Parameter-Aware Contrastive Knowledge Editing: Tracing and Rectifying based on Critical Transmission Paths

Songlin Zhai and Yuan Meng and Yuxin Zhang and Guilin Qi*
School of Computer Science and Engineering, Southeast University, China
{songlin_zhai, yuan_meng, zzyx_cs, gqi}@seu.edu.cn

Abstract

Large language models (LLMs) have encoded vast amounts of knowledge in their parameters, but the acquired knowledge can sometimes be incorrect or outdated over time, necessitating rectification after pre-training. Traditional localized methods in knowledge-based model editing (KME) typically assume that knowledge is stored in particular intermediate layers. However, recent research suggests that these methods do not identify the optimal locations for parameter editing, as knowledge gradually accumulates across all layers in LLMs during the forward pass rather than being stored in specific layers. This paper, for the first time, introduces the concept of critical transmission paths into KME for parameter updating. Specifically, these paths capture the key information flows that significantly influence the model predictions for the editing process. To facilitate this process, we also design a parameter-aware contrastive rectifying algorithm that considers less important paths as contrastive examples. Experiments on two prominent datasets and three widely used LLMs demonstrate the superiority of our method in editing performance.

1 Introduction

Large language models (LLMs) have become the cornerstone of natural language processing (NLP) research as they could serve as the knowledge repositories acquired from extensive pre-training corpora, providing a wealth of information for various NLP tasks (Touvron et al., 2023; Kamalloo et al., 2023; Lai and Nissim, 2024; Chen, 2024). While LLMs have encoded vast amounts of knowledge, the knowledge may be incorrect or outdated over time, highlighting the need for rectification after pre-training (Wang et al., 2024b; Zhang et al., 2024b). For example, an LLM trained before 2023 probably predicts "*PSG*" instead of "*Inter Miami CF*" for the prompt "*Which club does Lionel Messi play for?*" (see Figure 1). Although fine-tuning LLMs is an intuitive way to update knowledge, it technically suffers from issues of high computational costs, overfitting, and catastrophic forgetting (Meng et al., 2022; Li et al., 2024b). An emerging field known as ***k***nowledge-based ***m***odel ***e***diting (KME) offers a cost-effective post hoc modification to enhance the consistency of LLMs, spurring the development of various methods (Meng et al., 2022, 2023; Tan et al., 2024; Li et al., 2024b).

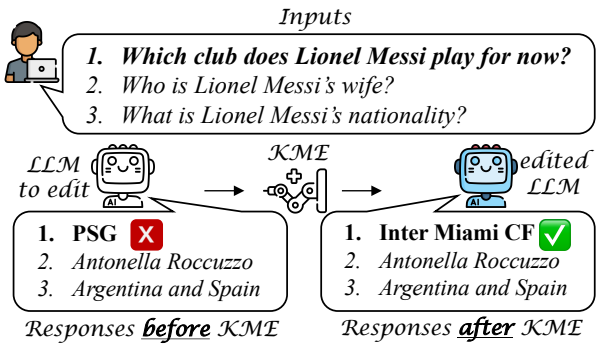


Figure 1: Illustration of KME for (*Lionel Messi, play_for, PSG→Inter Miami CF*). After editing, other knowledge should remain unaffected, *e.g.,* (*Lionel Messi, wife, Antonella Roccuzzo*), (*Lionel Messi, nationality, Argentina and Spain*).

The primary goal of KME is to precisely rectify the specified knowledge within LLMs without disrupting the remainder of the acquired knowledge (see Figure 1). This task is particularly challenging due to the distributed and entangled nature of the encoded knowledge (Hase et al., 2023; Wang et al., 2024b). To achieve this editing target, several lines of research have been proposed, with one promising approach being "*localized modification*" (Meng et al., 2022, 2023; Li et al., 2024b) that focuses on identifying the relevant model parameters storing specific knowledge (*i.e., where to edit*) and then update these weights (*i.e., how to rectify*) to yield desirable outputs. Traditional methods typically assume that the relevant parameters are localized in

*Corresponding Author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 28189–28200
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

certain intermediate LLM layers and harness causal tracing to perform the locating operation (Meng et al., 2022, 2023; Gupta et al., 2023). However, recent research indicates that the localization results from the causal tracing do not statistically correlate with the optimal positions for intervention (Hase et al., 2023). Additionally, causal tracing tends to show the largest causal effects on average in the early hidden layers (*e.g.,* 4-6 layers in GPT-J) and ignore the parameters outside this range (*e.g.,* 16-20 layers of GPT-J) (Hase et al., 2023), leading to sub-optimal editing effects.

To address these issues, this paper abandons the "layer-based localization" and introduces, for the first time, the critical transmission paths (Wang et al., 2018) into KME, since the LLMs' prediction process could be viewed as a forward pass involving gradual information accumulation across layers (Rogers et al., 2020; Geva et al., 2021, 2022; Hase et al., 2023). Specifically, a transmission path is a specific sequence of model parameters and connections across all layers within LLMs, describing an accumulation process from inputs to outputs (Montavon et al., 2019; Achtibat et al., 2024). Critical transmission paths capture the key information routes that significantly influence the model's predictions for the editing, and updating the parameters in these paths could increase the likelihood of yielding desired outputs. To identify the critical paths, we develop a perturbation-based path importance estimation method that measures how much each individual path contributes to correcting the model's predictions. Considering the vast search space of neuron-level paths, we propose a parameter packing strategy to partition the weights of two FFN matrices by column-wise and row-wise manners, inspired by the *key-value memories* viewpoint of FFNs (Geva et al., 2021). This strategy dramatically reduces the search space for candidate paths, thereby lessening the computational burden.

