CAKE: Causal-Guided Adaptive Knowledge Editing for LLMs

Shuxin Liu[†] **Jianhao Zhang**[†]

Hangzhou Institute for Advanced Study,
University of Chinese Academy of Sciences, Hangzhou, China
{liushuxin25, zhangjianhao24}@mails.ucas.ac.cn

Abstract

LLMs have static pre-trained knowledge, leading to obsolescence and hallucinations. Knowledge Editing (KE) addresses these issues and typically requires multi-layer modifications. However, existing state-of-the-art methods, largely following the Locate–Select–Assign–Edit (LSAE) paradigm, rely on fixed-layer selection and uniform residual assignment, ignoring the heterogeneous causal efficacy of different layers. To bridge this, we propose CAKE (**C**ausal-Guided **A**daptive **K**nowledge **E**diting), a collaborative editing method within the more general Locate–Weight–Assign–Edit (LWAE) paradigm that: (1) selectively identifies critical layers via causal tracing scores; and (2) adaptively allocates editing burdens based on causal weights rather than uniform assumptions. We formulate residual assignment as a constrained quadratic optimization problem and derive a solution for optimal residual allocation, showing that aligning edits with causal efficacy mitigates recursive error accumulation. Furthermore, we establish a generalized weight shift error bound, under which existing paradigms emerge as special, restricted cases. Experimental results demonstrate that CAKE achieves SOTA performance with comparable overhead, validating the superiority of causal-guided adaptation. Code at: https://github.com/zjh-vinky/CAKE.

1 Introduction

LLMs have demonstrated unprecedented capabilities in natural language tasks, powering numerous downstream applications (Brown et al., 2020; Achiam et al., 2023). However, their knowledge is inherently static, being fixed at pre-training time. As facts evolve, such as changes in political leadership or updates in scientific discoveries, LLMs inevitably exhibit knowledge obsolescence and generate hallucinations when queried about updated facts (Ji et al., 2023). Retraining or fine-tuning the model to update specific facts is computationally prohibitive and prone to catastrophic forgetting. Hence, KE has emerged as a vital research direction, aiming to precisely alter a model's behavior for specific inputs while preserving its general performance and unrelated knowledge. (Yao et al., 2023; De Cao et al., 2021a; Pan et al., 2025).

Among existing KE approaches, Locate-then-Edit methods represent the SOTA. MEMIT (Meng et al., 2023) extends single-layer editing ROME (Meng et al., 2022) to multi-layer settings by distributing the target residual across layers, but relies on a heuristic uniform allocation strategy that implicitly assumes linear superposition of residual effects. Recently, AlphaEdit (Fang et al., 2025) advanced the field by introducing null-space projection and operates within the same sequential, greedy optimization framework as MEMIT. BLUE (Li et al., 2025a) restricts the editing scope through fixed-boundary layer selection, which lacks the flexibility to adapt to instance-specific causal patterns. Consequently, existing LSAE paradigm suffers from a common limitation: they rely on either uniform weight distribution or rigid layer selection, ignoring the heterogeneous causal efficacy of different layers and thus failing to establish a more effective collaborative editing strategy.

To bridge this gap, we propose **CAKE** (**C**ausal-Guided **A**daptive **K**nowledge **E**diting), a new framework rooted in the generalized collaborative LWAE paradigm. Unlike prior methods that perform independent layer-wise updates, CAKE adopts a collaborative formulation that coordinates parameter updates across layers according to their causal efficacy. For key layer selection, we follow the prevailing practice adopted by SOTA KE methods and their derivatives, which typically rely on causal tracing scores to localize candidate edit layers. While recent mechanistic interpretability studies suggest that tracing effects may not fully predict

[†]Equal contributions.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 20040–20070

edit success (Hase et al., 2023; Yoon et al., 2024), we retain this metric to ensure rigorous consistency with established baselines. Crucially, our framework introduces a paradigm shift by: (1) selectively identifying critical layers via causal tracing, and (2) adaptively allocating editing burdens based on causal efficacy of each layer. More importantly, our framework is designed to be fundamentally metric-flexible: it serves as a generalized allocator that can seamlessly integrate superior layer importance scores as they emerge in future research, ensuring CAKE's longevity beyond current localization techniques. Theoretically, CAKE generalizes existing paradigms, seamlessly transitioning between sparse (ROME-like) and dense (MEMIT-like) patterns depending on instance-specific causal distribution. This leads to an error-complementary solution, where intentional adjustments in one layer are optimally compensated by others, enabling a synergistic update across depths and allowing distinct layers to function as a cohesive unit.

