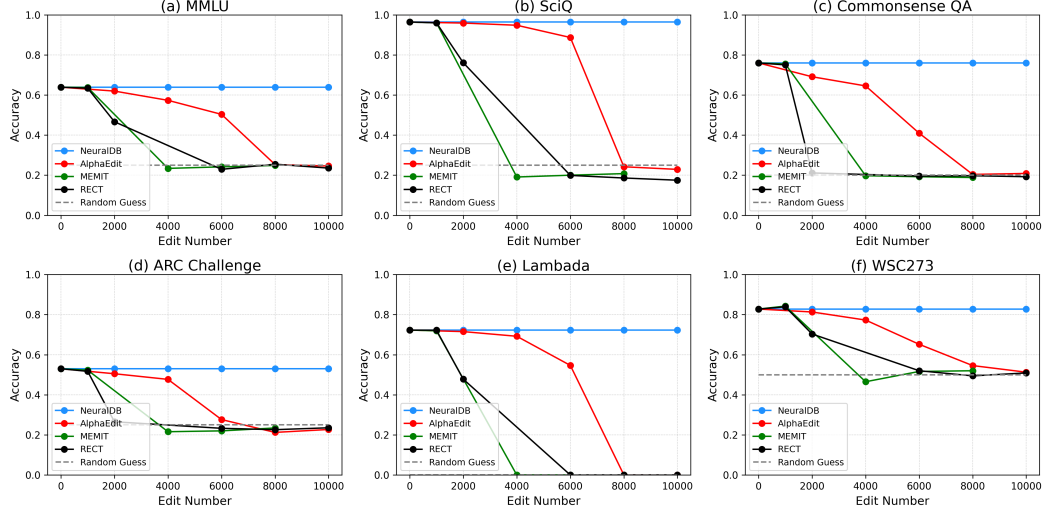


Figure 4: Results of general abilities after massive editing. The performance of NeuralDB is compared with baseline methods, including MEMIT, RECT, and AlphaEdit. The random guessing baseline for multi-choice datasets is indicated by dashed lines. NeuralDB demonstrates exceptional preservation of general abilities when the number of edits scales up.

5.1 The editing effectiveness of post-edited models

We evaluated the performance after editing 2,000 facts and also included the results of 10,000 facts in parentheses for MEMIT, RECT, AlphaEdit, and NeuralDB. The results are presented in Table 1. Our method achieves superior performance on all metrics in three models and two datasets. In particular, when scaling the number of edited facts from 2,000 to 10,000, our method exhibits only a negligible performance drop, in contrast to the substantial degradation observed in the baselines.

Our method demonstrates exceptional efficacy across various scenarios. Even after editing 10,000 facts, our post-edited models maintain strong efficacy and specificity. In contrast, all baseline methods exhibit a significant decline in both efficacy and specificity metrics. This highlights our method’s excellent scalability in effective editing.

In terms of specificity and fluency metrics, our post-edited models demonstrate nearly identical effectiveness to the pre-edited models, representing a substantial improvement over the baselines on the CounterFacts dataset. Furthermore, our method ensures that the generated text maintains coherence with a high degree of consistency. This indicates that our method can preserve the memory of unmodified facts while maintaining its generative capabilities.

5.2 The general abilities of post-edited models

We assessed post-edited models with various configurations, evaluating their performance across 2,000, 4,000, 6,000, 8,000, and 10,000 facts, as depicted in Fig. 4. The evaluation was conducted on lm-evaluation-harness [26]. The results show that our method effectively edits a large number of facts without compromising the general abilities of the models across various tasks. In contrast, existing L&E methods struggle with 4,000 facts editing and exhibit a rapid decline in general abilities as the number of edited facts increases. Notably, these baseline methods achieve favorable results on the SciQ dataset, likely due to the dataset’s content being well-represented in Wikipedia and thus captured by the sampled K_0 . However, their performance deteriorates on other tasks, highlighting the limitations of relying on Wikipedia-sampled K_0 . Our method, which directly incorporates the gated mechanism, offers a more precise and effective approach compared to approximations derived from Wikipedia. For results on more models, please see Appendix I.

5.3 Scaling up the number of edited facts

We further examine the scalability of the NeuralDB when applied to an extensive volume of knowledge. To obtain a sufficiently large set of facts for this investigation, we utilized the training set of

the ZsRE dataset for model editing. The results for the Llama3 8B (Instruct) model are presented in Table 2. These results demonstrate that the performance of NeuralDB remains highly stable as the number of edited facts increases from 10,000 to 100,000 with only marginal degradation observed. In evaluations of the model’s general ability, we find that scaling up the number of edited facts to 100,000 did not harm the general ability performance and led to a 0.7% improvement in the benchmark datasets. This underscores the superior scalability of our framework.