After determining "*where to edit*", we propose a parameter-aware contrastive rectification algorithm to better address "*how to rectify*". This algorithm treats each critical path as a positive example (*i.e.,* representing the parameters requiring updates). Meanwhile, it also selects an insignificant transmission path as a negative example (*i.e.,* negligible for current editing but important for other knowledge) to form the parameter-aware contrastive pair. The underlying motivation is straightforward: once updating the wrong portion of parameters, the current knowledge cannot be rectified effectively and other irrelevant knowledge will also be inadvertently altered. By demonstrating the consequences of improper rectifications to the model, it is promising to achieve a better editing effect. Experimental results demonstrate that our parameter-aware contrastive editor significantly surpasses all compared methods across most evaluation metrics. In addition, experiments on editing time also verify the superiority of our method in editing efficiency. To summarize, the contributions of this paper are listed as follows:

- We introduce the critical transmission paths into KME for the first time to select parameters, effectively addressing the limitations of representation-level causal de-noising.
- After pinpointing critical paths, we propose a parameter-aware contrastive rectification algorithm to facilitate the editing process.
- Experiments on two well-known datasets and three extensively adopted LLMs, along with comparisons to nine strong baselines, demonstrate the superiority of our method regarding editing performance and efficiency.

2 Preliminaries

2.1 Notations

Following the definition of previous KME works, knowledge editing aims to modify the original knowledge triple (s, r, o) encoded in LLMs into the targeted one (s, r, o^*). Here, s represents the subject (*e.g., Lionel Messi*), r a binary relation (*e.g., play_for*), o the old object (*e.g., PSG*), and o^* the expected object (*e.g., Inter Miami CF*). Specifically, we use $\varepsilon_i = (s, r, o \rightarrow o^*)$ to represent a specific editing request, with $\mathcal{E}$ being the collection of knowledge to be edited. During an editing process, (s, r) needs to be expressed as a natural language sentence to align with the input format of LLMs, and we use x_i to represent the corresponding input prompt, *e.g., Which club does Lionel Messi play for now?*. However, there are probably more than one reasonable textual expression, and we employ $\mathcal{X}_{\varepsilon_i}$ (aliased as $\mathcal{X}_i$) to denote other equivalent paraphrases of x_i. Analogously, we use y_i to represent the original textual model output for the input x_i. y_i^* indicates the desired model output of the target object o^*. The model to be edited is represented as a complex function $f_{\mathbf{\Theta}}$, where $\mathbf{\Theta}$ denotes the original model parameters.

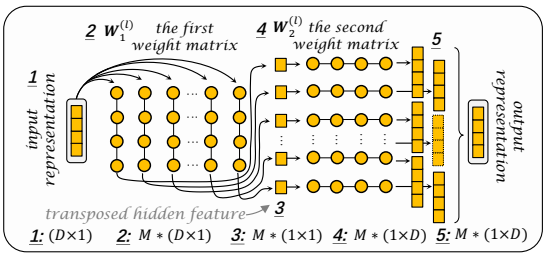


Figure 2: Illustration of transmission paths and the packing strategy. Before applying the packing strategy, each path is composed of each weight in FFNs across all layers, *e.g.,* $(\theta^1_{1,2}, \theta^2_{3,9})^{(1)} \rightarrow (\theta^1_{8,5}, \theta^2_{9,1})^{(2)} \rightarrow ... \rightarrow (\theta^1_{7,5}, \theta^2_{3,4})^{(L)}$. After applying the strategy, each path becomes a sequence of weight vectors, *e.g.,* $(\boldsymbol{\theta}^1_2, \boldsymbol{\theta}^2_9)^{(1)} \rightarrow ...(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_j)^{(l)}... \rightarrow (\boldsymbol{\theta}^1_{13}, \boldsymbol{\theta}^2_7)^{(L)}$, where $(\boldsymbol{\theta}^1_i, \boldsymbol{\theta}^2_j)^{(l)}$ (*i.e.,* $\boldsymbol{k}^{(l)}_i$ and $\boldsymbol{v}^{(l)}_j$) are the i-th column and the j-th row of $\boldsymbol{W}^{(l)}_1$ and $\boldsymbol{W}^{(l)}_2$, respectively.

2.2 Task Definition

KME aims to incorporate the new knowledge by precisely updating a small fraction of parameters in the given LLM, without negatively impacting other encoded knowledge that is irrelevant to the edit set. Formally, the editing target can be denoted as:

$$f_{\boldsymbol{\Theta}^*}(x) = \begin{cases} y^*_i, & \varepsilon_i \in \mathcal{E}, x \in \{x_i, \mathcal{X}_i\} \\ y_i, & \varepsilon_i \notin \mathcal{E}, x \in \{x_i, \mathcal{X}_i\} \end{cases} \quad (1)$$

Here, $\boldsymbol{\Theta}^* = \boldsymbol{\Theta} + \Delta\boldsymbol{\Theta}^*$ represents the updated model parameters on the editing set $\mathcal{E}$, where $\boldsymbol{\Theta}$ denotes the original parameters and $\Delta\boldsymbol{\Theta}^*$ is the parameter update matrix. Notably, $\Delta\boldsymbol{\Theta}^*$ should be sparse, indicating that only a small subset of parameters are modified during this process.

2.3 Feed-Forward Network (FFN)

Before introducing the transmission paths into KME, we first review a key module of LLMs, namely the ***f***eed-***f***orward ***n***etwork (FFN). It typically consists of two linear transformations separated by an activation function (*e.g.,* ReLU), which captures complex nonlinear relationships within the input representation. This module is positioned after the self-attention module and is defined as:

$$\text{FFN}^{(l)}(\boldsymbol{x}) = \text{ReLU}(\boldsymbol{x}^\top \boldsymbol{W}^{(l)}_1)\boldsymbol{W}^{(l)}_2 \quad (2)$$

where $\boldsymbol{x}$ is the input representation. $\boldsymbol{W}^{(l)}_1$ and $\boldsymbol{W}^{(l)}_2$ are the weight matrices at the l-th fully connected layer. Specifically, the sizes of these two matrices are $D \times M$ and $M \times D$ (*i.e.,* $\boldsymbol{W}^{(l)}_1 \in \mathbb{R}^{D\times M}$, $\boldsymbol{W}^{(l)}_2 \in \mathbb{R}^{M\times D}$), where D and M refer to the hidden dimensions of the model and the FFN layer (*e.g.,* $D = 4,096$ and $M = 14,336$ in Llama3 (8B)), respectively.