Our contributions are summarized as follows:

- We propose a new KE paradigm LWAE, and instantiate it with CAKE. By integrating causal tracing information, a critical dimension overlooked by prior methods, CAKE selectively identifies and adaptively allocates editing burdens to critical layers. This global coordination directly mitigates the recursive error accumulation inherent in previous greedy or heuristic approaches.
- We advance the theoretical understanding of multi-layer knowledge editing by formulating residual allocation as a constrained optimization problem. We derive an optimal residual solution under general causal weights and establish a generalized error bound, which strictly subsumes prior uniform or boundary-based analyses.
- We conduct extensive evaluations on KE benchmarks across GPT-2 XL (1.5B), GPT-J (6B), and LLaMA3 (8B). Experimental results demonstrate that CAKE achieves SOTA performance. Crucially, CAKE effectively reduces the error drift observed in sequential editing baselines.

2 Preliminaries

2.1 Multi-layer Knowledge Editing

The goal of KE is to correct the model using a dataset of editing samples $\mathcal{D} = (s, r, o \rightarrow o^*)$. For each sample, the model is updated to map the subject-relation pair (s, r) to the new target object o^*, overwriting the original o. Multi-layer KE is an efficient paradigm that updates the FFN parameters of targeted layers $\mathcal{L} = \{l_1, l_2, \ldots, l_L\}$ within f_θ.

We consider a Transformer-based LLM f_θ with parameters θ. In the context of locate-then-edit methods, the $\boldsymbol{W}_l$ (e.g., the output projection weights) within FFN at each layer l is typically viewed as a key-value associative memory. Let $\boldsymbol{W}_l$ denote the weight matrix to be updated at layer $l \in \mathcal{L}$. Multi-layer KE computes the optimal parameter perturbation $\boldsymbol{\Delta}_l$ for each target layer. Consequently, under the updated parameters θ^*, defined by replacing $\boldsymbol{W}_l$ with $\boldsymbol{W}_l + \boldsymbol{\Delta}_l$, the model must satisfy the following condition:

$$P(o^* \mid s, r; \theta^*) > P(o \mid s, r; \theta^*). \tag{1}$$

Meanwhile, to prevent catastrophic forgetting, the update should minimize the disruption to unrelated knowledge. The pursuit of $\boldsymbol{\Delta}_l$ for a specific layer is typically formulated as a least-squares optimization problem.

2.2 Existing Multi-layer Editing Paradigm

Existing multi-layer editing methods (e.g., MEMIT, AlphaEdit) can be unified under a structured **Locate-Select-Assign-Edit (LSAE)** paradigm:

- **Locate:** Identify the critical range of layers within the model responsible for encoding the specific knowledge.
- **Select:** Determine the specific subset of layers $\mathcal{L}$ within the located range to be updated.
- **Assign:** Decompose the total editing target (residual) into local objectives for each selected layer.
- **Edit:** Solve the optimization problem to compute $\boldsymbol{\Delta}_l$ that satisfies the local objective while preserving old memories.

However, current implementations exhibit significant limitations in the **Select** and **Assign** phases. They typically rely on rigid heuristics (e.g., uniform distribution) to select layers and allocate residuals, ignoring complex non-linear inter-layer dependencies. This oversimplification leads to error accumulation, where inaccuracies in earlier layers propagate through the network, ultimately compromising editing efficacy. To address these flaws, we propose CAKE which adopts a more effective paradigm detailed in the successive part as illustrated in Fig. 1.

3 Methodology

This section first introduces a new Multi-layer collaborative KE paradigm. The proposed method CAKE is then described. Connections between CAKE and previous method are finally discussed.

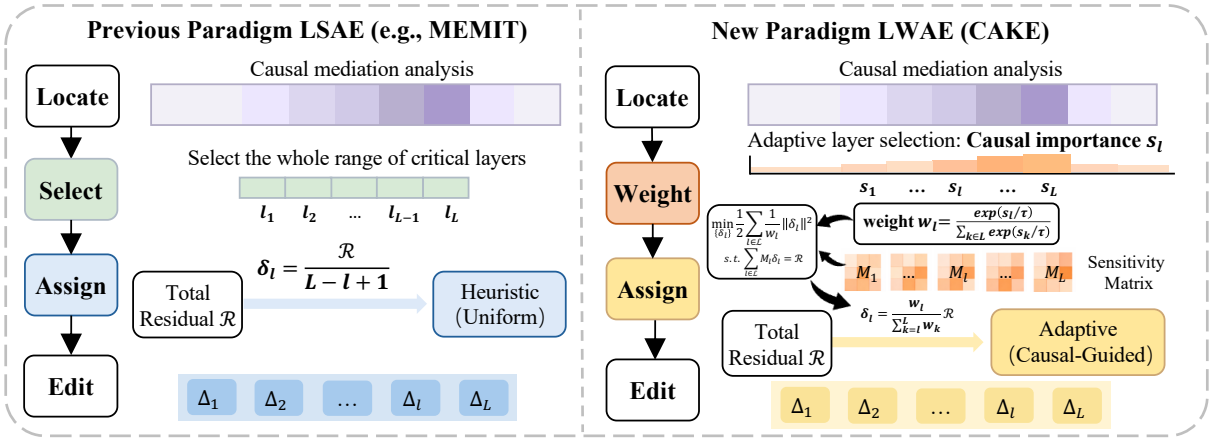

Figure 1: Two Locate-and-Edit paradims: Previous paradigm (left) and our New paradigm (right).

