CaKE: Circuit-aware Editing Enables Generalizable Knowledge Learners

Yunzhi Yao[♏,♓,*], **Jizhan Fang**[♏], **Jia-Chen Gu**[♓], **Ningyu Zhang**[♏,†],
Shumin Deng[ᘓ], **Huajun Chen**[♏,†], **Nanyun Peng**[♓,†],
[♏] Zhejiang University [ᘓ] National University of Singapore
[♓] University of California, Los Angeles
{yyztodd,fangjizhan,zhangningyu,huajunsir}@zju.edu.cn
gujc@ucla.edu,violetpeng@cs.ucla.edu,shumin@nus.edu.sg

Abstract

Knowledge Editing (KE) enables the modification of outdated or incorrect information in large language models (LLMs). While existing KE methods can update isolated facts, they often fail to generalize these updates to multi-hop reasoning tasks that rely on the modified knowledge. Through an analysis of reasoning circuits—the neural pathways LLMs use for knowledge-based inference, we find that current layer-localized KE approaches (e.g., MEMIT, WISE), which edit only single or a few model layers, inadequately integrate updated knowledge into these reasoning pathways. To address this limitation, we present **CaKE** (**C**ircuit-**a**ware **K**nowledge **E**diting), a novel method that enhances the effective integration of updated knowledge in LLMs. By only leveraging a few curated data samples guided by our circuit-based analysis, CaKE stimulates the model to develop appropriate reasoning circuits for newly incorporated knowledge. Experiments show that CaKE enables more accurate and consistent use of edited knowledge across related reasoning tasks, achieving an average improvement of 20% in multi-hop reasoning accuracy on the MQuAKE dataset while requiring less memory than existing KE methods.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in diverse tasks (Yang et al., 2024a; Dubey et al., 2024; OpenAI, 2024; Guo et al., 2025), achieving performance that rivals or even exceeds human experts. However, their practical deployment faces some critical limitations: parametric knowledge remains static after pretraining, making it challenging to keep up with evolving real-world information; their propensity for hallucinations also undermines reliability. Knowledge editing (KE) has emerged as a promising solution to update the knowledge in models precisely (Mitchell et al., 2021; Wang et al., 2024c; Jiang et al., 2025). Although existing KE methods achieve good results on simple factual updates (Yao et al., 2023; Zhang et al., 2024b), they often exhibit fundamental limitations: edits propagate inconsistently through related knowledge structures and downstream reasoning tasks (Cohen et al., 2024; Qin et al., 2024; Yao et al., 2023); excessive focus on surface-level pattern matching (Hoelscher-Obermaier et al., 2023), and locality issues for other unrelated knowledge and general ability (Gu et al., 2024; Gupta et al., 2024).

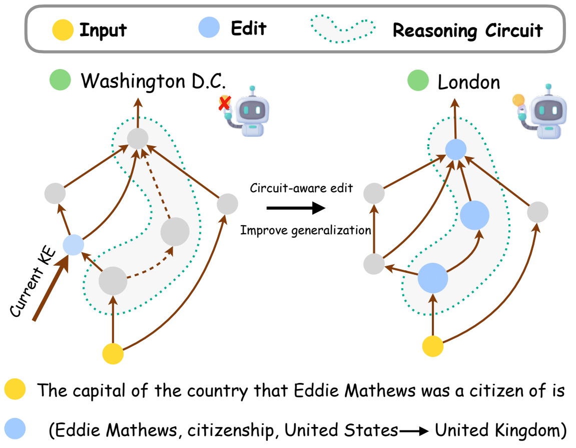
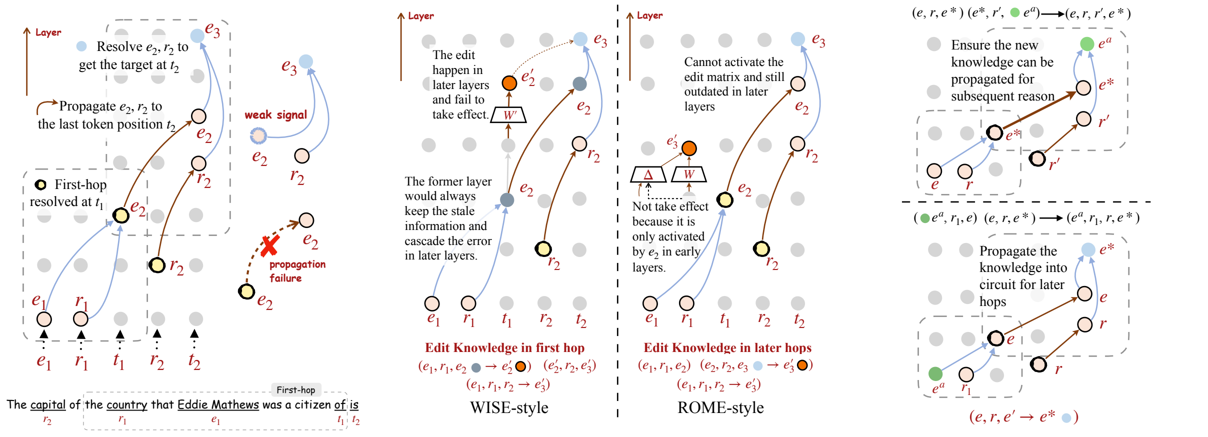

Input
Edit
Reasoning Circuit
Washington D.C.
London
Current KE
Circuit-aware edit
Improve generalization
The capital of the country that Eddie Mathews was a citizen of is
(Eddie Mathews, citizenship, United States ⟶ United Kingdom)

Figure 1: The current edit cannot propagate the new knowledge to the reasoning circuit for multi-hop reasoning. We propose a circuit-aware edit to improve the model's multi-hop reasoning performance involving the updated knowledge.

Our work specifically addresses the poor performance of edited models in downstream reasoning tasks that involve the updated knowledge (Zhong et al., 2023; Zhang et al., 2024d). Consider a representative case in Figure 1 : after editing '*Eddie Mathews, citizenship, United States → United Kingdom*', models correctly answer direct queries but fail multi-hop reasoning like '*The capital of the country that Eddie Mathews was a citizen of*

[*] Work done during Yunzhi's visit to UCLA.
[†] Corresponding Authors


Layer
Resolve e_2, r_2 to get the target at t_2
Propagate e_2, r_2 to the last token position t_2
First-hop resolved at t_1
e_3
e_2
r_2
e_1
r_1
t_1
t_2
weak signal
propagation failure
First-hop
The capital of the country that Eddie Mathews was a citizen of is
The edit happen in later layers and fail to take effect.
W'
The former layer would always keep the stale information and cascade the error in later layers.
e'_2
Edit Knowledge in first hop
$(e_1, r_1, e_2 \rightarrow e'_2)$ (e'_2, r_2, e'_3)
$(e_1, r_1, r_2 \rightarrow e'_3)$
WISE-style
Cannot activate the edit matrix and still outdated in later layers
Not take effect because it is only activated by e_2 in early layers.
Δ
W
e'_3
Edit Knowledge in later hops
(e_1, r_1, e_2) $(e_2, r_2, e_3 \rightarrow e'_3)$
$(e_1, r_1, r_2 \rightarrow e'_3)$
ROME-style
(e, r, e^*) $(e^*, r', e^a) \longrightarrow (e, r, r', e^*)$
Ensure the new knowledge can be propagated for subsequent reason
e^a
e^*
r'
e
r
(e^a, r_1, e) $(e, r, e^*) \longrightarrow (e^a, r_1, r, e^*)$
Propagate the knowledge into circuit for later hops
$(e, r, e' \rightarrow e^*)$

(a). Multi-hop reasoning circuit in LLMs: the second-hop reasoning process occurs at the last token position. Two primary reasons contribute to its failure: (1) weak e_2 signals, and, (2) fail to propagate the second-hop evidence e_2 to the final token position.

(b). Rethink the limitations of the most popular editing paradigms from a circuit perspective: both WISE-style (edits the later layers) and ROME-style (edits the early layers) fail to propagate their modifications through the reasoning circuit.

(c). Circuit-aware Knowledge Editing leverages additional features to curate reasoning tasks based on the updated knowledge, steering LLMs to construct a reasoning circuit that effectively processes the new information.

Figure 2: An overview of our work.

is?' (still outputting '*Washington D.C.*'). Critically, this is not merely an editing artifact: vanilla LLMs often correctly answer single-hop questions while failing their multi-hop counterparts (Yang et al., 2024b; Biran et al., 2024), suggesting deeper architectural limitations in knowledge utilization.