2.4 Transmission Paths

A transmission path describes the process of information accumulation from inputs to outputs and consists of a specific sequence of model parameters and connections spanning all layers within LLMs (Montavon et al., 2019; Achtibat et al., 2024). Following previous works (Meng et al., 2022, 2023; Li et al., 2024b; Zhang et al., 2024a), we also focus on the information accumulation in FFNs. Consequently, each transmission path can be formulated as a set of parameters in $\boldsymbol{W}^{(l)}_1$ and $\boldsymbol{W}^{(l)}_2$:

$$\tau = \{(\Theta^{(l)}_1, \Theta^{(l)}_2) \mid 1 \le l \le L\} \quad (3)$$

Here, $\tau \in \mathcal{T}$ represents a specific transmission path, with $\mathcal{T}$ being the set of paths in the given LLM. $\Theta^{(l)}_1$ and $\Theta^{(l)}_2$ are the nodes of path τ at the l-th layer, which are part of the parameters in $\boldsymbol{W}^{(l)}_1$ and $\boldsymbol{W}^{(l)}_2$. L refers to the number of hidden layers.

3 Parameter-Aware Contrastive KME

Based on the transmission path defined in Eq. 3, KME can be performed by identifying the critical paths from $\mathcal{T}$ (*i.e.,* finding the important $\Theta^{(l)}_1$ and $\Theta^{(l)}_2$ across all layers) and then updating the parameters along these paths. Intuitively, the nodes $\Theta^{(l)}_1$ and $\Theta^{(l)}_2$ shown in Eq. 3 could be selected from $\boldsymbol{W}^{(l)}_1$ and $\boldsymbol{W}^{(l)}_2$ in a neuron-by-neuron manner. However, the time complexity of this neuron-level selection will be astonishing $\mathcal{O}[L\times(D\times M)^2]$. Furthermore, the gradient updates on the neuron-level transmission path can compromise the model's robustness (Yu et al., 2023). To alleviate these issues, we propose a parameter-packing strategy that partitions the parameters of each FFN layer into different segments and packs the parameters within the same segment into a single node of one path.

3.1 Parameter Packing Strategy

Inspired by the investigation of Geva et al. (2021), which suggests that *the two matrices in each FFN layer can be treated as key-value memories*, we reformulate the FFN layer shown in Eq. 2 as follows:

$$\text{FFN}^{(l)}(\boldsymbol{x}) = g(\boldsymbol{x}^\top \underbrace{\boldsymbol{K}^{(l)}}_{\boldsymbol{W}^{(l)}_1})\underbrace{\boldsymbol{V}^{(l)}}_{\boldsymbol{W}^{(l)}_2} = \sum_{j=1}^{M} \underbrace{g(\boldsymbol{x}^\top \boldsymbol{k}^{(l)}_j)}_{\alpha^{(l)}_j} \boldsymbol{v}^{(l)}_j \quad (4)$$

where g is the activation function. $\boldsymbol{K}^{(l)}$ and $\boldsymbol{V}^{(l)}$ are the augmented versions[1] of $\boldsymbol{W}_1^{(l)}$ and $\boldsymbol{W}_2^{(l)}$, analogous to the key and value matrices used in the attention mechanism. Here, $\boldsymbol{k}_j^{(l)}$ and $\boldsymbol{v}_j^{(l)}$ represent the j-th column and row weight vectors in $\boldsymbol{W}_1^{(l)}$ and $\boldsymbol{W}_2^{(l)}$, respectively.

Eq. 4 implies that the output representation of the l-th FFN layer can be treated as a weighted sum over the value vectors $\boldsymbol{v}_j^{(l)}$, with $\alpha_j^{(l)}$ serving as the weighting coefficient, where each $\boldsymbol{v}_j^{(l)}$ has the size of $1 \times D$. Specifically, $\alpha_j^{(l)}$ is computed by taking the dot product between the transposed input token representation $\boldsymbol{x}^\top (1 \times D)$ and the column-wise key vector $\boldsymbol{k}_j^{(l)} (D \times 1)$. These observations motivate us to pack parameters of the first and second weight matrices of the FFN layer in a column-wise and row-wise manner, respectively. This packing strategy allows the transmission path defined in Eq. 3 to be reformulated as follows:

$$\tau = \{(\boldsymbol{k}_i^{(l)}, \boldsymbol{v}_j^{(l)}) \mid 1 \leq l \leq L, 1 \leq i, j \leq M\} \quad (5)$$

Eq. 5 states that the parameters in the two weight matrices (*i.e.,* $\boldsymbol{W}_1^{(l)}$ and $\boldsymbol{W}_2^{(l)}$) can be divided into different groups, corresponding to the key and the value vectors respectively. This strategy allows knowledge editing to be performed on these parameter blocks, thereby reducing the time complexity from $\mathcal{O}[L \times (D \times M)^2]$ to $\mathcal{O}(L \times M^2)$. Figure 2 illustrates the packing strategy and depicts the transmission paths in this configuration.

3.2 Tracing Critical Transmission Paths

After defining what constitutes a "*transmission path*", the next step is to determine "*where to perform editing*". Specifically, we need to identify the critical information transmission paths that influence the model to shift its prediction from the "old" answer to the desired one for a given editing request ε_i. To achieve this, we introduce a perturbation-based method to estimate the importance of each transmission path.

3.2.1 Impact Score of Transmission Paths

To estimate the importance, we first review the goal of knowledge editing, *i.e.,* increasing the relative likelihood of desired outputs without changing model behavior for unrelated inputs. Thus, the impact score of each transmission path can be measured by the degree of output interference in obtaining desired predictions after adding infinitesimal noise into parameters in τ. Based on this principle, we apply perturbation theory (Keinan, 2005) to estimate the impact score as[2]:

$$\begin{aligned}\phi(\tau|\varepsilon_i) &= \lim_{\boldsymbol{\epsilon}_\tau \to \mathbf{0}} \frac{\mathcal{L}(y_i^*|\boldsymbol{\Theta} + \boldsymbol{\epsilon}_\tau, x_i) - \mathcal{L}(y_i^*|\boldsymbol{\Theta}, x_i)}{\boldsymbol{\epsilon}_\tau} \\ &\approx \sum\nolimits_{\theta \in \tau} \frac{\partial \mathcal{L}}{\partial \theta}\end{aligned} \quad (6)$$

Here, $\phi(\tau|\varepsilon_i)$ represents the impact score of the transmission path τ for ε_i. $\mathcal{L}$ is the cross-entropy loss function, which measures the discrepancy between the model prediction and the expected output. $\boldsymbol{\epsilon}_\tau$ represents the noise introduced into the packed parameters of the transmission path τ.