3.1 The LWAE Paradigm

To overcome the limitations of rigid layer selection and uniform residual distribution in LSAE, we propose a refined paradigm: Locate-Weight-Assign-Edit (LWAE). This paradigm acknowledges that layers within the critical region exhibit heterogeneous contributions to knowledge storage, necessitating a dynamic allocation strategy. The four stages are defined as follows:

- **Locate:** Identical to the "Locate" step in LSAE.
- **Weight:** Evaluate the specific causal importance s_l of each layer $l \in \mathcal{L}$. Unlike the binary "Select" step in LSAE, this step quantifies *how much* each layer should contribute to the edit.
- **Assign:** Adaptively decompose the global editing target into local residuals based on the computed weights. Layers with higher importance (s_l) are assigned a larger proportion of the editing burden, minimizing the collective error.
- **Edit:** Identical to the "Edit" step in LSAE.

As illustrated in Fig. 1, LWAE leverages neglected causal information to upgrade heuristic "Select" to principled "Weight", and shifts from uniform to adaptive residual distribution. We instantiate this paradigm as a practical method described in the successive part.

3.2 The proposed method CAKE

3.2.1 Causal-Guided Weighting

The premise of effective multi-layer editing is recognizing that not all layers contribute equally to factual knowledge. To capture the heterogeneous efficacy of different layers, we leverage Causal Tracing (Meng et al., 2022) to compute a raw importance score s_l for each layer $l \in \mathcal{L}$.

To achieve a flexible transition between sparse and dense editing patterns, we normalize the causal scores into causal weights w_l with:

$$w_l = \frac{\exp\left(s_l/\tau\right)}{\sum_{k\in\mathcal{L}} \exp\left(s_k/\tau\right)}, \tag{2}$$

where τ is a hyper-parameter controlling sparsity. Notably, as $\tau \to 0$, this weighting mechanism subsumes the discrete "Select" function. As illustrated in Fig. 2, varying τ modulates the distribution of a given $\{s_l\}$ set: in (a) and (b), weights for non-critical layers vanish to near-zero, realizing an adaptive layer selection within a unified framework.

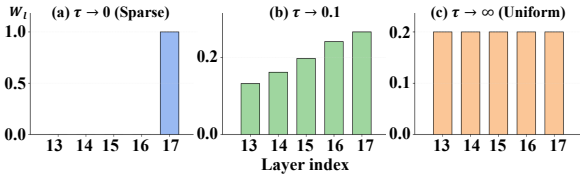

Figure 2: Weighting strategy for a set of Causal Tracing Scores {0.088, 0.090, 0.092, 0.094, 0.095}.

Entropy-regularized view. The adaptive weights $\{w_l\}$ is the optimal solution to the following entropy-regularized linear objective:

$$w^\star = \arg\max_{w\in\Delta^{|\mathcal{L}|}} \sum_{l\in\mathcal{L}} w_l s_l + \tau H(w), \tag{3}$$

where $\Delta^{|\mathcal{L}|}$ denotes the probability simplex and $H(w) = -\sum_l w_l \log w_l$ denotes the entropy of the weight distribution. In this view, s_l acts as a soft prior over layers, while τ controls the trade-off between concentrating updates on high-score layers and maintaining smooth multi-layer cooperation. When the attribution signal is less reliable, a larger τ strengthens the entropy term and leads to a smoother, more uniform allocation, thereby reducing over-reliance on a single signal. Conversely, when the attribution signal is informative,

a smaller τ places more emphasis on the utilization term $\sum_{l\in\mathcal{L}} w_l s_l$, encouraging the update budget to focus on the most causally salient layers. Hence, attribution in CAKE is used to guide adaptive allocation rather than to impose a rigid layer-selection rule. The core advantage of CAKE lies in cross-layer cooperative adaptive allocation, which can still reduce residual mismatch and editing errors even when the attribution signal is imperfect.

3.2.2 Adaptive Residual Assignment

With the causal weights $\{w_l\}$ established, our goal is to distribute the editing burden across layers. Formally, let $\boldsymbol{\mathcal{R}}$ denote the **total target residual**, evaluated at the last layer L in the key region $\mathcal{L}$, representing the discrepancy between the desired memory and the current model state. Furthermore, we adopt a *local first-order approximation* and model the propagation from the l-th layer to the final output by a **sensitivity matrix** $\boldsymbol{M}_l$. Under this approximation, matching the desired residual yields the consistency condition $\sum_{l\in\mathcal{L}} \boldsymbol{M}_l\boldsymbol{\delta}_l \approx \boldsymbol{\mathcal{R}}$, with a second-order remainder $\mathcal{O}\big(\sum_{l\in\mathcal{L}} \|\boldsymbol{\delta}_l\|_2^2\big)$. To solve for the optimal $\{\boldsymbol{\delta}_l\}$, we seek the minimum-perturbation solution under this first-order model. Dropping the second-order remainder, we enforce the first-order consistency exactly and use an inverse-weighted penalty $1/w_l$ so layers with larger w_l are penalized less and take larger edits, leading to the following constrained quadratic program:

$$\min_{\{\boldsymbol{\delta}_l\}} \frac{1}{2}\sum_{l\in\mathcal{L}} \frac{1}{w_l}\|\boldsymbol{\delta}_l\|_2^2 \quad \text{s.t.} \sum_{l\in\mathcal{L}} \boldsymbol{M}_l\boldsymbol{\delta}_l = \boldsymbol{\mathcal{R}}. \quad (4)$$