We trace these limitations to a misalignment between KE strategies and the inherent reasoning architectures of LLMs. To investigate this disconnect, we examine how LLMs leverage knowledge in downstream reasoning tasks. Recent analysis suggests that knowledge is not merely statically stored but dynamically activated through specialized circuits (Yao et al., 2024; Biran et al., 2024; Yu et al., 2025). However, these analyses overlook the phenomenon of LLM failures in reasoning circuits and fail to explore the underlying causes. Our investigation (§2) delves deeper into reasoning circuits, analyzing their structure and identifying the reasons behind failures in multi-hop reasoning. Specifically, the multi-hop reasoning emerges from coordinated computing circuits: early layers handle the first hop, extracting the bridge entity at the end-token of the first hop. This bridge entity, along with second-hop relation information, is then routed to the last token position in the middle layers. Subsequently, later layers utilize this information at the last token position to complete the reasoning process (Figure 2 (a)). We then analyze the entity and relation information at the last token position in failed multi-hop reasoning cases. Our observations reveal that critical information either fails to be properly routed to the last token position, or exhibits a weak signal, preventing effective reasoning. This explains why current KE methods underperform (§3.1): they optimize for isolated parameter changes rather than circuit-level integration needed for compositional reasoning (Figure 2b).

To bridge this fundamental gap, we propose **C**ircuit-**a**ware **K**nowledge **E**diting (CaKE) in §3.2. Unlike methods that only update localized static knowledge, CaKE actively constructs reasoning circuits that enable dynamic application of edited knowledge in downstream tasks. We first design circuit-aware training data that integrates across distinct segments of the reasoning process to force the LLM to leverage updated knowledge for latent reasoning (Figure 2c). Remarkably, we find that only a few such samples are sufficient to integrate knowledge across the reasoning circuit while maintaining strong general ability. Moreover, to prevent unintended data leakage, we construct these data using ad-hoc features (Zhang et al., 2024c) that are temporarily associated with the entities, such as '*Japan is colored green. The capital city of the country colored in green is*'. Finally, we guide LLMs to establish reasoning circuits by training with the curated data. Extensive experiments (§4) demonstrate CaKE's effectiveness: it outperforms existing knowledge editing methods on the MQuAKE multi-hop benchmark for both LLAMA3-8B-Instruct (Dubey et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2024a). Notably, CaKE achieves this while being more memory-efficient than alternatives and successfully scales to larger models like LLAMA3-70B-Instruct.

2 Analyzing Reasoning Circuit in LLM

2.1 Data Preparation

We employ the WikiData subset proposed by Biran et al. (2024) and name it *HoppingTooLate*, which contains 82,021 two-hop queries. We denote each fact as a triplet (e, r, e'), where e is the head entity, r is the relation, and e' is the tail entity. We view two-hop queries as (e_1, r_1, e_2) and (e_2, r_2, e_3), where e_1 is the source entity, e_2 is the bridge entity, and e_3 is the target entity. We focus on the latent reasoning framework to evaluate whether a model can output the expected answer e_3 directly given the composite query $(e_1, r_1, r_2, ?)$. For example, for the two facts *(Eddie Mathews, country_citizenship, United States)*,*(United States, capital, Washington D.C.)*, the composite query is *(Eddie Mathews, country_citizenship, capital,?)*. We transform the question into the natural language expression: '*The capital of the country that Eddie Mathews was a citizen of is?'*. In addition, we follow *HoppingTooLate* and define t_1 as the last token of the first-hop prompt (e.g., *'the country that Eddie Mathews was a citizen of'*) and t_2 as the last token of the whole two-hop prompt (e.g., '*The capital of the country that Eddie Mathews was a citizen of is')*.

2.2 Multi-hop Reasoning Circuit

Building on the insights from prior work (Biran et al., 2024; Yao et al., 2024), we can define a structured circuit mechanism for multi-hop reasoning in transformer-based LLMs, as illustrated in Figure 2(a). The three distinct computational phases: 1) The model processes the initial relation r_1 and entity e_1, encoding the bridge entity e_2 in the final token position of the first prompt segment (t_1). 2) Critical features, including e_2 and the second relation, r_2 are transferred to the last token position t_2, preparing for final resolution. 3) The model computes the target e_3 by resolving r_2 and e_2, giving the result in the final token position. Hence, based on the linearity theory (Hernandez et al., 2024), multi-hop reasoning in LLM can be formalized as:

$$F_n(F_{n-1}(e_{n-1}, r_{n-1}), r_n) \tag{1}$$

Each function F_{n-1} produces a bridge entity e_n for subsequent computation, demonstrating how intermediate results propagate vertically through network layers.

Model	Metric	Correct		Inconsistent		Incorrect	
		Cases	Layer	Cases	Layer	Cases	Layer
LLAMA3	e_2 from t_1	63.1%	6.3	75.2%	6.0	48.7%	8.2
	e_2 from t_2	67.8%	13.2	59.8%	9.8	17.7%	21.1
	r_2 from t_2	66.9%	14.0	49.0%	13.8	28.1%	13.7
	e_3 from t_2	56.5%	18.8	22.7%	20.7	18.3%	18.0
Qwen2.5	e_2 from t_1	71.2%	4.3	74.1%	4.7	46.7%	5.1
	e_2 from t_2	52.9%	7.9	63.7%	9.5	18.9%	13.5
	r_2 from t_2	75.8%	8.1	75.2 %	10.4	44.8%	9.7
	e_3 from t_2	71.2%	16.4	39.4%	17.4	25.2%	11.4

Table 1: The results of LLAMA3-8B-Instruct (32 layers) and Qwen2.5-7B-Instruct (28 layers). Cases are the percentage of data we can detect the information, and Layer is the mean of the earliest layer where the required information is detected.

2.3 Circuit in Failure Phenomena

Then, we aim to understand why language models sometimes fail at multi-hop reasoning despite successfully answering individual single-hop questions. For instance, a model may correctly answer *'the capital of Russia'* with *'Moscow'* and *'the country of citizenship of Fyodor Dostoyevsky'* with *'Russia'*, yet fail to answer the multi-hop question *'the capital of the country of citizenship of Fyodor Dostoyevsky is'* correctly. To systematically analyze this issue, we focus on the second hop of reasoning, as the model typically performs well on the first hop. We categorize the data from the *HoppingTooLate* dataset[1] into three subsets based on the model's behavior: **Correct**: The model answers both single-hop questions (e_1, r_1, e_2) and (e_2, r_2, e_3) correctly, as well as the multi-hop question $(e_1, r_1, r_2, ?)$. **Inconsistent**: The model answers both single-hop questions correctly but fails on the multi-hop question. However, we observe that some questions in the *Correct* set share the same 'bridge' entity e_2, even though they originate from distinct subject-relation pairs, that the model answers correctly. $(e'_1, r'_1, r_2, ?)$. This suggests that while the model can leverage knowledge in some contexts, it fails to generalize, indicating reasoning gaps rather than missing knowledge. **Incorrect**: The model answers both single-hop questions correctly but fails on the multi-hop question in all contexts (e'_1, r'_1, e_2). This implies a complete failure to employ the knowledge for multi-hop reasoning. To investigate these failure modes, we check whether the models construct the reasoning circuit by monitoring key variables (e_1, e_2, and r_2) at crit-

[1] We filter out short-cut cases as done by Biran et al. (2024).

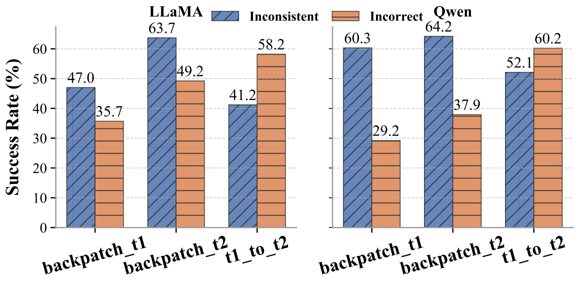


Figure 3: Results of the intervention on the failure cases in multi-hop reasoning of LLAMA3 and Qwen2.5.

ical positions (t_1 and t_2) across the model's layers using the PatchScope as Biran et al. (2024) did. Our analysis reveals several interesting patterns, extending beyond the 'hopping too late' problem identified by Biran et al. (2024).