Table 2: Editing accuracy and the post-edited model’s general performance of NeuralDB on Llama 3 (8B) when editing extremely large sets of facts.

Number of edits	0k	10k	20k	30k	40k	50k	60k	70k	80k	90k	100k
Efficacy (\uparrow)	37.0	96.9	96.6	96.6	96.4	96.1	96.0	95.9	95.8	95.6	95.5
Generalization (\uparrow)	36.3	91.4	91.4	91.2	91.0	90.7	90.6	90.6	90.5	90.4	90.2
Specificity (\uparrow)	31.9	35.1	35.3	35.2	35.2	35.2	35.2	35.2	35.1	35.1	35.1
MMLU ⁷ (\uparrow)	56.2	56.2	56.2	56.2	56.2	56.2	56.2	56.9	56.9	56.9	56.9

6 Related Work

6.1 Knowledge editing through parameter modification

Locate-and-edit The L&E paradigm is derived from casual trace experiments [9], suggesting that the factual memory of the Transformer models is primarily associated with the FFN layers [12], therefore Transformers can be understood using associative memory [27]. ROME [9] is proposed to edit the factual memory of the models by modifying the parameter of the target FFN layer. MEMIT [8] extends ROME to support multi-layer and batch editing versions, allowing the editing of thousands of factual knowledge. To address the challenge of post-edited models losing their general capabilities [28, 29], several solutions were proposed, including the dumping of sequential editing caches [13], null space projection [11], and regularization of the weights [30] and singular values [31].

Hypernetwork KE [32] trains a lightweight biLSTM-MLP editor to convert a single-sample gradient into a low-rank weight delta. MEND [33] factorizes two-layer gradients into rank-1 vectors and feeds them through a shared MLP, allowing memory-frugal batch edits. InstructEdit [34] prepends '[Task][Description][Input]' prompts so that gradients self-cluster, allowing one low-rank editor to perform reliable updates in diverse OOD tasks. All these approaches require additional fine-tuning of the hypernetwork with large datasets, incurring considerable computational overhead.

6.2 Knowledge editing without parameter modification

External module This line of work with external memory caches starts with SERAC [35], which augments a frozen model with explicit retrieval memory, a scope classifier, and a lightweight counterfactual Seq2Seq generator so that edits are applied by lookup rather than gradient updates. T-Patcher [36] injects a sparse “key–value–bias” triplet per error into the final FFN layer, ensuring each patch fires only on its intended trigger. GRACE [37] stores erroneous hidden states as discrete keys in a dynamic codebook whose values overwrite selected Transformer layers whenever the current activation falls inside an ϵ -ball, enabling millisecond-scale, thousand-edit hot fixing with $< 0.4\%$ extra parameters. MELO [38] similarly uses a hidden-state-indexed database to activate low-rank, per-edit adapter blocks on demand. MindBridge [39] encodes each edit as a standalone “memory modality” that can be shared across future model versions, preventing knowledge loss when the base LLM is upgraded. Similarly, the external modules of these methods require a large number of datasets for fine-tuning, which leads to high costs.

Prompt-based approaches Recent studies utilize prompt engineering to facilitate efficient KE. For example, MemPrompt [40] and IKE [41] embed updated knowledge into prompts to leverage in-context learning. For multi-hop QA tasks, MQUAKE [42] introduces a benchmark to evaluate

⁷The MMLU results are evaluated by the AlphaEdit project rather than the lm-evaluation-harness. Although they use different metrics, both sets of results reflect the general capabilities of the LLMs.

KE performance. MeLLO [42] stores edited facts externally and iteratively prompts the model to yield answers consistent with the updates. PokeMQA [43] improves retrieval and answer accuracy by decomposing multi-hop questions via prompts. RAE [44] retrieves refined facts and enhances the language model through in-context learning using a knowledge graph. To address multilingual KE, ReMaKE [45] integrates newly retrieved multilingual knowledge into prompts for better adaptation.