To solve this efficiently, we employ the method of Lagrange multipliers. Solving Eq. (4) yields the following closed-form solution:

Theorem 1 (Optimal Adaptive Allocation)

$$\boldsymbol{\delta}_l^* = w_l \boldsymbol{M}_l^T \Big(\sum_{j\in\mathcal{L}} w_j \boldsymbol{M}_j \boldsymbol{M}_j^T\Big)^{-1} \boldsymbol{\mathcal{R}}. \quad (5)$$

This result reveals a key insight: the optimal update magnitude for layer l is jointly determined by its causal weight w_l and its local sensitivity $\boldsymbol{M}_l$. The proof is provided in Appendix A.2.

3.2.3 The Editing Algorithm

Explicitly computing the sensitivity matrix $\boldsymbol{M}_l$ involves full-Jacobian computations, which is intractable for large-scale models. To bridge this gap, we simplify Eq. 5 by adopting the local linearity assumption used in MEMIT and AlphaEdit, approximating $\boldsymbol{M}_l \approx \boldsymbol{I}$. A theoretical discussion of this approximation and an empirical comparison with explicit sensitivity computation are provided in Appendix A.1. Under this assumption, the projection term in Theorem 1 simplifies, suggesting that the residual allocated to each layer should be proportional to its causal weight w_l. However, a single-shot global allocation based on the total residual $\boldsymbol{\mathcal{R}}$ is often fragile in practice, as early-layer updates may induce representation drift and alter subsequent propagation. Since practical editing proceeds sequentially and the effective residual evolves with model updates, we instantiate the global solution via a layer-wise allocation that adapts to the current residual at each step. To ensure consistent allocation under this update process, we perform a **dynamic weight re-normalization** at each step.

Specifically, when editing the l-th layer, we consider only the contributions of the current and subsequent unedited layers. The residual $\boldsymbol{\delta}_l$ allocated to the current layer is derived as follows:

$$\boldsymbol{\delta}_l = \frac{w_l}{\sum_{k=l}^{L} w_k} \cdot \boldsymbol{\mathcal{R}}', \quad (6)$$

where $\boldsymbol{\mathcal{R}}'$ denotes the current total residual, recomputed based on the model parameters updated through the previous layers.

Finally, with the target $\boldsymbol{\delta}_l$ determined, we solve for the parameter update $\boldsymbol{\Delta}_l$ using the standard least-squares objective on the local key-value pairs, identical to the update step in AlphaEdit. The complete procedure is summarized in Algorithm 1.

Algorithm 1 CAKE

Require: $\mathcal{L}, \{s_l\}, l \in \mathcal{L}, \mathcal{D}, \{\boldsymbol{W}_l\}, l \in \mathcal{L}, \tau$
Ensure: Updated $\{\boldsymbol{W}_l^*\}, l \in \mathcal{L}$
1: Compute initial total residual $\boldsymbol{\mathcal{R}}$
2: **for** $l = l_{start}$ **to** l_{end} **do**
3: $\quad \boldsymbol{\delta}_l \leftarrow (w_l / \sum_{k=l}^{l_{end}} w_k) \cdot \boldsymbol{\mathcal{R}}$
4: $\quad$ Compute $\boldsymbol{\Delta}_l$ given target $\boldsymbol{\delta}_l$
5: $\quad \boldsymbol{W}_l \leftarrow \boldsymbol{W}_l + \boldsymbol{\Delta}_l$
6: $\quad$ Update $\boldsymbol{\mathcal{R}}$ with current model parameters
7: **end for**
8: **return** $\{\boldsymbol{W}_l\}, l \in \mathcal{L}$

3.3 Connections with Existing Methods

Our framework generalizes prior arts as special cases governed by the causal scores $\{s_l\}$ and τ.

Dense (MEMIT-like) strategy (Uniform allocation). As $\tau \approx \infty$ or when causal scores are identical across layers, the softmax weights become uniform ($\frac{w_l}{\sum_{k=l}^{L} w_k} \approx 1/(L-l+1)$). Consequently, our dynamic allocation (Eq. 6) simplifies to dividing the residual equally among remaining layers,

strictly recovering the strategies of MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2025).

Sparse (ROME-like) strategy (Single-layer allocation). If $\tau \approx 0$, only the layer with the highest causal score has the weight 1 and the weights of other layers equal to 0. This reproduces the single-layer allocation of ROME (Meng et al., 2022).