After estimating the impact scores of all transmission paths, we can identify which paths are most sensitive to the current knowledge being edited. Intuitively, we select the paths with the highest scores as the critical ones ($\mathcal{T}^+$ or $\mathcal{T}^+(\varepsilon_i)$) that can be defined as $\mathcal{T}^+(\varepsilon_i) = \{\tau \mid 1 \leq r(\phi(\tau|\varepsilon_i)) \leq N\}$. Here, $r(\cdot)$ returns the rank position of the given path score within the score list of the entire path set $\mathcal{T}$ (descending order). N is the size of critical transmission paths.

3.3 Parameter-Aware Contrastive Editing

After obtaining $\mathcal{T}^+$, we also sample another path with the lowest score as the negative example (*i.e.,* $\mathcal{T}^-$). The rationale behind this design is to demonstrate the consequences of editing parameters that should not be modified, thereby enhancing the effectiveness of editing the correct parameters. Formally, the parameter-aware contrastive loss can be formulated as follows:

$$\mathcal{J}(\varepsilon_i) = \mathcal{L}(f_{\boldsymbol{\Theta}^*}(x_i), y_i^*) + \lambda \mathcal{L}(f_{\boldsymbol{\Theta}'}(x_i), y_i) \quad (7)$$

where $\mathcal{J}$ is the loss associated with the edit ε_i. $\mathcal{L}$ measures the differences between the model predictions $f_{\boldsymbol{\Theta}^*}(x_i)$ (or $f_{\boldsymbol{\Theta}'}(x_i)$) and the corresponding labels y_i^* (or y_i). Specifically, the consequent model outputs after editing parameters that should not be modified are represented by $f_{\boldsymbol{\Theta}'}(x_i)$. $\boldsymbol{\Theta}^*$ represents the parameters optimized along the positive paths $\mathcal{T}^+$, while $\boldsymbol{\Theta}'$ refers to the parameters updated along the negative path $\mathcal{T}^-$. The term λ scales the loss associated with $\mathcal{T}^-$.

[1]When the bias term is included in the FFN layer, these two matrices will be the augmented matrices.

[2]Referring to the definition of the derivative: $h'(x) = \lim_{\Delta x \to 0} \frac{h(x+\Delta x) - h(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta h}{\Delta x}$.

Editor	ZsRE				CounterFact			
	Efficacy	*Locality*	*Generality*	*Score*	*Efficacy*	*Locality*	*Generality*	*Score*
GPT-J (6B) Original Model	26.32	/	25.79	26.06	16.22	/	18.56	17.39
Full-C (Zhu et al., 2021)	72.37	19.66	68.91	53.65	92.15	43.35	72.38	69.29
ROME (Meng et al., 2022)	56.42	9.86	54.65	40.31	57.50	52.05	54.20	54.58
MEMIT (Meng et al., 2023)	94.91	30.39	90.22	71.84	98.55	63.64	95.50	85.90
PRUNE (Ma et al., 2025)	0.15	0.00	0.15	0.10	86.15	53.87	86.85	75.62
RECT (Gu et al., 2024)	96.38	27.79	91.21	71.79	98.80	72.22	86.58	85.87
AlphaEdit (Fang et al., 2025)	99.79	28.29	**96.00**	74.69	99.75	**75.48**	**96.38**	**90.54**
Ours	**100**	**93.22**	63.75	**85.66**	**100**	17.00	12.00	43.00
Llama3 (8B) Original Model	36.99	/	36.34	36.67	7.85	/	10.58	9.22
Full-C (Zhu et al., 2021)	30.48	15.49	30.22	25.40	83.33	46.63	67.79	65.92
ROME (Meng et al., 2022)	2.01	0.69	1.80	1.50	64.40	49.44	61.42	58.42
MEMIT (Meng et al., 2023)	34.62	18.49	31.28	28.13	65.65	51.56	64.65	60.62
PRUNE (Ma et al., 2025)	24.77	20.69	23.87	23.11	68.25	49.82	64.75	60.94
RECT (Gu et al., 2024)	86.05	31.67	80.54	66.09	66.05	61.41	63.62	63.69
AlphaEdit (Fang et al., 2025)	94.47	32.55	**91.13**	72.72	98.90	**67.88**	**94.22**	**87.00**
Ours	**98.21**	**85.36**	77.04	**86.87**	**100**	16.00	23.00	46.33

Table 1: Average performance comparison under the ***batch*** editing with batch_size = 100. The baseline results are from Fang et al. (2025).

The loss in Eq. 7 indicates that the optimization of parameters along the critical paths aims to minimize the distance between the edited model output and the expected outcome, thereby ensuring the effectiveness of the edit request. Conversely, if the editing is applied to parameters within other insignificant paths, the model predictions should not change drastically for the current edit, as ensured by $\mathcal{L}(f_{\mathbf{\Theta}'}(x_i), y_i)$. This aspect of the optimization also ensures that other knowledge remains undisturbed, as these paths are likely crucial for other knowledge. Maintaining the original output further ensures that unrelated knowledge is not affected.

4 Experiments

4.1 Experimental Settings

Datasets & LLMs: Following previous works (Mitchell et al., 2022a; Meng et al., 2022, 2023; Li et al., 2024b), we also use the EDIT sets of **ZsRE** (Levy et al., 2017) and **CounterFact** (Meng et al., 2022) to evaluate our method. Specifically, **ZsRE** is a question-answering dataset designed for zero-shot relation extraction, while **CounterFact** focuses on inserting counterfactual knowledge into models. Additionally, three prominent auto-regressive LLMs are employed to perform editing, *i.e.,* **GPT-J** (6B) (Wang and Komatsuzaki, 2021), **Llama2** (7B) (Touvron et al., 2023) and **Llama3** (8B) (Llama Team, 2024).