7 Conclusion

In this paper, we introduce NeuralDB, a scalable knowledge editing framework designed to construct a neural KV database from edited facts and integrate it into the target FFN layer within LLMs using a non-linear gated function. This integration ensures that the general capabilities of LLMs are preserved. The neural database is designed to be easily maintained, facilitating efficient addition and modification of edited facts within the models. We conducted comprehensive experiments across various LLMs to validate the effectiveness of our framework. Our results on the ZsRE and CounterFacts datasets, utilizing GPT2-XL, GPT-J (6B), and Llama-3 (8B), demonstrate that NeuralDB editing can effectively modify hundreds of thousands of facts without degrading the quality of generated text. Additionally, our findings from six generic text understanding and generation tasks confirm that our method preserves the general abilities of LLMs unrelated to the target edited facts. These results highlight the robustness and scalability of NeuralDB editing, positioning it as a valuable tool for enhancing the adaptability and accuracy of LLMs in diverse applications.

References

- [1] Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. Is your llm outdated? evaluating llms at temporal generalization. *arXiv preprint arXiv:2405.08460*, 2024.
- [2] Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36:5539–5568, 2023.
- [3] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.
- [4] Zorik Gekhman, Gal Yona, Roei Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. Does fine-tuning llms on new knowledge encourage hallucinations? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7765–7784, 2024.
- [5] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024.
- [6] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *ICLR*. OpenReview.net, 2022.
- [7] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *CoRR*, abs/2305.12740, 2023.
- [8] Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *ICLR*. OpenReview.net, 2023.
- [9] Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=-h6WAS6eE4>.
- [10] Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. PMET: precise model editing in a transformer. In *AAAI*, pages 18564–18572. AAAI Press, 2024.

- [11] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=HvSytvg3Jh>.
- [12] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL <https://aclanthology.org/2021.emnlp-main.446/>.
- [13] Xiusheng Huang, Jiayang Liu, Yequan Wang, and Kang Liu. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=xjXYgdFM5M>.
- [14] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [15] Ben Wang. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- [16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [17] Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, pages 19449–19457. AAAI Press, 2024.
- [18] Xiaojie Gu, Guangxu Chen, Shuliang Liu, Jungang Li, Aiwei Liu, Sicheng Tao, Junyan Zhang, and Xuming Hu. Editing large language models via adaptive gradient guidance. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM)*, 2025.
- [19] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342. Association for Computational Linguistics, 2017.
- [20] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin, editors, *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL <https://aclanthology.org/W17-4413/>.
- [21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=d7KBjmI3GmQ>.
- [22] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL <https://aclanthology.org/N19-1421/>.
- [23] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

- [24] Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. A surprisingly robust trick for the Winograd schema challenge. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4837–4842, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1478. URL <https://aclanthology.org/P19-1478/>.
- [25] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL <https://aclanthology.org/P16-1144/>.
- [26] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL <https://zenodo.org/records/12608602>.
- [27] Xueyan Niu, Bo Bai, Lei Deng, and Wei Han. Beyond scaling laws: Understanding transformer performance with associative memory. *arXiv preprint arXiv:2405.08707*, 2024.
- [28] Qi Li, Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Xinglin Pan, and Xiaowen Chu. Should we really edit language models? on the evaluation of edited language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=m0DS400mSY>.
- [29] Cheng-Hsun Hsueh, Paul Kuo-Ming Huang, Tzu-Han Lin, Che Wei Liao, Hung-Chieh Fang, Chao-Wei Huang, and Yun-Nung Chen. Editing the mind of giants: An in-depth exploration of pitfalls of knowledge editing in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9417–9429, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.550. URL <https://aclanthology.org/2024.findings-emnlp.550/>.
- [30] Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.934. URL <https://aclanthology.org/2024.emnlp-main.934/>.
- [31] Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=bfI8cp8qmk>.
- [32] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*, 2021.
- [33] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.
- [34] Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. Instructedit: Instruction-based knowledge editing for large language models. *arXiv preprint arXiv:2402.16123*, 2024.
- [35] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.

- [36] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*, 2023.
- [37] Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.
- [38] Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457, 2024.
- [39] Shuaike Li, Kai Zhang, Qi Liu, and Enhong Chen. Mindbridge: Scalable and cross-model knowledge editing via memory-augmented modality. *arXiv preprint arXiv:2503.02701*, 2025.
- [40] Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve GPT-3 after deployment. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.183. URL <https://aclanthology.org/2022.emnlp-main.183/>.
- [41] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=hsjQHAM8MV>.
- [42] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=0hTPJBnncc>.
- [43] Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. PokeMQA: Programmable knowledge editing for multi-hop question answering. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8069–8083, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.438. URL <https://aclanthology.org/2024.acl-long.438/>.
- [44] Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. Retrieval-enhanced knowledge editing in language models for multi-hop question answering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM ’24*, page 2056–2066, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679722. URL <https://doi.org/10.1145/3627673.3679722>.
- [45] Weixuan Wang, Barry Haddow, and Alexandra Birch. Retrieval-augmented multilingual knowledge editing. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 335–354, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.21. URL <https://aclanthology.org/2024.acl-long.21/>.