We list the results in Table 1 (more details in Figure 6 in the Appendix). For the *correct* subset, we observe strong evidence of the reasoning circuit functioning as expected: a large portion of e_2 is detected at both t_1 (e_2 from t_1) and t_2 (e_2 from t_2) in both LLAMA3 and Qwen2.5 models. The model correctly uses the r_2 and e_2 information at t_2 to produce the final answer e_3. Contrastly, in the *Incosistent* subsets, we can find that despite detecting e_2 and r_2 at t_2, the model often fails to produce the correct e_3 answer (e_3 from t_2: only 22.7% in LLAMA3 and 39.4% in Qwen2.5 of cases we can detect at t_2). We hypothesize that the e_2 information, though present, may be insufficient to trigger the second-hop reasoning circuit, leading to the failure to execute the function $F(e_2, r_2)$ effectively. What's more, in the *Incorrect* subsets, we can find that the needed e_2 information is rarely detected at the t_2 position (e_2 from t_2: Only 17.7% in LLAMA3 and 18.9% in Qwen2.5). Even when e_2 is detected, it typically emerges in much later layers (layer 21 in LLAMA3 and layer 13.5 in Qwen2.5), making it too late to be effectively utilized for the second-hop computation, aligned with Biran et al. (2024)'s findings. We conjecture the model fails to propagate e_2 to the t_2 position, resulting in the variable e_2 missing for conducting the $F(e_2, r_2)$ function.

Evaluation To test our hypothesis, we conduct interventions to enhance the information at the detected layers (details in Figure 6) to see if we can improve the model's performance in these failure cases. We test three ways: back-patching the t_1 and t_2 position as Biran et al. (2024) did, which would enhance the information at the position, and cross-position patching the information from t_1 to the t_2 position, which explicitly propagates the information from t_1 to t_2 (details in Figure 10 in Appendix). From the results in Figure 3, we can find a high success rate for all the inconsistent and incorrect cases, but they demonstrate different paradigms. For the inconsistent cases, back-patching would lead to better performance, while for the incorrect cases, patching knowledge from the t_1 to t_2 usually shows better outcomes. This proves our previous hypothesis that for the incorrect cases, due to the **propagation failure**, the model fails to move the e_2 to t_2 position, and manual routing via cross-patching can mitigate the issue. Meanwhile, for inconsistent cases, amplification via back-patching compensates the **weak signal** when valid e_2 representations reach t_2 but lack sufficient magnitude for subsequent reasoning.

3 Circuits-aware Knowledge Editing

Building on our previous reasoning analysis, we rethink the reason why current knowledge editing methods fail under multi-hop reasoning circumstances despite their great performance under single-fact editing.

3.1 Rethinking KE from the Circuit View

Here, we aim to figure out what happens when we edit the model with the current KE methods.

Unified Editing Details When updating a piece of knowledge $(e, r, o \rightarrow o')$, the most popular knowledge editing techniques would modify the parameters that are responsible for the knowledge. There are two kinds of paradigms: editing the Feed-Forward Networks (FFN) in the early layers, such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) or modifying the later layers' FFN output, like WISE (Wang et al., 2024c) and T-Patcher (Huang et al., 2023). This is mainly based on the key-value memory features of the FFN (Geva et al., 2020). However, some studies have queried the effectiveness of these localization settings (Chang et al., 2024; Hase et al., 2024) as the localization area is not correlated to the performance of the knowledge editing methods. Here, we propose a unified view of the mechanisms and limitations from the circuit perspective. ROME-style would modify the weight $\boldsymbol{W}$ with a perturbation $\boldsymbol{\Delta}$ and obtain a new weight $\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{\Delta}$. When calculating the Δ, ROME-style methods, apply the *least squares estimation* and *null space constraint* to make sure the Δ is only activated by the corresponding entity representation $\boldsymbol{e}_{in}$ and keep the original output for other representations. In parallel, WISE-style editing methods would directly introduce the new weight $\boldsymbol{W}'$ that would be activated by the related representation $\boldsymbol{e}_{in}$, and $\boldsymbol{W}'$ would encode the updated knowledge.(More details in Appendix B.2). Hence, these two editing paradigms can be represented uniformly by a gated function $\mathcal{G}(\cdot)$:

$$\text{FFN}_{\text{out}}(\mathbf{x}) = \underbrace{\boldsymbol{W}\mathbf{x}}_{\text{Original term}} + \mathcal{G}(\mathbf{x}) \cdot \underbrace{\boldsymbol{\delta}(\mathbf{x})}_{\text{Edit term}} \tag{2}$$

$$\mathcal{G}(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \boldsymbol{e}_{in} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Here, for ROME-style method,$\boldsymbol{\delta}(\mathbf{x}) = \boldsymbol{\Delta}\mathbf{x}$ and for WISE-style method, $\boldsymbol{\delta}(\mathbf{x}) = (\boldsymbol{W}' - \boldsymbol{W})\mathbf{x}$. When the gating function $\mathcal{G}(\cdot)$ is activated by the input $\mathbf{x}$, the edit term $\boldsymbol{\delta}(\mathbf{x})$ is applied, thereby modifying the knowledge within the computational circuit.

Defect from circuit view In single-hop knowledge editing, both these kinds of methods would give us the correct information, but for the multi-hop cases, they would fail. As shown in Figure 2 (b), both these layer-specific editing methods **cannot propagate the updated knowledge to the reasoning circuit**, leading to unsatisfactory multi-hop reasoning performances. Consider the two-hop reasoning process from §2: the model must first correctly compute $e_2 = F_1(\langle e_1, r_1 \rangle)$ in early layers. The representation of e_2 then propagates to the final token position t_2 (typically where answers are generated), where it combines with r_2 to compute $e_3 = F_2(\langle e_2, r_2 \rangle)$ in later layers.

WISE-style editing shows critical limitations when handling first-hop facts $(e_1, r_1, e_2 \to e_2')$ in multi-hop reasoning. As the edit is applied to later layers, the early layers remain unchanged and continue to produce the original e_2 representation during computation. This creates a fundamental mismatch: while the later layers perform the second-hop computation $F_2(\langle e_2, r_2 \rangle)$, they operate on the unmodified e_2 from early layers. Consequently, the gating mechanism $\mathcal{G}(\cdot)$ designed for first-hop edits becomes effectively bypassed in the reasoning process. Similarly, ROME-style editing fails when the edited fact $(e_2, r_2, e_3 \to e_3')$ serves as the second-hop question. For the edit to take effect, the gating function $\mathcal{G}(\cdot)$ must be activated by e_2 in early layers. However, e_2's representation only appears after the first hop completes in the computational pathway - potentially after the edited layers. In this scenario, the gated function $\mathcal{G}(\mathbf{x})$ in earlier layers remains unactivated, causing the model to default to stale knowledge and produce incorrect answers.

To systematically evaluate these limitations, we conduct experiments on editing different positions in multi-hop questions. Table 6 reveals that WISE achieves poorer performance when editing the first hop versus the second hop. Conversely, when using ROME to edit the second hop, the performance is worse than when editing the first hop. These results demonstrate that layer-specific editing methods fundamentally lack the ability to generalize updated knowledge for downstream reasoning tasks.

3.2 Proposed Method: CaKE

Inspired by previous analysis, we propose a novel method, **C**ircuit-**a**ware **K**nowledge **E**diting (**CaKE**), which makes sure the models build the reasoning circuit with the updated knowledge. As we show in the previous section, a successful reasoning circuit is one that, after editing the model's knowledge, ensures: The updated computation F_1 or F_2 accurately reflects the new knowledge, and the bridge entity e_2 is correctly computed and propagated to t_2. Hence, simply editing a single layer or several layers is not enough to enable the circuit for reasoning. Here, CaKE comprises two key components: (1) generating circuit-aware data that explicitly requires reasoning with the updated knowledge, and (2) training the model to construct robust reasoning circuits that integrate the new knowledge.