Baselines & Evaluation Metrics: To evaluate the effectiveness of the proposed method, we compare it with nine baselines. For the editing performance comparison, we adopt three fundamental metrics for evaluation, *i.e.,* **Efficacy**, **Generality**, and **Locality**. In this paper, we consider the *batch editing* and a more difficult scenario, *i.e., consecutive editing*. *Batch editing* refers to the simultaneous processing of multiple edit requests. For the *consecutive editing*, all edit requests are done successively without rolling back parameters after each edit, and the evaluation is conducted after all knowledge updates have been completed.

The experiments of all methods were conducted on an *NVIDIA A100-SXM4-40GB* machine. The baseline methods are implemented using the widely adopted EasyEdit toolkit[3], with hyperparameters configured according to the recommended settings.

4.2 Comparison of Editing Effects

The performance of all compared methods in Table 2 is evaluated on 3K samples from both datasets under consecutive editing. It presents the average performance, where *Score* is the mean result of *Efficacy*, *Locality*, and *Generality*. While most methods show satisfactory performance under batch

[3] https://github.com/zjunlp/EasyEdit

Editor	ZsRE				CounterFact			
	Efficacy	*Locality*	*Generality*	*Score*	*Efficacy*	*Locality*	*Generality*	*Score*
GPT-J (6B) Original Model	21.65	/	21.10	21.37	0.30	/	0.23	0.27
Full-C (Zhu et al., 2021)	11.04	1.59	8.41	7.01	21.33	1.27	7.97	10.19
ROME (Meng et al., 2022)	31.87	18.29	28.10	26.09	0.13	0.03	0.20	0.12
KN (Dai et al., 2022)	0.00	0.01	0.00	0.003	0.01	0.00	0.007	0.006
MEMIT (Meng et al., 2023)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PMET (Li et al., 2024b)	0.02	0.03	0.02	0.02	0.00	0.00	0.00	0.00
AlphaEdit (Fang et al., 2025)	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
LoRA (Xu et al., 2024)	1.11	0.01	1.15	0.76	0.97	0.13	0.67	0.59
EMMET (Gupta et al., 2024b)	55.21	37.47	51.67	48.12	70.20	33.03	**41.17**	48.13
R-ROME (Gupta et al., 2024a)	54.74	13.33	**51.76**	39.96	69.27	**41.87**	37.40	**49.51**
Ours	**88.74**	**51.28**	49.50	**63.17**	**90.70**	1.83	5.33	32.62
Llama2 (7B) Original Model	34.73	/	34.59	34.66	15.19	/	11.55	13.37
Full-C (Zhu et al., 2021)	7.88	0.55	6.73	5.05	2.24	2.31	0.05	1.53
ROME (Meng et al., 2022)	9.16	1.12	8.29	6.19	36.96	3.24	18.77	19.66
MEMIT (Meng et al., 2023)	0.00	0.03	0.00	0.01	0.00	6.43	0.00	2.14
KN (Dai et al., 2022)	1.02	0.03	0.09	0.38	0.37	0.02	0.29	0.23
PMET (Li et al., 2024b)	3.68	1.83	3.68	3.06	0.23	0.47	0.17	0.29
AlphaEdit (Fang et al., 2025)	2.83	0.97	2.81	2.20	0.00	4.41	0.00	1.47
LoRA (Xu et al., 2024)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
EMMET (Gupta et al., 2024b)	25.01	2.87	22.43	16.77	38.67	5.83	**28.35**	24.28
R-ROME (Gupta et al., 2024a)	21.21	1.52	17.78	13.50	41.06	5.66	25.92	24.21
Ours	**84.09**	**75.77**	**66.20**	**75.35**	**71.46**	**20.62**	20.96	**37.68**
Llama3 (8B) Original Model	26.27	/	25.98	26.13	0.87	/	0.75	0.81
Full-C (Zhu et al., 2021)	7.69	0.69	6.66	5.01	5.75	0.13	0.47	2.12
ROME (Meng et al., 2022)	3.39	0.15	2.80	2.11	25.07	0.97	13.23	13.09
MEMIT (Meng et al., 2023)	0.00	3.96	0.00	1.32	0.00	**7.22**	0.00	2.41
KN (Dai et al., 2022)	0.03	0.01	0.01	0.02	0.11	0.02	0.05	0.06
PMET (Li et al., 2024b)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
AlphaEdit (Fang et al., 2025)	0.01	0.003	0.00	0.004	0.33	0.07	0.17	0.19
LoRA (Xu et al., 2024)	11.45	5.35	11.16	9.32	0.77	0.17	1.17	0.70
EMMET (Gupta et al., 2024b)	5.17	0.43	4.86	3.49	54.50	1.28	**38.82**	31.53
R-ROME (Gupta et al., 2024a)	2.71	0.35	2.43	1.83	48.92	1.47	36.62	29.00
Ours	**94.03**	**59.01**	**67.35**	**73.46**	**93.53**	1.93	7.11	**34.19**

Table 2: Average performance of all compared methods under the ***consecutive*** editing.

editing (see Table 1), their performance significantly deteriorates under consecutive editing, with some models even dropping to zero. This suggests that consecutive editing severely impacts the original LLMs, likely due to inaccurate identification of relevant parameters and ineffective updates. This results in negative effects not only on the edited knowledge but also on the original knowledge stored within the model. In contrast, our method consistently outperforms the compared methods by significant margins across most metrics, highlighting the effectiveness of critical information transmission paths. Additionally, we observe that our method exhibits relatively low performance on the *Generality* and *Locality* metrics when evaluated on the CounterFact dataset. This limitation arises from an inherent challenge in information path-based knowledge editing, *i.e.,* if an inappropriate intermediate node is selected and directly modified, it may disrupt the intended information transmission path, leading to unintended alterations in the model's original behavior. Unlike other benchmarks, CounterFact primarily focuses on inserting new factual knowledge rather