A Limitation

Although NeuralDB can effectively edit thousands of pieces of knowledge, the corresponding storage overhead grows linearly with the number of edited entries. This may lead to additional memory usage, especially when editing a large amount of knowledge. For instance, after editing 100,000 pieces of knowledge for LLama3 8B, the additional memory usage reached 20% of the original model.

B Border impact

As information evolves rapidly in society, LLMs must also adapt swiftly. Pre-training a new LLM requires a substantial amount of time and resources. Knowledge editing enables models to update information and enhance their domain-specific knowledge. Our approach significantly extends the upper limit of the number of edits that can be made without compromising the model’s performance. This advancement has the potential to drive broader applications of knowledge editing.

The wider impacts include positive contributions to society. By improving the efficiency of knowledge updates, models can adapt more quickly to changing environments and information. This not only benefits research and education, but also provides up-to-date information support in critical fields such as healthcare and justice, promoting scientific and timely decision making. Furthermore, the proliferation of knowledge editing will encourage interdisciplinary collaboration, allowing experts from different fields to share and integrate knowledge more effectively to address complex societal issues.

As the capabilities for knowledge editing expand, it is imperative to address the potential ethical and social implications. Ensuring the accuracy and credibility of information is essential to prevent the dissemination of misinformation. Furthermore, a transparent editing process and traceability are crucial for building public trust and ensuring responsible use of technology. The proposed NeuralDB editing framework, explicitly derived from updated knowledge, facilitates interpretable editing. Consequently, our research not only focuses on technological advancement but also emphasizes the broader applications and societal impacts of these innovations.

C Baselines

In this section, we present the six baseline methods evaluated in our work. For each method, we adopt the default hyperparameter settings provided in the official code of the corresponding papers.

- **Fine-Tune (FT)** [8] is a fine-tuning method that updates the FFN of a transformer layer to incorporate new factual knowledge. The target layer is selected on the basis of its relevance to the knowledge being edited. FT operates by maximizing the likelihood of the target output using the standard next-token prediction loss.
- **MEND** [6] introduces a hypernetwork that maps fine-tuning gradients into efficient weight updates for a pre-trained model. By applying low-rank decomposition to the gradients, it reduces parameter complexity and enables lightweight, localized edits without full model retraining.
- **ROME** [9] performs KE by interpreting the FFN in a transformer layer as a linear associative memory. It derives key-value pairs from internal activations and computes a weight update that ensures the edited layer produces the desired hidden representation. A rank-one modification is then applied to the FFN weights, aligning the model’s internal representations with the new factual knowledge.
- **MEMIT** [8] extends the ROME framework to support simultaneous editing of a large number of factual knowledge items. It models the updates as a joint optimization over key-value pairs and applies rank-one modifications to the FFNs. To prevent interference between edits, MEMIT distributes the updates in a top-down manner across critical FFN layers, achieving efficient, scalable, and stable insertion of new factual knowledge.
- **RECT** [18] reduces the degradation of general capabilities caused by KE. It regularizes weight updates by constraining their magnitude and selectively updates only the top- $k\%$ of parameters with the largest changes during fine-tuning. This reduces overfitting and

helps preserve the model’s reasoning and question-answering abilities, while still achieving effective factual edits.

- **AlphaEdit** [11] introduces a null-space projection mechanism to preserve existing knowledge during editing. It projects the update direction onto the null space of prior knowledge and then applies the projected perturbation to model parameters. This approach reduces interference with previously edited facts and enables effective integration of new information.

D Datasets and Metrics

In this section, we describe the datasets and evaluation metrics employed in our experiments.

D.1 Datasets

We evaluate our methods on two types of datasets: CounterFact and ZsRE for assessing KE, and six benchmarks including SciQ, MMLU, CommonsenseQA, ARC Challenge, WSC273, and LAMBADA for evaluating the general capabilities of post-edited models.