Data Generation For each updated knowledge item, we construct the following contexts to mitigate these issues: (1) **Original Narrative**: We begin by generating straightforward factual statements that explicitly convey the updated information. For example, when updating the fact k: (PersonX, citizen_country, Switzerland → Japan), we use the narrative representation: *'PersonX is a citizen of Japan'* and generate several paraphrases. These statements serve as the foundation for the model to learn the updated knowledge. (2) **Circuit-aware Tasks**: Next, we design specialized reasoning scenarios that address two critical challenges: preventing *failure propagation* and mitigating *weak signals*, while ensuring that updated knowledge is properly integrated across different layers (in Figure 2c). Moreover, to avoid introducing extrane-

Method	Model	MQUAKE-CF		MQUAKE-CF-v2		MQUAKE-T	
		H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑
Pre-edited		79.0	27.0	78.4	28.6	71.0	5.3
AdaLoRA	LLaMA3-8B-Ins	66.0	27.6	64.7	24.6	**92.3**	66.0
WISE		38.2	24.0	37.2	21.0	63.5	62.9
MeLLo		16.5	16.1	19.5	16.0	42.3	50.1
ROME		86.8	17.6	86.4	15.5	89.5	8.4
MEMIT		76.3	11.5	74.0	10.0	86.0	3.7
AlphaEdit		66.1	10.1	63.7	8.5	73.4	1.0
IFMET ♣		81.9	23.2	75.3	36.5	82.1	46.1
CaKE(ours)		**90.6**	**57.3**	**90.1**	**57.1**	91.5	**81.4**

Table 2: Comparison of CaKE with existing methods on MQuAKE for LLAMA3-8B-Instruct. The best results are highlighted in bold, while the second-best results are underlined. ♣ means the results are based on our re-implementation since the original code is not open by the authors, and we will update it after the source code is open.

ous knowledge that could leak into downstream evaluations—and to test the generalization of our method (inspired by prior research (Zhang et al., 2024c))—we incorporate ad-hoc features into these scenarios. Particularly, these tasks link the facts with intermediate attributes or reasoning steps and fall into two categories: **Late-layer Knowledge Integration:** These tasks ensure that the updated knowledge is effectively learned in the later layers, alleviating issues such as *weak signals and the limitations of ROME-style editing*. Take the fact k: (PersonX, citizen_country, Switzerland $\rightarrow$ Japan) as an example; we construct a seed prompt like: *'Suppose {random_entity_1} wears red clothes, {random_entity_2} wears blue clothes, and {PersonX} wears green clothes. The country of citizenship of the person in green is:'* Here, the model is expected to output 'Japan,' requiring it to employ the new fact k in later layers. **Reasoning Circuit Enhancement:** These tasks require the model to use the updated knowledge for subsequent reasoning, thereby mitigating *propagation failure* and *WISE-style's limitations*. Following the same fact k, the seed prompt is *'In a book about countries, Japan is mentioned on page 6 of the book, while China is mentioned on page 72. On which page of the book is the country of citizenship of the {PersonX} shown?'* Here, the model must first recall the updated citizenship (Japan) and then use this information to determine the correct page number (6).

For each relation type, we design these seed task templates and employ GLM-4-plus (GLM et al., 2024) to generate diverse expressions following these templates (see Appendix A for details). Specifically, we create 3 distinct samples per category for each edited fact, which our experiments show are sufficient to enable effective reasoning with the updated knowledge. This minimal data requirement demonstrates the efficiency of our approach in adapting models to new information.

Edit Training After obtaining the curated circuit-aware data $\mathcal{D}$, we fine-tune the LLM using LoRA, enabling the model to optimize its internal knowledge organization. We minimize the cross-entropy loss $\mathcal{L}$ between the model's outputs and the ground-truth tokens expressing the updated fact:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\left[-\sum_{t=1}^{|\mathbf{y}|}\log p(y_t \mid \mathbf{x}, \theta_{\text{LoRA}})\right] \quad (4)$$

where θ_{LoRA} represents the LoRA parameters, $\mathbf{x}$ is the input prompt, and $\mathbf{y}$ is the desired updated output sequence.

4 Experiments

4.1 Experiment Settings

We mainly utilize the multi-hop reasoning knowledge editing dataset MQuAKE (Zhong et al., 2023), which considers different numbers of hops (from 2 to 4) and different positions of the knowledge used in the multi-hop questions. We utilize three versions of the datasets: MQuAKE-CF-3k and MQuAKE-CF-3k-v2, which are two subsets that contain different question types and editing hopping numbers, and MQuAKE-T is a time-aware knowledge editing benchmark.

Baselines and Models We consider several knowledge editing baselines, including: IFMET (Zhang et al., 2024d), AlphaEdit (Fang et al., 2024), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023),WISE (Wang et al., 2024c) and MeLLo (Zhong et al., 2023). Here, AlphaEdit, ROME, and MEMIT are methods that edit the model's parameters at early layers; WISE adds additional parameters at later layers, and IFMET edits both the early and later layers' FFN to achieve better multi-hop reasoning performance. MeLLo is a prompt-based retrieval-augmented method that keeps the model's parameters unchanged. We conduct experiments on LLAMA-3-8B-Instruct, Qwen-2.5-7B-Instruct, and LLAMA-3-70B-Instruct.

Evalutation Metric Following Zhong et al. (2023), we evaluate model performance using Multi-hop Accuracy (MAcc) and Hop-wise Answering Accuracy (H-Acc). MAcc measures the accuracy of multi-hop question answering, while H-Acc assesses correctness at each reasoning step. For both metrics, we consider a prediction correct if the ground-truth answer appears in the generated text as Cohen et al. (2024); Zhong et al. (2023) did. Higher values indicate better reasoning capability. For KE, we also need to consider locality, which ensures edits do not affect unrelated knowledge and abilities. To assess this, we evaluate the model on general benchmarks, including CommonsenseQA (Talmor et al., 2019), BigBench-Hard (Suzgun et al., 2023), MMLU (Hendrycks et al., 2021), and GSM8k (Cobbe et al., 2021).

4.2 Experiments Results

Main Results We show the results for LLAMA3-8B-Instruct in Table 2 and Qwen2.5-7B-Instruct in Table 7. From the table, we can find that although current KE methods achieve high hop-wise accuracy (H-Acc.), their performance on the three versions of MQuAKE is quite low (with an average accuracy of less than 20%). For example, MEMIT and ROME achieve over 80% accuracy on single-hop questions in MQuAKE-v2; however, their accuracy on multi-hop reasoning drops to only around 10%, indicating that the LLM fails to effectively utilize the updated knowledge during reasoning. In contrast, CaKE demonstrates significant improvements in multi-hop reasoning. In the LLAMA3-8B-Instruct model, CaKE achieves accuracies of 57.3, 57.2 and 81.5 in MQuAKE-CF, MQuAKE-CF-v2 and MQuAKE-T, respectively, outperforming all the compared methods. Additionally, IFMET, which also considers different layers for multi-hop reasoning but neglects the information flow within the circuit, performs not as well as CaKE. Moreover, when compared with RAG-based methods such as MeLLo, CaKE also yields better results. Furthermore, compared to the baseline LoRA tuning methods that simply incorporate the raw knowledge, the improvements observed with CaKE underscore the effectiveness of our approach. Results in Qwen-2.5-Instruct also demonstrate the same phenomenon.

	CSQA	BBH	MMLU	GSM8k
LLaMA3-8B-Ins	76.09	67.89	63.83	75.20
MEMIT	76.08	67.88	63.82	75.21
ROME	72.98	61.37	62.95	74.59
CAKE	75.10	67.20	62.98	76.04
Qwen2.5-7B-Ins	82.31	33.39	71.80	82.26
MEMIT	82.39	37.37	71.80	81.96
ROME	72.57	34.22	63.38	72.21
CAKE	82.64	37.44	71.76	82.79

Table 3: **Locality Performance** on several general benchmarks of CaKE and other editing methods.

Position and Number of Hop We also compare the performance on different hops and positions in Figure 9. Even when the model is trained solely on two-hop questions, CaKE yields improvements across varying numbers of editing hops. The benefits are particularly pronounced for four-hop questions, where methods like IFMET (designed only for two-hop scenarios) struggle. Besides, CaKE enhances performance regardless of the position of the edited knowledge within the multi-hop questions, demonstrating the generalizability of CaKE.

Efficiency and Scalability We evaluate computational efficiency in Table 10, demonstrating that CaKE achieves better performance while requiring less memory than MEMIT with comparable editing times. This efficiency advantage enables CaKE to scale effectively to larger models, as evidenced in Table 8, where it maintains superior performance on LLAMA3-70B-Instruct.

5 Analysis

5.1 Locality Performance

In this section, we evaluate the model's performance on general ability benchmarks to ensure that acquiring new knowledge does not compro-

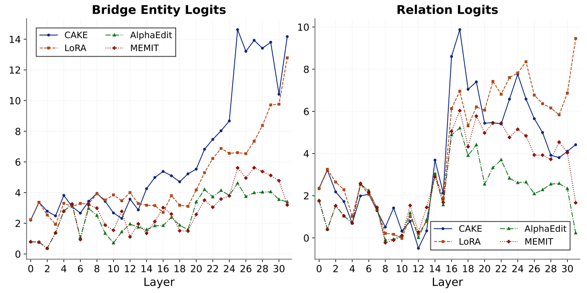


Figure 4: e_2 and r_2's logits at t_2 in models after different knowledge editing methods.

mise its overall capabilities. As shown in Table 3, CaKE achieves performance comparable to the original model on both the LLAMA3-8B and Qwen2.5-7B models across different kinds of tasks, including math, commonsense, and diverse understanding tasks.