Boundary Strategy (BLUE). Since the approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$ underestimates the high output sensitivity of later layers, applying a depth-increasing correction (e.g., $(l/L)^\alpha$) to the decreasing $\{s_l\}$ yields a **U-shaped** weight distribution. As $\tau \approx 0$, this concentrates allocation on the first and last layers, recovering BLUE (Li et al., 2025a).

In essence, these methods implicitly assume rigid causal distributions (either perfectly uniform or boundary-concentrated). Our approach, by contrast, adaptively fits the diverse causal structures of real-world knowledge, avoiding suboptimal editing when these extreme assumptions fail.

4 Theoretical Analysis

In this section, we provide a rigorous theoretical analysis of our proposed method. We generalize the error bound framework of Li et al. (2025a) from uniform to arbitrary layer-wise weight distributions, and conduct the analysis under a layer-wise editing procedure over critical layers $l \in \mathcal{L}$.

4.1 Generalized Weight Shift Error

In the LWAE paradigm, the update at layer l is governed by a normalized weight w_l. We define the *dynamic allocation ratio* η_l for the layer-wise process as the proportion of the current weight relative to the remaining weight budget $\eta_l = w_l / \sum_{k=l}^{L} w_k$.

Theorem 2 (Generalized Weighted Error Bound) *Under the local first-order linearization framework in Sec. 3.2.2 and assuming monotonic residual reduction, let $\boldsymbol{\Delta}_l^*$ and $\boldsymbol{\Delta}_l$ denote the exact and the actual weight shift, respectively. The upper bound for the weight shift error at layer l is given by:*

$$\|\boldsymbol{\Delta}_l^* - \boldsymbol{\Delta}_l\|_2 \leq (\underbrace{\|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L\|_2}_{\textit{Term A}} + \underbrace{(1-\eta_l)\|\boldsymbol{\mathcal{R}}_L\|_2}_{\textit{Term B}})\|\boldsymbol{Q}\|_2, \quad (7)$$

where $\boldsymbol{\mathcal{R}}_l^*, \boldsymbol{\mathcal{R}}_L$ denote the ideal, and current total residuals respectively. Here $\boldsymbol{Q}$ is the least-squares projection operator, given by $\boldsymbol{Q} = \boldsymbol{K}_1^{l^T}\left(\boldsymbol{K}_0^l \boldsymbol{K}_0^{l^T} + \boldsymbol{K}_1^l \boldsymbol{K}_1^{l^T}\right)^{-1}$. The proof is provided in Appendix A.3.

Remark. Batch size affects the magnitude of $\|\mathcal{R}_L\|_2$ and $\|\boldsymbol{Q}\|_2$ through the number of newly introduced keys in $\boldsymbol{K}_1^l$. For sequential editing, the layer-wise projection operator $\boldsymbol{Q}$ naturally extends to a history-dependent form by augmenting $\boldsymbol{K}_1^l$ with cached keys $\boldsymbol{K}_p^l$ from previous edits as the operator analyzed in Lemma 4.3 of Li et al. (2025a).

4.2 Optimality of Attribution-Based Weighting

While baselines such as MEMIT and BLUE suffer from either high mismatch risk or excessive propagation noise (see Appendix A.4), CAKE addresses both failure modes by allocating update weights according to layer-wise attribution scores ($w_l \propto s_l$). Assuming the *Projection Alignment* hypothesis, we analyze this design and obtain the following three corollaries. Proofs are provided in Appendix A.5.

Corollary 1 (Attribution–Mismatch Alignment) *Let $m_l := \|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L\|_2$ denotes the mismatch at layer l. We assume that higher attribution implies smaller mismatch: $s_i \geq s_j \Rightarrow m_i \leq m_j$. Hence, a causal-score–prioritized allocation ($w_l \propto s_l$) places more weight on smaller m_l, which decreases the aggregated $\sum_{l=1}^{L} w_l m_l$, thereby tightening the Term A contribution in Eq. 7.*

Corollary 2 (Residual Propagation Suppression) *For causally irrelevant layers (small s_l), CAKE assigns negligible weights ($w_l \propto s_l$), concentrating the update budget on a few layers, and the aggregated residual-propagation contribution induced by Term B is controlled by the* effective number of update layers $N_{\text{eff}} \approx 1/\|\boldsymbol{w}\|_2^2$, *thereby tightening the Term B contribution in Eq. 7.*

Corollary 3 (Effective Depth Regularization) *Concentrating update weights on high-attribution layers reduces the* effective number of update layers *($N_{\text{eff}} \approx 1/\|\boldsymbol{w}\|_2^2$), which bounds the cumulative contribution of projection operators $\sum \|\boldsymbol{Q}\|_2$ and mitigates error amplification during the layer-wise procedure.*

5 Experiments

In this section, We evaluate CAKE aiming to answer the following questions: (1) Can CAKE enhance the performance compared to methods relying on rigid allocation paradigms? (2) Does CAKE preserve the model's general capabilities during large-scale sequential editing streams? (3) Does the causal-guided allocation mechanism truly function as theoretically expected to minimize residual drift and representation shift?