- **Counterfact** [9] is a challenging benchmark that focuses on editing incorrect factual statements in language models. Each instance includes a subject and an incorrect attribute to be updated. To assess edit locality, it provides contrastive prompts involving related but distinct entities, ensuring changes do not affect nearby facts. Additionally, the dataset includes paraphrased and semantically equivalent prompts to evaluate the generalization, fluency, and consistency of the edited model responses.
- **ZsRE** [19] is a question-answering dataset commonly used in knowledge editing evaluation. Each example includes a subject, a target answer to be edited, paraphrased questions for testing generalization, and unrelated questions for evaluating locality. The dataset also features human-written question variants, enabling assessment of model robustness to semantically equivalent inputs.
- **SciQ** [20] is a multiple-choice science QA dataset covering topics such as physics, chemistry, and biology. It is used to evaluate a model’s ability to recall factual scientific knowledge.
- **MMLU** [21] is a multitask benchmark containing questions from 57 academic and professional disciplines. It assesses factual knowledge and generalization across diverse domains in zero- and few-shot settings.
- **CommonsenseQA** [22] is a multiple-choice QA dataset that evaluates a model’s ability to reason over everyday commonsense. It focuses on applying implicit world knowledge to select the correct answer among distractors.
- **ARC Challenge** [23] is a science QA dataset designed to require reasoning beyond simple retrieval. It contains complex grade-school level questions that challenge a model’s problem-solving abilities.
- **WSC273** [24] is a coreference resolution benchmark derived from the Winograd Schema Challenge. It is designed to test whether a model can correctly identify what ambiguous pronouns refer to, based on context and commonsense reasoning.
- **Lambada** [25] is a word prediction benchmark composed of narrative passages where the final word can only be inferred from the entire context. It is designed to evaluate a model’s ability to capture long-range dependencies and maintain discourse-level coherence.

D.2 Metrics

Here, we introduce the evaluation metrics for the CounterFact and ZsRE datasets, which are selected based on previous works [9, 11].

D.2.1 CounterFact Metrics

Given a language model f , a query (s_i, r_i) , an edited object \hat{o}_i , and the original object o_i , the evaluation metrics for CounterFact are defined as follows.

- **Efficacy (success of edited facts):** The proportion of instances in which the model prefers the edited object \hat{o}_i over the original object o_i when prompted with (s_i, r_i) :

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i \mid (s_i, r_i)] > \mathbb{P}_f [o_i \mid (s_i, r_i)]] . \quad (13)$$

- **Generalization (success of paraphrased facts):** The proportion of paraphrased prompts $F_i(s_i, r_i)$, representing rephrasings of the original query (s_i, r_i) , for which the model assigns higher likelihood to \hat{o}_i than to o_i :

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i \mid F_i(s_i, r_i)] > \mathbb{P}_f [o_i \mid F_i(s_i, r_i)]] . \quad (14)$$

- **Specificity (success of neighborhood facts):** The proportion of neighborhood prompts $N_i(s_i, r_i)$, which involve subjects semantically related to but distinct from the original subject s_i , for which the model assigns higher likelihood to the correct object o_i than to the edited object \hat{o}_i :

$$\mathbb{E}_i [\mathbb{P}_f [\hat{o}_i \mid N_i(s_i, r_i)] < \mathbb{P}_f [o_i \mid N_i(s_i, r_i)]] . \quad (15)$$

- **Fluency (generation entropy):** Measures output repetition based on the entropy of n-gram distributions in model outputs. Specifically, it computes a weighted combination of bigram and trigram entropies, where $g_n(\cdot)$ denotes the n-gram frequency distribution:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k) . \quad (16)$$

- **Consistency (reference score):** Consistency is measured by prompting the model f with a subject s and computing the cosine similarity between the TF-IDF vectors of its generated output and a reference Wikipedia passage about the object o .

D.2.2 ZsRE Metrics

Given a language model f , a query (s_i, r_i) , an edited object \hat{o}_i , and the original object o_i , the evaluation metrics for ZsRE are defined as follows:

- **Efficacy (success of edited facts):** Top-1 accuracy on the edited samples, measuring the proportion of cases in which the model ranks the edited object \hat{o}_i as the most likely prediction given the prompt (s_i, r_i) :

$$\mathbb{E}_i \left[\hat{o}_i = \arg \max_o \mathbb{P}_f (o \mid (s_i, r_i)) \right] \quad (17)$$

- **Generalization (success of paraphrased facts):** Top-1 accuracy on paraphrased prompts $F_i(s_i, r_i)$, which are rephrasings of the original query (s_i, r_i) , measuring the proportion of cases in which the model ranks the edited object \hat{o}_i as the most likely prediction for the given paraphrased prompt:

$$\mathbb{E}_i \left[\hat{o}_i = \arg \max_o \mathbb{P}_f (o \mid F_i(s_i, r_i)) \right] \quad (18)$$