5.2 Case Analysis

In this part, we show the cases in which the **CaKE helps the model learn the multi-hop reasoning circuit and other methods fail**. For illustration, we consider the two-hop question: 'The capital city of the country that Eddie Mathews was a citizen of is'. Here, the editing case is *(Eddie Mathews, citizenship, United States → United Kingdom)*, and the updated model is expected to output 'London'. However, CaKE gives the correct answer, while other methods fail: MEMIT gives us the 'Moscow', AlphaEdit gives us 'Birmingham', and LoRA gives us 'not known'. To further understand these differences, we analyze the computing circuit of each method to determine *whether the updated model successfully propagates the bridge entity e_2 and relation r_2 to the last token t_2 position.*

Figure 4 compares the logits of e_2 and r_2 at position t_2 across different editing methods. Here, CaKE generates significantly stronger logits for the bridge entity e_2 compared to AlphaEdit and MEMIT. This demonstrates CaKE 's ability to propagate critical information to target positions for subsequent reasoning steps. Similarly, CaKE produces more prominent r_2 logits, indicating more robust circuit construction and information flow compared to baseline methods.

6 Related Work

Knowledge Learning and Editing Knowledge editing (Lampinen et al., 2025; Jiang et al., 2024a; Sun et al., 2024; Hsueh et al., 2024; Powell et al., 2024; Wang et al., 2024a; Rozner et al., 2024; Zhang et al., 2024a; Wang et al., 2024e; Shi et al., 2024; Huang et al., 2024b; Guo et al., 2024; Wang et al., 2025; Feng et al., 2025; Yang et al., 2025; Li et al., 2024b; Huang et al., 2024a) has emerged as a promising approach for updating models in an ever-changing world. Current knowledge editing methods typically follow one of several strategies: modifying the MLP components in earlier layers (Meng et al., 2022, 2023), enhancing the MLP in later layers (Hartvigsen et al., 2023), or retrieving relevant facts as prompts (Jiang et al., 2024b; Zhong et al., 2023). However, most existing knowledge editing techniques concentrate on simple factual updates and frequently fail to generalize to more complex downstream tasks, such as multi-hop reasoning scenarios.

Model Interpretability Knowledge editing is primarily based on the intrinsic knowledge mechanisms of neural models' "black boxes" (Ferrando et al., 2024). Consequently, understanding how knowledge in LLMs is acquired and stored has garnered significant attention (Wang et al., 2024b). Recent studies (Zhou et al., 2023) demonstrate that most knowledge is learned during the pretraining stage and is predominantly stored in the Feed-Forward Network (Geva et al., 2020). Beyond these localized findings, researchers (Geva et al., 2023; Yao et al., 2024) have investigated the computational circuits—the pathways connecting Transformer components—to elucidate how LLMs perform knowledge recall. Building on this, subsequent work has explored the relationship between knowledge editing and these circuits (Ge et al., 2024). In contrast, our work focuses on the mechanisms underlying multi-hop reasoning in LLMs and aims to improve the generalization of edited knowledge.

7 Conclusion

In this paper, we identify that existing knowledge editing methods fall short due to their isolated parameter adjustments by examining the multi-hop reasoning circuits within LLMs. We present CaKE , a method designed to align knowledge editing with the inherent reasoning architectures of LLMs. CaKE incorporates circuit-aware tasks that compel the model to dynamically integrate and utilize new knowledge during reasoning. Experimental results demonstrate that CaKE achieves generalizable multi-hop knowledge editing.

Limitation

Dataset Our work primarily focuses on the factual knowledge embedded in large language models (LLMs) and their capacity for multi-hop reasoning over these facts. We recognize that LLM reasoning also encompasses other domains—such as long-form mathematics and reverse-curse reasoning—that merit further investigation.

Reasoning Pattern As discussed in the previous analysis, we concentrate on direct reasoning phenomena. Current LLMs have shown impressive capabilities in slow-thinking paradigms, including chain-of-thought and reflective reasoning. Beyond direct reasoning, enhancing the utilization of knowledge within these paradigms represents an important avenue for future research.

Fine-grained Circuit Components Our analysis revealed relational information within the circuits; however, CaKE currently does not delve deeply into these relationships. We believe that a more focused investigation into these components is necessary. Additionally, while our study emphasizes general circuit behavior, developing a more concise and effective method for knowledge editing remains an exciting challenge for future work.

Data Attribution Although we demonstrate the ability to construct reasoning circuits using curated data, the connection between a model's acquired abilities in its parameters and its training data is still underexplored. A deeper understanding of this relationship could lead to more efficient training processes and the generation of higher-quality synthetic data.

References

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14113–14130, Miami, Florida, USA. Association for Computational Linguistics.

Ting-Yun Chang, Jesse Thomason, and Robin Jia. 2024. Do localization methods actually localize memorized data in llms? a tale of two benchmarks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3190–3211.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 11:283–298.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Yujie Feng, Liming Zhan, Zexin Lu, Yongxin Xu, Xu Chu, Yasha Wang, Jiannong Cao, Philip S Yu, and Xiao-Ming Wu. 2025. Geoedit: Geometric knowledge editing for large language models. *arXiv preprint arXiv:2502.19953*.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*.

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. 2024. A framework for few-shot language model evaluation.

Huaizhi Ge, Frank Rudzicz, and Zining Zhu. 2024. What do the circuits mean? a knowledge edit view. *arXiv preprint arXiv:2406.17241*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16801–16819.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Phillip Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite. 2024. Mechanistic unlearning: Robust knowledge unlearning and editing via mechanistic localization. *arXiv preprint arXiv:2410.12949*.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. 2024. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15202–15232, Bangkok, Thailand. 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. *Advances in Neural Information Processing Systems*, 36.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559.

Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. 2023. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4902–4919.

Cheng-Hsun Hsueh, Paul Kuo-Ming Huang, Tzu-Han Lin, Che-Wei Liao, Hung-Chieh Fang, Chao-Wei Huang, and Yun-Nung Chen. 2024. Editing the mind of giants: An in-depth exploration of pitfalls of knowledge editing in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 9417–9429. Association for Computational Linguistics.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. 2024a. Can knowledge editing really correct hallucinations? *arXiv preprint arXiv:2410.16251*.

Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. 2024b. Reasons and solutions for the decline in model performance after editing. *Advances in Neural Information Processing Systems*, 37:68833–68853.

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

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024a. Learning to edit: Aligning llms with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 4689–4705. Association for Computational Linguistics.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, Qun Liu, and Wei Wang. 2024b. Learning to edit: Aligning LLMs with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4705, Bangkok, Thailand. Association for Computational Linguistics.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8987–9001, Bangkok, Thailand. Association for Computational Linguistics.

Andrew K Lampinen, Arslan Chaudhry, Stephanie CY Chan, Cody Wild, Diane Wan, Alex Ku, Jörg Bornschein, Razvan Pascanu, Murray Shanahan, and James L McClelland. 2025. On the generalization of language models from in-context learning and finetuning: a controlled study. *arXiv preprint arXiv:2505.00661*.

Ming Li, Yanhong Li, and Tianyi Zhou. 2024a. What happened in llms layers when trained for fast vs. slow thinking: A gradient perspective. *arXiv preprint arXiv:2410.23743*.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024b. Understanding and patching compositional reasoning in llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9668–9688.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

OpenAI. 2024. Introducing OpenAI O1 preview.

Derek Powell, Walter Gerych, and Thomas Hartvigsen. 2024. TAXI: evaluating categorical knowledge editing for language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 15343–15352. Association for Computational Linguistics.

Jiaxin Qin, Zixuan Zhang, Chi Han, Pengfei Yu, Manling Li, and Heng Ji. 2024. Why does new knowledge create messy ripple effects in llms? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12602–12609.

Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. 2024. Knowledge editing in language models via adapted direct preference optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 4761–4774. Association for Computational Linguistics.

Yucheng Shi, Qiaoyu Tan, Xuansheng Wu, Shaochen Zhong, Kaixiong Zhou, and Ninghao Liu. 2024. 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 2024, Boise, ID, USA, October 21-25, 2024*, pages 2056–2066. ACM.

Zengkui Sun, Yijin Liu, Jiaan Wang, Fandong Meng, Jinan Xu, Yufeng Chen, and Jie Zhou. 2024. Outdated issue aware decoding for factual knowledge editing. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9282–9293. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *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.

Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. 2024a. Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 996–1008. Association for Computational Linguistics.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, et al. 2024b. Knowledge mechanisms in large language models: A survey and perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7097–7135.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024c. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024d. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand. Association for Computational Linguistics.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2025. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3):59:1–59:37.

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

Zhiwei Wang, Yunji Wang, Zhongwang Zhang, Zhangchen Zhou, Hui Jin, Tianyang Hu, Jiacheng Sun, Zhenguo Li, Yaoyu Zhang, and Zhi-Qin John Xu. 2024f. Towards understanding how transformer perform multi-step reasoning with matching operation. *arXiv preprint arXiv:2405.15302*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Sohee Yang, Nora Kassner, Elena Gribovskaya, Sebastian Riedel, and Mor Geva. 2024b. Do large language models perform latent multi-hop reasoning without exploiting shortcuts? *Preprint*, arXiv:2411.16679.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. 2025. The mirage of model editing: Revisiting evaluation in the wild. *arXiv preprint arXiv:2502.11177*.

Xinhao Yao, Ruifeng Ren, Yun Liao, and Yong Liu. 2025a. Unveiling the mechanisms of explicit cot training: How chain-of-thought enhances reasoning generalization. *arXiv preprint arXiv:2502.04667*.

Yunzhi Yao, Canyu Chen, Jia-Chen Gu, Shumin Deng, Manling Li, and Nanyun Peng. 2025b. Reflection on knowledge editing: Charting the next steps. `https://yyzcowtodd.cn/rethinkedit`. Notion Blog.

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

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *Advances in Neural Information Processing Systems*.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. Back attention: Understanding and enhancing multi-hop reasoning in large language models. *arXiv preprint arXiv:2502.10835*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, 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, et al. 2024b. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Xiao Zhang, Miao Li, and Ji Wu. 2024c. Co-occurrence is not factual association in language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. 2024d. Locate-then-edit for multi-hop factual recall under knowledge editing. *arXiv preprint arXiv:2410.06331*.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: less is more for alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Appendix

A Setting Detail

Dataset We list the details of the dataset in Table 4.

Model	Correct	Inconsistent	Incorrect
LLaMA3-8B-Ins.	1,005	1,032	1,240
Qwen2.5-7B-Ins.	241	252	275

Table 4: The dataset we used in the analysis.

Environment Setting We run our experiments on 2 NVIDIA-A800 GPUs. For data generation, we utilize glm-4-plus and glm-4-air and a total of 10,000,000 tokens (about 20 dollars) to generate all synthetic data for the whole 7,867 data samples. The cost is approximately 0.002 dollars per edit, which also demonstrates the efficiency of CaKE method. We use LLM-Eval (Gao et al., 2024) to test the model's general performance.

Data Generation We first construct the question template $\mathcal{T}$ for each relation type, and we list some of them in Table 5. We then generate the data using the following prompt:

> **Prompt for Constructing the circuit-aware data**
>
> Here are some question templates for the specific relation. As you can see, the question use the knowledge in the input to conduct reasoning in different hops for multi-hop reasoning. Please generate 3 different questions that share the same features as the template. Please return a python json file. $\{\mathcal{T}\}$ Here is the input question:

It should be noted that we do not ask the model to strictly follow the expression of the template, and we also show some data samples in Appendix A to show the diversity of the generated data.

B Implementation Detail

B.1 Analyzing Method

Patch Scope The process is carried out as follows. First, a source prompt, a source token, and a source layer are provided. The prompt is processed through the model's forward computation, and the hidden representation v of the source token at the specified layer is extracted and stored. This representation v is the focus of our investigation, as we seek to determine whether it encodes a specific entity. Next, we employ the same prompt used by Ghandeharioun et al. (2024): "Syria: Syria is a country in the Middle East. Leonardo DiCaprio: Leonardo DiCaprio is an American actor. Samsung: Samsung is a South Korean multinational corporation. x" This prompt is passed through the model, but the hidden representation of 'x' is replaced with v at a chosen target layer. The forward computation then proceeds, and the resulting generated text is analyzed to evaluate the effects of this substitution. We conduct different patch analyses and show them in Figure 8 and Figure 10. When we conduct back-patch and cross-patch, the source prompt and target prompt are the same.

B.2 Editing Method

We utilize EasyEdit (Wang et al., 2024d) to conduct our editing experiments. For ROME, MEMIT, WISE, AlphaEdit, and MeLLo, we directly employ the original parameters provided by their respective papers. Below, we introduce these methods in detail and describe our implementation.

ROME and MEMIT ROME leverages causal analysis to identify knowledge within specific MLP layers and modifies the corresponding weight matrix using least squares approximation. It operates under the strong assumption that the MLP layers primarily store knowledge and injects new information into these layers iteratively using a Lagrangian remainder. In our experiments, we edit the 5th layer of both LLAMA3-8B-Instruct and Qwen2.5-7B-Instruct.

Similarly, MEMIT assumes that the FFN layers function as a knowledge key-value store. It directly modifies the parameters of selected layers through least squares approximation. Unlike ROME, which updates a single layer, MEMIT is a multi-layer editing algorithm capable of simultaneously updating hundreds or thousands of facts

IFMET IFMET builds upon MEMIT by not only modifying earlier MLP layers in transformers but also adjusting later layers to enhance multi-hop reasoning for the edited knowledge. To ensure the updated knowledge propagates effectively, IFMET constructs an additional support set that reinforces learning in later layers. Based on our analysis in §2, we edit layers [17,18,19,20] for LLAMA3-8B-Instruct and layers [15,16,17,18] for Qwen2.5-7B-Instruct.

Knowledge Type	**Template**	**Answer**
{target_person} works in the field of {target_field}	In a book related to different fields, Section A discusses {random_field}, Section B discusses {random_field}, and Section C discusses {target_field}. If you want to learn about {target_person}'s field, which section should you read?	The working field of {target_person} is discussed in Section C.
	In a biography book, Section A discusses the life of {random_person}, Section B discusses the life of {random_person}, and Section C discusses the life of {target_person}. The field of the person in Section C is?	The person in Section C works in the field of {target_field}.
{target_person} speaks the language of {target_language}.	The following facts are known: 1. {target_person} wears red clothes. 2. {random_person} wears blue clothes. 3. {random_person} wears green clothes. The language that the person in red clothes speaks is?	The language that the person in red clothes speaks is {target_language}.
	At a global company: {target_language}-speaking employees work in Team A. {random_language}-speaking employees work in Team B. In which team would {target_person} work when he/she is at work?	{target_person} would work in Team A when he/she is at work.

Table 5: Sample templates for generating the circuit-aware data.

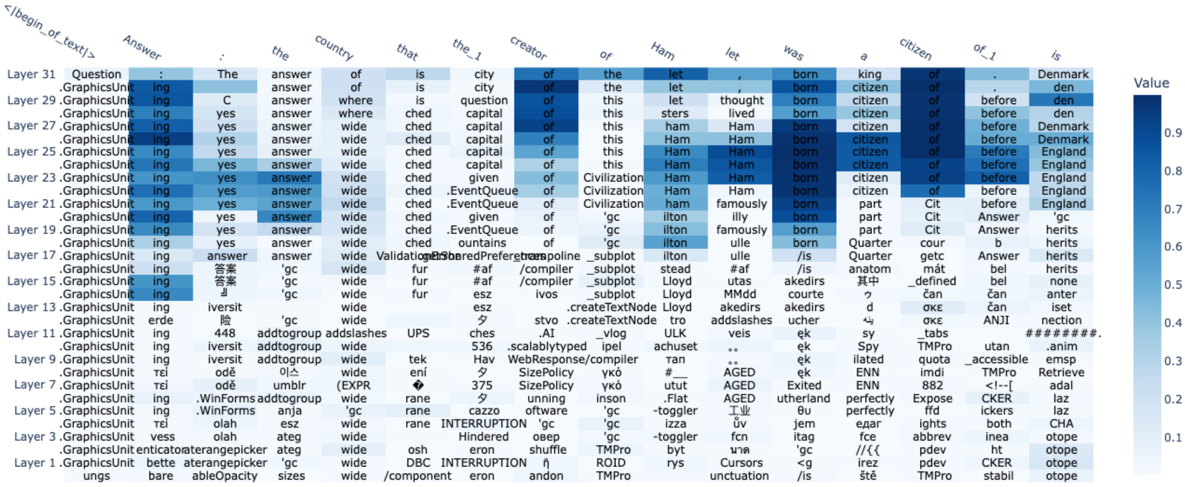


Figure 5: The failure case of the multi-hop reasoning.