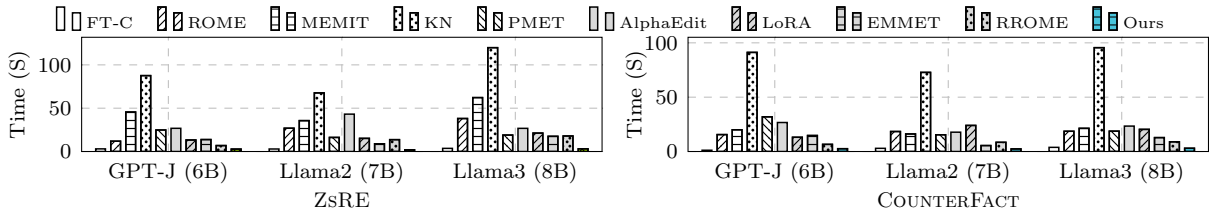


Figure 3: Comparison of average time per editing among all methods on two datasets.

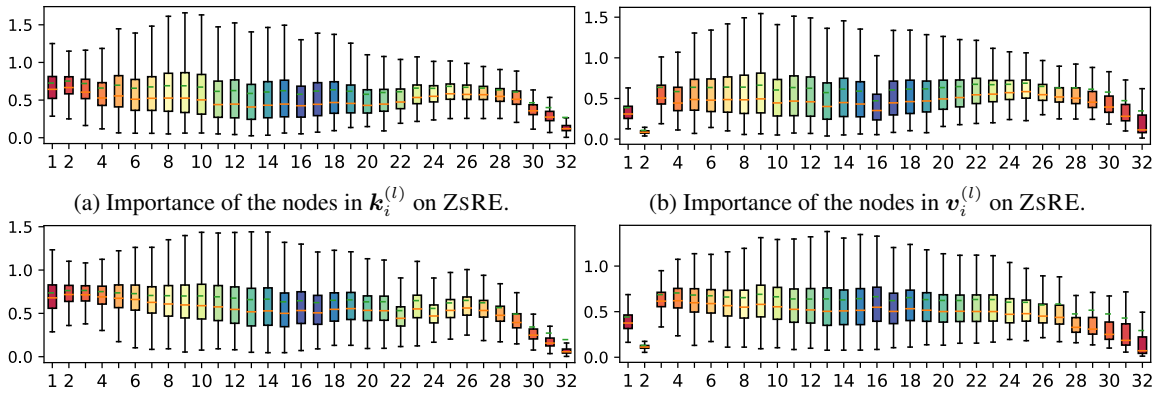
than modifying pre-existing information. Consequently, the internal information pathways corresponding to such novel knowledge are often not well-established within the model. Editing certain nodes along these underdeveloped paths can inadvertently interfere with previously learned knowledge. We have identified this issue and are currently investigating adaptive rectification strategies that dynamically adjust parameter updates in the critical paths across different layers, aiming to mitigate such unintended side effects and improve both *Generality* and *Locality* in knowledge editing.

4.3 Comparison of Editing Time

Editing efficiency is a critical factor in evaluating the KME method. Figure 3 presents the average time per edit for all compared methods, with experiments conducted on an *NVIDIA A100* machine equipped with an *Intel(R) Xeon(R) Gold 5215@2.50GHz* CPU. Our method demonstrates strong efficiency across all models, requiring only 2.8, 2.0, and 3.1 seconds per edit for GPT-J, Llama2 (7B), and Llama3 (8B) on the ZsRE dataset. In comparison, FT-C shows a slightly longer editing time, with 3.06, 2.91, and 3.63 seconds for the same models on the ZsRE dataset. Notably, methods like MEMIT, AlphaEdit, and PMET incur significantly longer editing times compared to other localization-based approaches. Among all methods, KN exhibits the longest editing time, making it considerably less efficient than others.

4.4 Analysis of Critical Transmission Path

The critical path refers to the information accumulation path across all layers for a given input. To better understand $\mathcal{T}^+$, we estimate the contribution of each node in every layer to the overall impact of the critical transmission path, shown in Figure 4. Specifically, the *x-axis* denotes the index of the hidden layer, while the *y-axis* represents the distribution of importance scores. The left panel presents the results for the packed key nodes $\boldsymbol{k}_i^{(l)}$, while the right one for the value nodes $\boldsymbol{v}_i^{(l)}$. This box-plot visualization provides several important insights: **(1)** Notably, the same model exhibits consistent trends across different datasets. This consistency highlights the stability of the model's internal information flow and further supports the robustness of the critical transmission paths identified by our method. **(2)** All hidden layers contribute to the knowledge editing, revealing a key limitation of prior methods that focus only on specific layers. This finding suggests that effective knowledge editing should account for the entire network rather than being restricted to particular layers. Relying solely on middle layers may result in deficiencies or even disrupt the model's forward information accumulation, ultimately degrading editing performance. Moreover, the specific choice of which middle layers to modify can significantly affect the outcome, possibly explaining the performance variability observed across several baseline methods (see Table 2). **(3)** The influence of different layers is not uniformly distributed. While all layers play a role in editing, nodes in the middle layers (*e.g.,* layers 4-18 in Llama 3 8B) exert a stronger influence. Therefore, these layers should be given higher priority when performing model updates. Currently, our method uniformly optimizes all nodes along the path; however, ongoing work explores adaptive re-weighting strategy that emphasizes nodes in these more impactful middle layers. **(4)** Despite the strong influence of middle layers (*e.g.,* 4–18), node importance varies significantly within these layers. Some nodes may be highly entangled with other unrelated knowledge and are thus unsuitable for editing. Incorrectly updating such nodes can reduce editing success and cause unintended disruptions to the model's broader knowledge. Accurate identification and selection of critical nodes within these layers are therefore essential. This further highlights the effectiveness of our method in identifying important nodes as reflected in its strong experimental performance. **(5)** Early (layers 1–4) and late (layers 28–32) layers exhibit distinct behaviors. Interestingly, early layers contribute



(a) Importance of the nodes in $\boldsymbol{k}_i^{(l)}$ on ZsRE.

(b) Importance of the nodes in $\boldsymbol{v}_i^{(l)}$ on ZsRE.