Model	Method	Counterfact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3 (8B)	ROME	$66.20_{\pm 0.44}$	$63.42_{\pm 0.45}$	$48.66_{\pm 0.28}$	$458.72_{\pm 0.28}$	$2.09_{\pm 0.04}$	$1.32_{\pm 0.03}$	$1.36_{\pm 0.02}$	$0.67_{\pm 0.02}$
	MEMIT	$65.55_{\pm 0.38}$	$63.98_{\pm 0.32}$	$51.22_{\pm 0.44}$	$452.52_{\pm 1.27}$	$5.91_{\pm 0.17}$	$36.29_{\pm 0.29}$	$31.50_{\pm 0.30}$	$18.65_{\pm 0.22}$
	PRUNE	$66.85_{\pm 0.39}$	$64.60_{\pm 0.40}$	$49.84_{\pm 0.28}$	$440.63_{\pm 1.17}$	$6.16_{\pm 0.08}$	$0.48_{\pm 0.04}$	$0.61_{\pm 0.09}$	$1.29_{\pm 0.03}$
	RECT	$64.50_{\pm 0.32}$	$62.38_{\pm 0.40}$	$60.74_{\pm 0.33}$	$520.98_{\pm 0.39}$	$19.24_{\pm 0.12}$	$87.36_{\pm 0.28}$	$82.42_{\pm 0.20}$	$32.11_{\pm 0.27}$
	AlphaEdit	$99.20_{\pm 0.12}$	$86.25_{\pm 0.15}$	$77.12_{\pm 0.24}$	$624.91_{\pm 0.20}$	$30.22_{\pm 0.16}$	$94.38_{\pm 0.11}$	$88.19_{\pm 0.14}$	$32.72_{\pm 0.18}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.92_{\pm 0.07}$	$97.58_{\pm 0.42}$	$77.10_{\pm 0.83}$	$624.97_{\pm 0.27}$	$33.87_{\pm 0.47}$	$95.90_{\pm 0.51}$	$92.23_{\pm 0.50}$	$32.40_{\pm 0.71}$
	CAKE	**$99.95_{\pm 0.04}$**	**$97.92_{\pm 0.24}$**	**$78.79_{\pm 0.32}$**	**$626.59_{\pm 0.46}$**	**$34.97_{\pm 0.38}$**	**$96.80_{\pm 0.15}$**	**$93.40_{\pm 0.18}$**	**$33.46_{\pm 0.24}$**
GPT-J (6B)	ROME	$52.35_{\pm 0.27}$	$50.64_{\pm 0.35}$	$51.02_{\pm 0.63}$	$581.53_{\pm 0.20}$	$1.50_{\pm 0.06}$	$55.75_{\pm 0.27}$	$52.86_{\pm 0.30}$	$13.29_{\pm 0.22}$
	MEMIT	$98.15_{\pm 0.15}$	$95.75_{\pm 0.13}$	$63.77_{\pm 0.25}$	$594.12_{\pm 0.68}$	$39.37_{\pm 0.20}$	$96.78_{\pm 0.15}$	$93.46_{\pm 0.29}$	**$31.28_{\pm 0.33}$**
	PRUNE	$84.40_{\pm 0.27}$	$87.50_{\pm 0.23}$	$53.30_{\pm 0.32}$	$460.23_{\pm 0.62}$	$23.38_{\pm 0.18}$	$25.59_{\pm 0.14}$	$24.14_{\pm 0.19}$	$20.52_{\pm 0.11}$
	RECT	$98.80_{\pm 0.09}$	$86.22_{\pm 0.22}$	$72.58_{\pm 0.25}$	$616.64_{\pm 0.14}$	$41.47_{\pm 0.16}$	$96.22_{\pm 0.20}$	$91.66_{\pm 0.25}$	$28.33_{\pm 0.21}$
	AlphaEdit	$99.78_{\pm 0.12}$	$96.43_{\pm 0.19}$	$76.19_{\pm 0.23}$	$618.44_{\pm 0.20}$	$42.32_{\pm 0.18}$	$99.61_{\pm 0.12}$	$95.93_{\pm 0.23}$	$28.27_{\pm 0.25}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.75_{\pm 0.14}$	$97.72_{\pm 0.56}$	$76.16_{\pm 0.89}$	**$621.59_{\pm 0.72}$**	$41.58_{\pm 0.39}$	$99.40_{\pm 0.22}$	$93.58_{\pm 0.51}$	$28.36_{\pm 0.76}$
	CAKE	**$99.85_{\pm 0.13}$**	**$98.50_{\pm 0.26}$**	**$76.54_{\pm 0.35}$**	$620.92_{\pm 0.32}$	**$43.01_{\pm 0.29}$**	**$99.84_{\pm 0.11}$**	**$97.32_{\pm 0.18}$**	$29.79_{\pm 0.20}$
GPT2-XL (1.5B)	ROME	$51.70_{\pm 0.41}$	$49.05_{\pm 0.24}$	$50.83_{\pm 0.27}$	$527.93_{\pm 0.92}$	$1.03_{\pm 0.03}$	$41.32_{\pm 0.48}$	$37.51_{\pm 0.35}$	$12.29_{\pm 0.26}$
	MEMIT	$94.80_{\pm 0.18}$	$85.65_{\pm 0.28}$	$60.25_{\pm 0.29}$	$472.17_{\pm 0.41}$	$22.04_{\pm 0.18}$	$79.64_{\pm 0.22}$	$72.56_{\pm 0.29}$	$26.06_{\pm 0.16}$
	PRUNE	$75.70_{\pm 0.31}$	$73.72_{\pm 0.24}$	$52.23_{\pm 0.19}$	$528.84_{\pm 0.22}$	$13.88_{\pm 0.17}$	$28.35_{\pm 0.23}$	$25.40_{\pm 0.27}$	$14.45_{\pm 0.15}$
	RECT	$92.40_{\pm 0.10}$	$81.32_{\pm 0.30}$	$65.02_{\pm 0.33}$	$483.13_{\pm 0.89}$	$20.43_{\pm 0.12}$	$83.36_{\pm 0.41}$	$75.32_{\pm 0.32}$	$24.87_{\pm 0.27}$
	AlphaEdit	$99.45_{\pm 0.22}$	$94.32_{\pm 0.28}$	$66.09_{\pm 0.19}$	$592.13_{\pm 0.31}$	$38.82_{\pm 0.19}$	$93.60_{\pm 0.23}$	$83.70_{\pm 0.33}$	$25.91_{\pm 0.25}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.10_{\pm 0.21}$	$92.07_{\pm 0.48}$	$69.10_{\pm 0.72}$	$612.55_{\pm 0.65}$	$39.67_{\pm 0.32}$	$97.50_{\pm 0.43}$	$88.89_{\pm 0.79}$	$26.44_{\pm 0.80}$
	CAKE	**$99.63_{\pm 0.21}$**	**$95.92_{\pm 0.34}$**	**$72.68_{\pm 0.23}$**	**$614.53_{\pm 0.29}$**	**$41.02_{\pm 0.31}$**	**$97.93_{\pm 0.19}$**	**$91.09_{\pm 0.36}$**	**$26.92_{\pm 0.27}$**