WISE WISE represents a different approach to model editing, focusing on later layers instead of earlier ones. It modifies the model's FFN output using a gating mechanism:

$$\mathrm{FFN}_{\mathrm{out}}(\mathbf{x}) = \begin{cases} \mathcal{G}(\mathbf{x}) \cdot \mathbf{W}_{v'} & \text{if } \mathcal{G}(\mathbf{x}) > \epsilon, \\ \mathcal{G}(\mathbf{x}) \cdot \mathbf{W}_{v} & \text{otherwise.} \end{cases} \tag{5}$$

Here, $\mathcal{G}(\mathbf{x})$ is a gate function that computes the activation score of the hidden reprsentation: $\|\mathcal{A}(\mathbf{x}) \cdot (\mathbf{W}_{v'} - \mathbf{W}_{v})\|_2$. If the gate is activated, the model uses the updated knowledge to generate responses; otherwise, it relies on the original knowledge. Different methods define the gate function differently, but the core idea is to ensure that the updated memory aligns with relevant question representations.

Edit Method	**LLAMA3-8B**		**Qwen2.5-7B**	
	First_hop	Second_hop	First_hop	Second_hop
ROME	**16.66**	7.81	**10.57**	8.33
WISE	49.85	**67.36**	8.33	**33.59**

Table 6: Performance comparison of edit methods across different positions for the edited fact.

MeLLo MeLLo is a non-parametric editing method that modifies a model's knowledge through prompting rather than weight updates. It maintains a memory of newly introduced facts and guides the model to decompose multi-hop queries into sub-questions. At each step, the model checks this memory to verify whether its existing knowledge contradicts the new facts. We follow the prompt structure provided in the original MeLLo method. However, in our experiments, we observe that the model struggles to consistently adhere to the intended reasoning pattern.

CaKE We utilize the original LoRA (Hu et al., 2022) and add parameters in the FFN module in

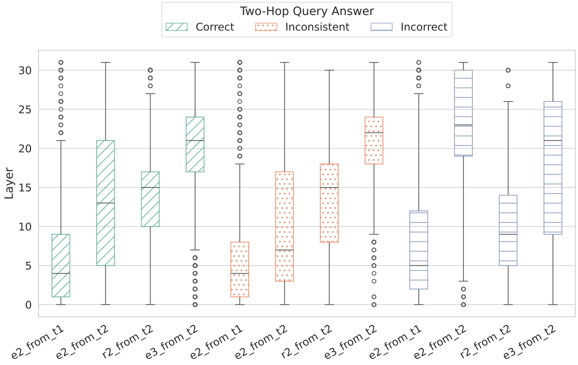


Figure 6: The distribution of the layers allows us to detect the information from critical positions in the model via patch_scope.

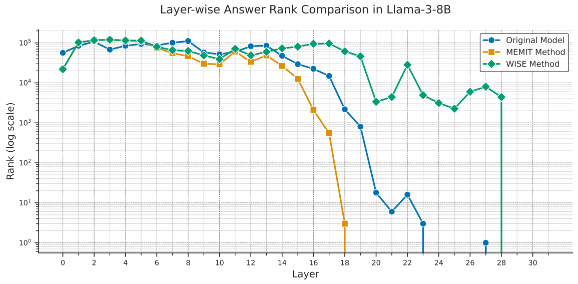


Figure 7: The target answer token's rank in the vocabulary of different editing methods when editing the fact '*The official language of Japan is Japanese* $\rightarrow$ *Korean.*'

the model. The hyperparameters are as follows:

- epoch: [40, 50, 60]
- batch size: [4]
- learning rate: [1e-4]
- rank: [8]
- lora_alpha: [32]

B.3 Unified Analysis

We first compare the different behaviors between the MEMIT-edited, WISE-edited, and the original model in Figure 7. Here, we edit the fact: '*The official language of Japan is Japanese* $\rightarrow$ *Korean.*' and map each layer's output to the embedding space and draw the rank of the target-token in the vocabulary as Yao et al. (2024) did. From the figure, we can see that in the original model and the MEMIT-edited model, the answer token is dealt with gradually through the mid-to-later layers, and MEMIT would make this happen in advance. On the contrary, the WISE method would directly alter the information at the edited layer, as we can see the sharp drop at layer 29. The distinct behaviors arise because the editing only takes effect when the gated function $\mathcal{G}(\mathbf{x})$ is activated by the specific input representation. ROME-style methods inject a modified representation into the existing computational flow relatively early or mid-stream, relying on subsequent layers to interpret this new representation. WISE-style methods, particularly when applied to later layers, act more like a direct 'fix' or 'override' at the point of editing, with the change being more immediately apparent.

C More Analysis

C.1 Concurrence or Reasoning?

Studies such as Yang et al. (2024b); Ju et al. (2024); Hou et al. (2023); Zhang et al. (2024c) those have discovered shortcuts in multi-hop reasoning. In the case of $((e_1,'', e_2), (e_2, r_2, e_3))$ (i.e., the query without r1), the model predicts correctly due to a high correlation between e_1 and e_3. For instance, given the query: "The capital city of the country where the Eiffel Tower is located is..." LLMs can sometimes provide the correct answer even without the intermediate context ('the country where the Eiffel Tower is located'). In our analysis, we find that apart from the occurrence, the LLM would also sometimes conduct latent reasoning, such as 'latently conducting the r1 completion'. If the model gives the correct e3 for $((e_1,'', e_2), (e_2, r_2, ?))$ due to the occurrence, once we edit the $(e_1, r_1, e_2 \rightarrow e'_2)$, the model would fail to give us the new answer. We select the shortcut data and conduct the editing in the first hop $(e_1, r_1, e_2 \rightarrow e'_2)$ and then evaluate the model to see whether the edited model would output updated knowledge (e_1, r_1, r_2, e'_3). We conduct experiments on LLAMA3-8B-Instruct with the AlphaEdit method and demonstrate that about 65% percent of cases would give us the updated knowledge for the multi-hop questions, showing that edits to intermediate hops (e.g., updating the country) can disrupt reasoning when relying on pre-existing-shortcuts and correctly give us the newly updated reasoning results. This means that the LLM itself does not simply answer the questions due to the high correlation between e1 and e3, **but actually conducts the latent reasoning**.

C.2 Circuit Analysis

We present the model's critical information detection results in Figure 6. The results indicate that knowledge is distributed across different layers, with incorrect cases appearing in later layers compared to correct and inconsistent cases.

Method	Model	MQUAKE-CF		MQUAKE-CF-v2		MQUAKE-T	
		H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑	Hop-wise.↑	MAcc.↑
Pre-edited		73.4	40.7	72.8	39.5	56.1	15.6
AdaLoRA	Qwen2.5-7B-Ins	35.1	24.9	36.5	25.9	25.0	28.6
WISE		41.2	9.8	26.5	8.0	50.2	36.5
MeLLo		35.5	7.8	34.5	7.6	52.7	56.5
ROME		75.4	10.7	73.4	8.8	86.7	17.7
MEMIT		82.6	11.1	83.4	9.6	88.9	18.5
AlphaEdit		73.8	12.6	75.1	10.5	82.2	17.2
IFMET ♣		83.7	25.7	84.6	24.5	90.0	52.8
CaKE(ours)		**90.6**	**61.4**	**90.3**	**63.05**	**95.5**	**87.8**

Table 7: Comparison of CaKE with existing methods on MQuAKE on Qwen2.5-7B-Instruct. The best results are highlighted in bold, while the second-best results are underlined. ♣ means the results are based on our own implementation since the original code is not open by the authors, and we will update it after the source code is open.

Method	Model	MQUAKE-CF		MQUAKE-CF-v2		MQUAKE-T	
		H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑	H-Acc.↑	MAcc.↑
Pre-edited		75.6	34.7	76.8	37.7	60.1	15.6
LoRA	L-70B	93.1	53.2	90.5	50.2	90.1	90.6
MeLLo		8.0	6.4	8.6	9.9	11.6	32.9
CaKE(ours)		**93.5**	**65.4**	**93.3**	**63.3**	**91.1**	**94.6**

Table 8: Comparison of CaKE with existing methods on MQuAKE for LLAMA3-70B-Instruct. The best results are highlighted in bold, while the second-best results are underlined. Due to the computational limitations, we just ran the LoRA and MeLLo in the 70B model. Here, LoRA for the 70B model is added for all layers because the performance on a single layer is extremely low.