- **Specificity (success of neighborhood facts):** Top-1 accuracy on neighborhood prompts $N_i(s_i, r_i)$, which involve subjects related to but distinct from s_i . Specificity reflects whether the model preserves correct predictions on unaffected inputs by still preferring o_i over \hat{o}_i :

$$\mathbb{E}_i \left[o_i = \arg \max_o \mathbb{P}_f (o \mid N_i(s_i, r_i)) \right] \quad (19)$$

E Implementation details

We provide the details of the implementation of the experiments. To reproduce our methods, a GPU with 40G memory is required. Our framework is built on L&E methods like MEMIT [8] and AlphaEdit [11]. We first sequentially compute the key vector \mathbf{k} and the residual vector \mathbf{r} for the target edited facts. The details of the computation can be found in Appendix F. Then we stack them as the key matrix \mathbf{K}_1 and the residual matrix \mathbf{R}_1 and construct a neural KV database $(\mathbf{K}_1, \mathbf{R}_1)$. Then we integrate the non-linear retrieval module with the target FFN layer l^* . Our module only involves one hyperparameter γ to control the gated mechanisms. We provide the details of the setting in the following Table 3.

Table 3: Hyper-parameters of NeuralDB for various models in the main experiments

	GPT2-xl	GPT-J (6B)	Llama 3 Instruct (8B)
Layer found by casual trace	17	17	17
Layer l^* used by NeuralDB	17	8	7
γ	0.65	0.65	0.65

F The computation of \mathbf{k}_i and \mathbf{r}_i

We follow previous locating-and-editing methods [9, 8, 11] to derive the key vector and residual vector from the given edited fact $(s_i, r_i, o_i \rightarrow \hat{o}_i)$. Let l^* denote the FFN layer to be updated.

For the key vector \mathbf{k}_i , we retrieve the specified activation from LLM inferring the prompt. We denote $\mathbf{k}^{l^*}(\mathbf{x})$ as the key activation of the prompt \mathbf{x} in layer l^* . Then the target key vector are computed by the following average over random prefix \mathbf{x}_j :

$$\mathbf{k}_i = \frac{1}{N} \sum_{j=1}^N \mathbf{k}^{l^*}(\mathbf{x}_j + \mathbf{s}_i), \quad (20)$$

where \mathbf{s}_i is the subject of edited fact. The \mathbf{x}_j is the prefix randomly generated by the language model f to improve the robustness of the expressive ability of \mathbf{k}_i .

For the target vector \mathbf{r}_i , we wish to find some vector to decode the new answer \hat{o}_i . We utilize the supervised learning to derive $\mathbf{r}_i = \arg \min_{\mathbf{r}} L(\mathbf{r})$, where the loss object $L(\mathbf{r})$ is defined as following:

$$\frac{1}{N} \sum_{j=1}^N (-\log \mathbb{P}_{f(\mathbf{h}^{l^*} + \mathbf{r})}[o^* | x_j + p] + D_{\text{KL}}(\mathbb{P}_{f(\mathbf{h}^{l^*} + \mathbf{r})}[x | p'] \| \mathbb{P}_f[x | p'])). \quad (21)$$

p is the factual prompt while p' is its variant (the form of “subject is a”). $f(\mathbf{h}^{l^*} + \mathbf{r})$ indicates substituting the output of the i -th MLP with an additional learnable parameter \mathbf{r} . This optimization also uses the random prefix text x_j to enhance the robustness.

G Additional memory usage and computation time

In this section, we provide a detailed discussion of additional resources of our new module.

Memory usage We cache the key matrix \mathbf{K}_1 and the residual matrix \mathbf{R}_1 and construct the new module, which totally take $(d_1 + d_2) \times m$ parameters with m denoting the number of edited facts. For Llama 3 8B model with $d_1 = 14,336$, $d_2 = 4,096$, the memory of 10,000 facts is about 150 million parameters. Compared to the total 8B parameters, the additional memory for 1M facts is only 2.2%. Additionally, our memory grows linearly with the facts and is easily scaled to more facts.

Computation time We report the average evaluation time for three models and two datasets in Table 4. The results show that the averaged time has only a slight improvement compared with the methods without an additional module.

Table 4: The averaged time of evaluation post-edited models on CounterFacts and ZsRE

Model	Llama3			GPTJ-6B		
Method	MEMIT	AlphaEdit	NeuralDB	MEMIT	AlphaEdit	NeuralDB
CounterFacts	4.12	4.11	4.18	3.81	3.76	3.90
ZsRE	0.22	0.22	0.22	0.16	0.16	0.17

Algorithm 1 Old multi-layer method

Require: Input Transformer model f , target layers list $L = [l_1, \dots, l_n]$, request facts \mathcal{F} , Function COMPUTE_KEY to compute the keys of facts at layer l , COMPUTE_RESIDUAL compute the residual of facts at layer l .