(c) Importance of the nodes in $\boldsymbol{k}_i^{(l)}$ on CounterFact.

(d) Importance of the nodes in $\boldsymbol{v}_i^{(l)}$ on CounterFact.

Figure 4: Importance of each node in $\mathcal{T}^+$ across all Llama3 (8B) layers on ZsRE (Top) CounterFact (Bottom).

more significantly than late layers. Additionally, importance scores in the late layers remain relatively stable, suggesting that nodes in these layers behave more uniformly and have a consistent level of influence across edits.

4.5 Effects of the Contrastive Rectification

The contrastive loss defined in Eq. 7 plays a crucial role in enhancing the rectification process. In Figure 6, the result for $\lambda = 0$ illustrates the editing performance without the contrastive rectification. From these results, we can observe that the contrastive rectification significantly improves the model's *Efficacy* by approximately 4%, while maintaining the stability of *Generality* and *Locality*. This outcome suggests that introducing contrastive rectification allows the model to focus more effectively on relevant knowledge, enhancing its editing accuracy without disrupting its ability to generalize or localize information. Additionally, the size of $|\mathcal{T}^-|$ is a hyperparameter of the model, representing the number of negative transmission paths used in contrastive optimization. However, setting this value too high inevitably leads to a notable decline in both *Efficacy* and *Generality*. This degradation is likely due to the model placing excessive emphasis on the contrastive loss, effectively becoming overly focused on learning "what should not be done". As a result, the model's ability to successfully apply edits and generalize to related contexts is compromised. Based on empirical evaluation, we set $|\mathcal{T}^-| = 1$ to achieve optimal performance.

4.6 Analysis of the size of $\mathcal{T}^+$

Figure 5 (the left column) shows the effects of varying the size of $|\mathcal{T}^+|$ on three evaluation metrics. In particular, two special cases are represented:

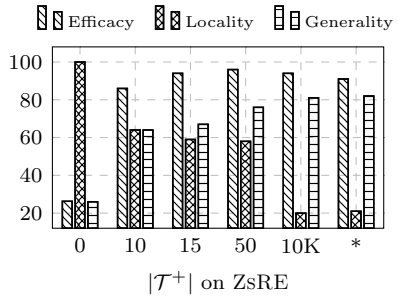


Figure 5: Analysis of $|\mathcal{T}^+|$ for Llama3 (8B).

$|\mathcal{T}^+| = 0$ and $|\mathcal{T}^+| = *$, which correspond to the original model and full FFN parameters fine-tuning, respectively. It can be observed that the editing success rate and the model generality slightly rise as $|\mathcal{T}^+|$ increases. This is because the information accumulation path associated with the current edit is effectively modified, allowing the model to better integrate new information. However, once $|\mathcal{T}^+|$ surpasses a certain threshold (15 for Llama3 (8B) on $3K$ ZsRE), *Locality* declines sharply. This drop can be attributed to the introduction of irrelevant paths into the optimization process, which may act as noise and negatively affect the original model's knowledge. For practical purposes, we set $|\mathcal{T}^+| = 15$ as a reasonable compromise between improvement and stability.

4.7 Analysis of λ

λ serves to balance the contrastive loss in the optimization process. Figure 6 (the right column) visualizes the impact of varying λ on the model's performance. As shown, the result generally declines as λ increases. This is likely due to the overemphasis placed on the contrastive loss, leading to the overfitting of expected predictions. This effect is most

apparent in the evaluation of the *Efficacy*, where the model struggles to maintain accuracy when the contrastive loss becomes too dominant. Moreover, the *Locality* metric exhibits lower fluctuations at larger values of λ, indicating that the contrastive optimization could benefit the preservation of unrelated knowledge. Given these observations, we set $\lambda = 0.1$ as the optimal value to balance the contrastive loss. This value provides sufficient effects of the contrastive rectification without excessively impacting overall performance.

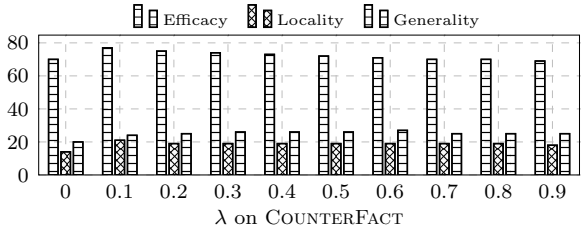


Figure 6: Analysis of λ for Llama2 (7B).

5 Related Work

According to Wang et al. (2024b); Mazzia et al. (2023); Zhang et al. (2024b), KME methods can be classified into two main categories, *i.e.,* parameter-preserved and parameter-modified.

5.1 Preserving Parameters

Methods for preserving parameters typically involve external memories (Wang et al., 2024a), in-context learning, or altering the LLM's representation space. Mitchell et al. (2022b) introduce SERAC to store edits in explicit memory and reason over them. Li et al. (2023) optimize controllability and robustness by considering interactions with factual context. Wang et al. (2024c) employ depth-first search-based constrained decoding for multi-step reasoning in knowledge editing. Zheng et al. (2023) explore in-context learning in factual knowledge editing. Other works focus on adding extra parameters, such as patching models with natural language (Murty et al., 2022), adapting parameters to factual texts (Dong et al., 2022), altering subject word embeddings (Li et al., 2025), and using Key-Value adaptors (Hartvigsen et al., 2023). Some research also addresses reducing computational resources (Yu et al., 2024) or editing in the representation space (Hernandez et al., 2024).

5.2 Modifying Parameters

Research on modifying LLM parameters includes fine-tuning, hyper-network, and localization-based methods. Ni et al. (2024) propose forgetting-before-learning for effective fine-tuning. Hyper-network-based methods include Cao et al. (2021), with Mitchell et al. (2022a) improving it via low-rank gradient decomposition. Tan et al. (2024) resolve synchronous editing via least squares aggregation. Localization methods like ROME (Meng et al., 2022) edit FFN weights using causal intervention, with subsequent improvements by Meng et al. (2023), Li et al. (2024b), and Hu et al. (2024). Others include Gupta et al. (2023) on commonsense judgment, Wu et al. (2023) and Li et al. (2024a) on neuron editing and model computation impacts.