Evaluation To validate this circuit hypothesis, we conduct a causal analysis to determine whether modifying the variables in the function F leads to corresponding changes in the output. Our intervention strategy focuses on the critical last token position for the second hop in Figure 8, where intermediate variables are stored. Specifically, we consider: 1).Entity Patching: Replacing the representation of e_2 with an alternative entity e_{patch}. For example, given the prompt *'The official currency of the country where the Eiffel Tower is located is'*, we substitute the representation of the bridge entity *'France'* with *'China'*, expecting the output to change to *'Renminbi'*. 2).Relation Patching: Replacing the representation of r_2 with an alternative relation r_{patch}. For instance, given the same prompt, we substitute the representation of *'official currency'* with *'capital'*, expecting the output to change to *'Paris'*. A successful patching (producing $F_2(e_{patch}, r_2)$ or $F_2(e_2, r_{patch})$) would confirm the model's reliance on these specific representations for reasoning. We conduct experiments on LLAMA3-8B-Instruct and Qwen2.5-7B-Instruct and employ PatchScope (Ghandeharioun et al., 2024) for targeted activation patching (detailed in §B.1). Table 9 shows that in both the LLAMA3-8B and Qwen2.5-7B models, substituting variable representations at the last token position leads to corresponding behavioral changes, particularly in entity patching, where accuracy exceeds 80%. These results provide mechanistic evidence for the reasoning circuit we identified before.

Model	Entity Patch	Relation Patch
LLaMA3-8B-Ins.	85.35	56.20
Qwen2.5-7B-Ins.	97.29	55.40

Table 9: Activation Patching Success Rates (%).

Determining the optimal layer for editing remains challenging, so we choose to adjust the model across all layers. In the future, we aim to refine our approach by performing more targeted edits.

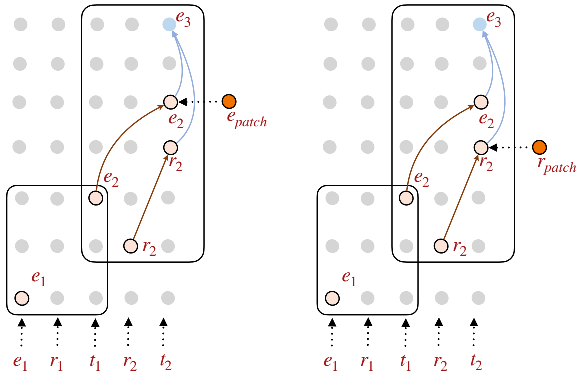


Figure 8: The way we test the function of the second hop. If the model conducts the function at the later layers, changing the representation would change the output of the model.

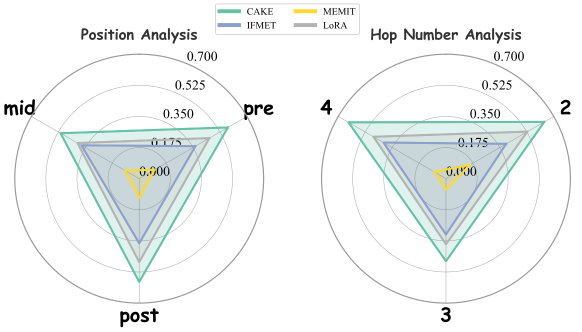


Figure 9: Accuracies of different number hops and edit-positions in MQuAKE-CF-3k-v2 on LLAMA3-8B-Instruct.

C.3 Failure Phenomenon

In the multi-hop reasoning, we view several failure cases to see how the language model made mistakes for reasoning and we see it as the **circuit competition**. Here, we find the LLM tends to give us a wrong answer for the middle cases of the different entities that appeared in the middle steps. Take ‘The country that the creator of Hamlet was a citizen of’ as an example; the bridge entity here is ‘William Shakespeare’. We view ‘Hamlet’ as an entity that would influence the model to give us the results ‘Denmark’, which means the model has been distracted by other entities’ information. As shown in Figure 5, the model gives us the correct answer ‘England’ around layer 27 but output the wrong answer ‘Denmark’, which is actually the country of the ‘Hamlet’.

C.4 Discussion with Chain-of-Thought

Instead of directly providing an answer, chain-of-thought (CoT) reasoning generates intermediate steps sequentially. As proposed by Yang et al. (2024b), CoT not only facilitates knowledge activation in large language models but also transforms them into effective in-context reasoners. The CoT process builds a chain of relevant facts within the prompt context, where each step’s output serves as an *in-context memory* that subsequent steps can reference. This approach reduces the risk of losing track of intermediate facts as the sequence length increases, thereby promoting more coherent multi-hop reasoning. Moreover, because a significant portion of the model’s knowledge is stored in earlier layers, CoT can better leverage these neurons by decomposing complex questions into simpler sub-questions (Wang et al., 2024f; Yao et al., 2025a). Consequently, the reasoning circuit required for a single-hop inference is much simpler than that for multi-hop reasoning. This observation aligns with recent findings (Li et al., 2024a), which demonstrate that fast thinking without CoT leads to larger gradients and greater gradient disparities across layers compared to CoT. Nonetheless, inconsistencies in the intermediate reasoning steps still occur, highlighting potential areas for improvement. We believe that further analysis is needed to address these issues, and we leave this exploration for future work.

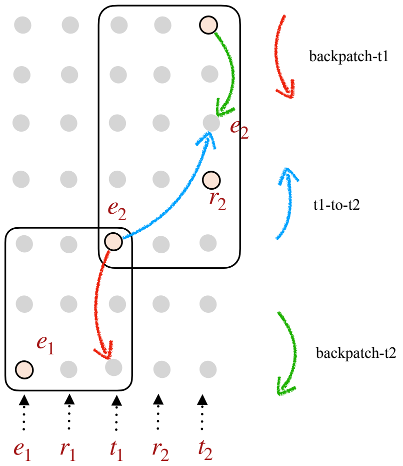


Figure 10: The way we conduct the backpatch and e_1 to e_2. We substitute the hidden representations from the source position to the target position.

C.5 Efficiency Analysis

We also compare the efficiency of CaKE with other baselines in Table 10. We compare the wall-clock time and memory usage here on LLAMA3-8B model and sample 100 numbers of data to run the analysis from MQuAKE-CF-3k. Here, the time

Method	Wall-clock Time	Memory (BF16)
ROME	2.71s	20.68GB
MEMIT	30.11s	24.42GB
WISE	76.01s	21.37GB
IFMET	44.72s	25.19GB
AlphaEdit	17.60s	38.80GB
CaKE	43.54s	18.52GB

Table 10: Time and Memory requirements Comparison

is the average time for one edit, and memory is the peak VRAM usage monitored by pynvml using one A6000 GPU. Here, AlphaEdit does not support BF16 or FP16, so the computation is FP32.

We can see that ROME and MEMIT require more memory than other methods. This is due to the matrix processes, like the calculation of an inverse matrix, which are memory-intensive. When the matrix is larger, the requirements are even more, which limits the method's scalability. Also, the improvement methods based on MEMIT like IFMET would also increase the time requirement but still require large memory. The memory would require even more when the model becomes larger (Yao et al., 2025b). Contrastly, CaKE can handle a 70B-sized edit via two A100 GPUs and achieve better performance, which demonstrates the efficiency of our proposed method.

C.6 Multiple Edit Test

We also conduct experiments on multiple edit scenarios. We can find that CaKE still shows competitive performance in multiple-edit scenarios compared to other methods.

Table 11: Performance Comparison with Different Edit Numbers

Method	Edit_num=10	Edit_num=100
MEMIT	16.0	12.5
IFMET	27.5	19.5
AlphaEdit	12.7	7.5
CaKE	**59.0**	**34.5**

C.7 Data Example

We show an example of the generated data for the fact '*The author of Misery is Richard Dawkins.*' in the following box.

Q1. If someone is looking for the person responsible for penning *Misery*, whose name should they search for?

Q2. If Alice resides in a mansion, Bob resides in a cottage, and Richard Dawkins resides in a villa. Therefore, the author of *Misery* resides in?

Q3. Given that Sarah prefers tea, James prefers coffee, and Richard Dawkins prefers herbal tea, what does the author of *Misery* prefer to drink?

Q4. During a book club meeting, the first discussion was on *Misery*, the second on *The Blind Watchmaker*, and the third on *River Out of Eden*. Who wrote the book that was the subject of the first discussion?

Q5. A library display features three novels: *Misery* on the top shelf, *The Extended Phenotype* on the middle shelf, and *Climbing Mount Improbable* on the bottom shelf. Who is the author of the novel placed on the top shelf?