Table 1: Comparison of CAKE with original locate-then-Knowledge Editing methods and AlphaEdit variants on the knowledge editing task. The best results are highlighted in **bold**, while the second-best results are underlined.

5.1 Experiment Setup

Datasets & Evaluation Metrics. We utilize two standard datasets: ZsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022). We report standard metrics: Efficacy (success rate), Generalization (paraphrase robustness), Specificity (neighborhood success), Fluency (generation entropy) and Consistency (reference score) to comprehensively assess the performance of CAKE.

Baselines & Implementation Details. We compare CAKE against the following representative knowledge editing methods: ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), PRUNE (Ma et al., 2025), RECT (Gu et al., 2024), AlphaEdit (Fang et al., 2025), and BLUE (Li et al., 2025a). All methods are evaluated on three model architectures of varying scales, including GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), and LLaMA3 (8B) (Meta, 2024). For causal tracing, we follow the ROME implementation and compute the Average Indirect Effect (AIE) to estimate layer importance, see Appendix E for the definition and computation procedure. We set $\tau = 0.1$ and utilize the official implementations provided in the AlphaEdit repository to maintain consistency for all baselines. All experiments are conducted on a single NVIDIA A100 GPU.

5.2 Main Results

To assess the effectiveness and stability of CAKE, we follow the experimental protocol established by Fang et al. (2025), randomly sampling 2,000 editing instances with a batch size of 100. As shown in Table 1, CAKE outperforms across CounterFact and ZsRE for all three evaluated models. On CounterFact with GPT2-XL, CAKE improves specificity by 9.97% relative to AlphaEdit, reflecting a substantial reduction in unintended parameter interference under deep editing, while simultaneously improving fluency and consistency. On CounterFact with LLaMA3, CAKE achieves a consistent specificity gain of 2.17% relative to AlphaEdit, together with further improvements in efficacy and fluency. Compared to $\text{AlphaEdit}_{\text{BLUE}}$, CAKE attains notable gains on ZsRE with GPT-J, improving generalization by 4.00% while preserving high efficacy. Overall, the results indicate that CAKE improves edit quality consistently across model scales, achieving stronger generalization while better preserving specificity and generation quality, without sacrificing efficacy.