6 Conclusion

This paper introduced the critical transmission paths into KME for the first time, alleviating the limitations of layer-based localization methods. Specifically, critical paths captured key information flows across layers, and we proposed a perturbation-based model to identify these paths for each editing request. Additionally, we introduced a parameter-aware contrastive loss that incorporates both critical and inconsequential paths, enhancing the efficacy and generality of the edited model. Extensive experiments on three popular LLMs and two widely used KME datasets showed that our approach outperformed all compared methods in both editing performance and efficacy.

7 Limitations

While our method achieves notable improvements in KME, several limitations remain. First, the current method assumes that all layers within the identified critical transmission path contribute equally to the editing process. However, this assumption may overlook the varying degrees of influence that different layers exert. As illustrated in Figure 4, we are currently investigating layer-wise contributions to editing effectiveness. Incorporating a more fine-grained, layer-sensitive optimization strategy could further enhance performance through adaptive weighting. Second, our method operates with a fixed-size critical transmission path during the editing phase. This static configuration may not be optimal across different types of edits or task requirements. To address this, future work will focus on dynamically adjusting the node number based on the specific characteristics of each editing request, thereby improving the model's adaptability and robustness across diverse scenarios.

8 Ethical Considerations

While Knowledge-based Model Editing (KME) offers a powerful technique for correcting incorrect knowledge within language models, it is crucial to recognize its potential for misuse. Specifically, KME could be exploited to inject biased, harmful, or misleading information into the model. Since the technique allows for the modification of a model's internal knowledge, careful oversight is necessary to ensure that the changes made are ethical and do not promote discrimination, harmful stereotypes, or other detrimental effects.

One of the primary ethical concerns lies in the potential for malicious actors to alter a model's knowledge base with the intent to manipulate its output. Without robust safeguards and verification processes, there is a risk that KME could facilitate the introduction of harmful content that undermines the reliability and fairness of the model's outputs. This is especially important in contexts where the model is used for decision-making or information dissemination, as biased or harmful knowledge could lead to real-world consequences.

Acknowledgments

This study is supported by National Social Science Foundation Key Program of China (No.23&ZD222). We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper.

References

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. 2024. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In *ICML*, volume 235, pages 135–168. Proceedings of Machine Learning Research.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *EMNLP*, pages 6491–6506. Association for Computational Linguistics.

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *Data Intelligence*, 6:587–620.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *ACL*, pages 8493–8502. Association for Computational Linguistics.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. 2022. Calibrating factual knowledge in pretrained language models. In *EMNLP Findings*, pages 5937–5947. Association for Computational Linguistics.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained knowledge editing for language models. In *ICLR*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *EMNLP*, pages 30–45. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *EMNLP*, 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. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *EMNLP*, pages 16801–16819. Association for Computational Linguistics.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. 2024a. Rebuilding ROME : Resolving model collapse during sequential model editing. In *EMNLP*, pages 21738–21744. Association for Computational Linguistics.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024b. A unified framework for model editing. In *EMNLP*, pages 15403–15418. Association for Computational Linguistics.

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *EMNLP*, pages 8214–8232. 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 *NeurIPS*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *NeurIPS*.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2024. Inspecting and editing knowledge representations in language models. In *COLM*.

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

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *ACL (Volume 1: Long Papers)*, pages 5591–5606. Association for Computational Linguistics.

Alonr Keinan. 2005. *Localization of function via multi-perturbation analysis: theory and applications for the analysis of neural networks*. Ph.D. thesis, Tel Aviv University, Israel.

Huiyuan Lai and Malvina Nissim. 2024. A survey on automatic generation of figurative language: From rule-based systems to large language models. *ACM Comput. Surv.*, 56(10):244.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342. Association for Computational Linguistics.

Daliang Li, Ankit Singh Rawat, Manzil Zaheer, Xin Wang, Michal Lukasik, Andreas Veit, Felix X. Yu, and Sanjiv Kumar. 2023. Large language models with controllable working memory. In *ACL Findings*, pages 1774–1793. Association for Computational Linguistics.

Jiahang Li, Taoyu Chen, and Yuanli Wang. 2024a. Trace and edit relation associations in GPT. *CoRR*, abs/2401.02976.

Xiaopeng Li, Shasha Li, Shezheng Song, Huijun Liu, Bin Ji, Xi Wang, Jun Ma, Jie Yu, Xiaodong Liu, Jing Wang, and Weimin Zhang. 2025. Swea: Updating factual knowledge in large language models via subject word embedding altering. *AAAI*, 39(23):24494–24502.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024b. PMET: precise model editing in a transformer. In *AAAI*, pages 18564–18572. AAAI Press.

AI @ Meta Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

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

Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. 2023. A survey on knowledge editing of neural networks. *CoRR*, abs/2310.19704.

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

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *ICLR*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *ICLR*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *ICML*, volume 162, pages 15817–15831. Proceedings of Machine Learning Research.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: An overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700, pages 193–209. Springer.

Shikhar Murty, Christopher D. Manning, Scott M. Lundberg, and Marco Túlio Ribeiro. 2022. Fixing model bugs with natural language patches. In *EMNLP*, pages 11600–11613. Association for Computational Linguistics.

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

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *ICLR*.

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. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. WISE: Rethinking the knowledge

memory for lifelong model editing of large language models. In *NeurIPS*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57.

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

Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. 2018. Interpret neural networks by identifying critical data routing paths. In *CVPR*, pages 8906–8914. Computer Vision Foundation/IEEE Computer Society.

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 *EMNLP*, pages 2875–2886. Association for Computational Linguistics.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2024. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *ICLR*. OpenReview.net.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. 2023. Unlearning bias in language models by partitioning gradients. In *ACL Findings*, page 6032–6048. Association for Computational Linguistics.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, pages 19449–19457. AAAI Press.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. In *EMNLP*, pages 22647–22662. Association for Computational Linguistics.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024b. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/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 *EMNLP*, pages 4862–4876. Association for Computational Linguistics.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2021. Modifying memories in transformer models.