- 1: $R \leftarrow \text{COMPUTE_RESIDUAL}(f, \mathcal{F}, l_n)$
- 2: **for** l in L **do**
- 3: $K_i \leftarrow \text{COMPUTE_KEY}(f, \mathcal{F}, l)$
- 4: $R_i \leftarrow R / (l_n - l + 1)$
- 5: Perform KE at layer l with (K_i, R_i)
- 6: **end for**

Algorithm 2 New multi-layer method

Require: Input Transformer model f , target layers list $L = [l_1, \dots, l_n]$, request facts \mathcal{F} , Function COMPUTE_KEY to compute the keys of facts at layer l , COMPUTE_RESIDUAL compute the residual of facts at layer l .

- 1: **for** l in L **do**
- 2: $K_i \leftarrow \text{COMPUTE_KEY}(f, \mathcal{F}, l)$
- 3: $R_i \leftarrow \text{COMPUTE_RESIDUAL}(f, \mathcal{F}, l)$
- 4: Perform KE at layer l with (K_i, R_i)
- 5: **end for**

H Ablation study

H.1 The impact of layer selection and the setting of the parameter γ

Table 5: Ablation study on Llama 3 (8B)

Gamma	Layer	E	G	S	F	C
0.65	7	99.2	85.9	85.6	631.9	32.6
0.65	8	99.2	79.3	85.1	631.5	33.3
0.65	9	99.2	77.4	84.9	631.6	32.4
0.55	7	99.1	91.9	83.4	631.6	32.7
0.75	7	99.2	74.1	86.2	632.3	32.2

We conducted an ablation study to investigate the choice of hyperparameters, including γ and the target layer, with results provided in Table 5. For the configuration shown in Table 1, where $\gamma = 0.65$ and $L = 7$ for Llama3, we vary γ between 0.55 and 0.75, and layer between 8 and 9. Although layer 8 is determined by the causal trace, our results show that its results are not suboptimal. An increase in γ can improve the specificity while damaging the generation, which aligns the gated mechanism. In general, the results demonstrate that it is necessary to search for optimal hyperparameters.

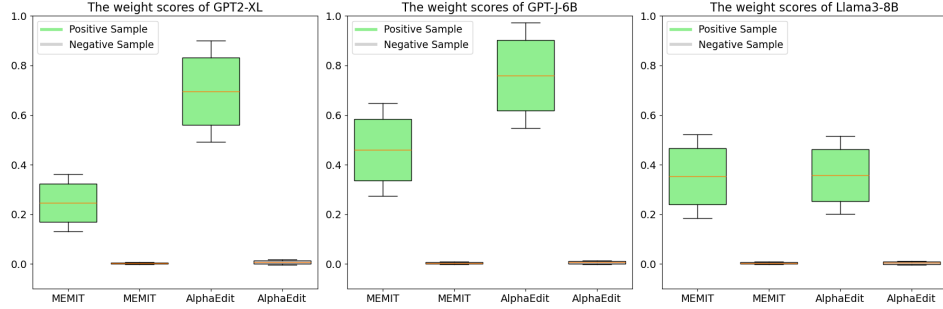
H.2 Multi-layer editing

We also propose two algorithms for multi-layer updating, and compare these two algorithms on two different pre-trained models in Table 6. The experimental results indicate that the new multi-layer method can greatly scale the number of editable facts, albeit with some loss in performance, while the old multi-layer method—though achieving better editing accuracy—requires substantially more storage.

I Additional experiments

Table 6: Editing performance under different layer setup

Model	Layer Setup	Efficacy \uparrow	Generalization \uparrow	Specificity \uparrow	Fluency \uparrow
GPT-J	[8] baseline	99.08	93.48	75.52	620.53
	[6,7,8] new multi layers	94.44	91.72	75.93	617.44
	[6,7,8] old multi layers	99.31	93.23	76.78	616.00
GPT2-XL	[17] baseline	99.04	95.96	70.72	621.90
	[15,16,17] new multi layers	94.81	92.68	70.26	618.51
	[15,16,17] old multi layers	99.08	94.01	71.33	624.48



(a) Paraphrased facts



(b) Neighborhood facts

Figure 5: Visualization of weighted scores for paraphrased facts and neighborhood facts, using MEMIT and AlphaEdit across three models. The boxplots are generated from the mean and variance of weight scores, with the center line indicating the mean, boxes showing ± 1 standard deviation, and whiskers ± 1.5 .

I.1 Im-evaluation-hardness

We conducted more experiments on Im-evaluation-hardness using GPT2-XL and GPT-J, see Table 7 for details.

I.2 Weighted score visualization of paraphrased and neighborhood facts

We further conduct experiments on paraphrased and neighborhood facts to examine the distribution of weighted scores under both MEMIT and AlphaEdit. As shown in Fig. 5, the scores for positive samples in paraphrased facts are consistently higher, while those for negative samples remain close to 0. For neighborhood facts, where all components are considered negative, the scores are likewise consistently close to 0. These results confirm that, during inference, residuals unrelated to the edited facts remain inactive, resulting in near-zero weighted scores.

Table 7: Performance of each task across different editing budgets (1,000, 2,000, 4,000, 6,000, 8,000, 10,000) under various model–algorithm configurations.

(a) GPT-J (AlphaEdit)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.9110	0.9080	0.9060	0.8900	0.8850	0.7410
logiq_a	0.2151	0.2243	0.2089	0.2181	0.2304	0.2181
commonsense_qa	0.2080	0.2146	0.2048	0.1884	0.1925	0.1785
arc_easy	0.6658	0.6477	0.6326	0.6010	0.5804	0.4870
MMLU	0.2660	0.2688	0.2622	0.2592	0.2587	0.2535
arc_challenge	0.3276	0.3148	0.2901	0.2782	0.2611	0.2261
lambada	0.6722	0.6604	0.6057	0.5158	0.4036	0.2203
winogrande	0.6346	0.6227	0.6093	0.5991	0.5730	0.5635
wsc273	0.8425	0.8352	0.7985	0.7399	0.7179	0.6264

(b) GPT-J (NeuralDB)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.9160	0.9160	0.9160	0.9160	0.9160	0.9160
logiq_a	0.2120	0.2120	0.2120	0.2120	0.2120	0.2120
commonsense_qa	0.2080	0.2080	0.2080	0.2080	0.2080	0.2080
arc_easy	0.6692	0.6692	0.6692	0.6692	0.6692	0.6692
MMLU	0.2695	0.2697	0.2697	0.2695	0.2695	0.2698
arc_challenge	0.3396	0.3396	0.3404	0.3404	0.3404	0.3404
lambada	0.6829	0.6827	0.6827	0.6821	0.6821	0.6819
winogrande	0.6409	0.6417	0.6417	0.6417	0.6417	0.6409
wsc273	0.8242	0.8242	0.8242	0.8242	0.8242	0.8242

(c) GPT-2 XL (AlphaEdit)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.8250	0.8230	0.7920	0.7440	0.6390	0.4920
logiq_a	0.2289	0.2273	0.2012	0.2104	0.1951	0.1889
commonsense_qa	0.1908	0.1957	0.1916	0.1974	0.2080	0.1933
arc_easy	0.5682	0.5484	0.4987	0.4693	0.4066	0.3493
MMLU	0.2618	0.2562	0.2464	0.2312	0.2369	0.2315
arc_challenge	0.2423	0.2346	0.2398	0.2108	0.1887	0.2065
lambada	0.4881	0.4170	0.2610	0.1467	0.0767	0.0231
winogrande	0.5904	0.5549	0.5564	0.5272	0.5201	0.5067
wsc273	0.6520	0.6227	0.5714	0.5861	0.5678	0.5421

(d) GPT-2 XL (NeuralDB)

Task	1,000	2,000	4,000	6,000	8,000	10,000
sciq	0.8240	0.8290	0.8280	0.8280	0.8280	0.8280
logiq_a	0.2212	0.2181	0.2181	0.2181	0.2181	0.2197
commonsense_qa	0.1900	0.1933	0.1941	0.1941	0.1941	0.1941
arc_easy	0.5770	0.5785	0.5848	0.5848	0.5848	0.5848
MMLU	0.2532	0.2545	0.2547	0.2546	0.2543	0.2544
arc_challenge	0.2509	0.2509	0.2491	0.2500	0.2500	0.2517
lambada	0.5055	0.5053	0.5077	0.5069	0.5065	0.5053
winogrande	0.5770	0.5785	0.5848	0.5848	0.5848	0.5848
wsc273	0.6777	0.6667	0.6850	0.6850	0.6850	0.6850