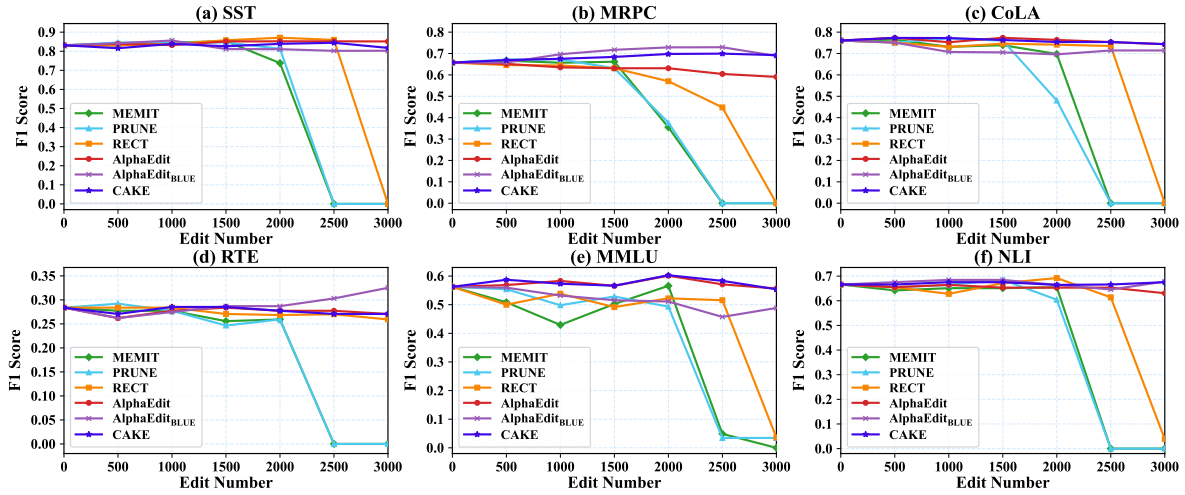

Figure 3: F1 scores of post-edited LLaMA3 (8B) across six tasks for general capability evaluation.

5.3 Sustainability of General Capabilities under Sequential Editing

To evaluate general language understanding after editing, we assess the general capabilities of post-edited LLMs on six tasks from the GLUE benchmark (Wang et al., 2018), including SST (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), CoLA (Warstadt et al., 2019), RTE (Bentivogli et al., 2009), NLI (Williams et al., 2018), and MMLU (Hendrycks et al., 2021).

We perform 3,000 sequential edits on LLaMA3 with a batch size of 100, evaluating performance every 500 edits. As shown in Fig. 3, MEMIT, PRUNE, and RECT exhibit severe degradation of general capabilities as edits accumulate. AlphaEdit demonstrates strong general capability retention throughout sequential editing, a property that is effectively inherited by AlphaEdit$_{\text{BLUE}}$. Building on this foundation, CAKE consistently matches or even slightly exceeds AlphaEdit and AlphaEdit$_{\text{BLUE}}$, indicating improved long-horizon robustness under continuous knowledge updates across the majority of evaluation tasks.

5.4 Ablation Study on Core Mechanisms

As discussed in Sec. 3.3, τ controls two mechanisms in CAKE: multi-layer collaboration and adaptive allocation. We ablate them on LLaMA3 with ZsRE dataset under three regimes: a ROME-like variant without multi-layer collaboration, a MEMIT-like variant without adaptive allocation, and CAKE retains both mechanisms. All variants are instantiated within AlphaEdit; thus the ROME-like regime additionally uses null-space projection.

As illustrated in Table 2, we report the effective number of updated layers $N_{\text{eff}} \approx 1/\|\boldsymbol{w}\|_2^2$ to characterize how residuals are distributed. Removing multi-layer collaboration collapses updates to one layer ($N_{\text{eff}} \approx 1$), yielding a sparse (ROME-like) regime that preserves locality but suffers from limited editing effectiveness and generalization. Removing adaptive allocation leads to near-uniform updates across layers ($N_{\text{eff}} \approx 5$), corresponding to a dense (MEMIT-like) regime that achieves higher effectiveness but introduces substantial interference, degrading specificity and generalization. In contrast, CAKE strikes the best balance: residuals are distributed across layers while remaining concentrated on causally salient ones ($N_{\text{eff}} \approx 4.85$), resulting in consistently superior performance across effectiveness, generalization, and specificity.

Regime	N_{eff}	Eff.↑	Gen.↑	Spe.↑
Sparse (ROME-like)	1.00	90.35	84.55	29.13
Adaptive (CAKE)	4.85	**96.80**	**93.40**	**33.46**
Dense (MEMIT-like)	5.00	94.24	89.36	32.68

Table 2: Ablation of core mechanisms in CAKE.

5.5 Residual Drift Analysis

To assess the mechanistic effect of causal-guided allocation and validate Corollary 3, we analyze recursive error accumulation across sequential layer updates. Specifically, we quantify the cumulative discrepancy between the residual assigned to each layer and the residual reduction it actually induces, which we refer to as *Cumulative Drift*. As illustrated in Fig. 5, MEMIT exhibits the fastest growth in cumulative drift, reflecting severe mismatch accumulation under uniform residual propagation. AlphaEdit mitigates this effect but still shows steadily increasing drift. In contrast, CAKE

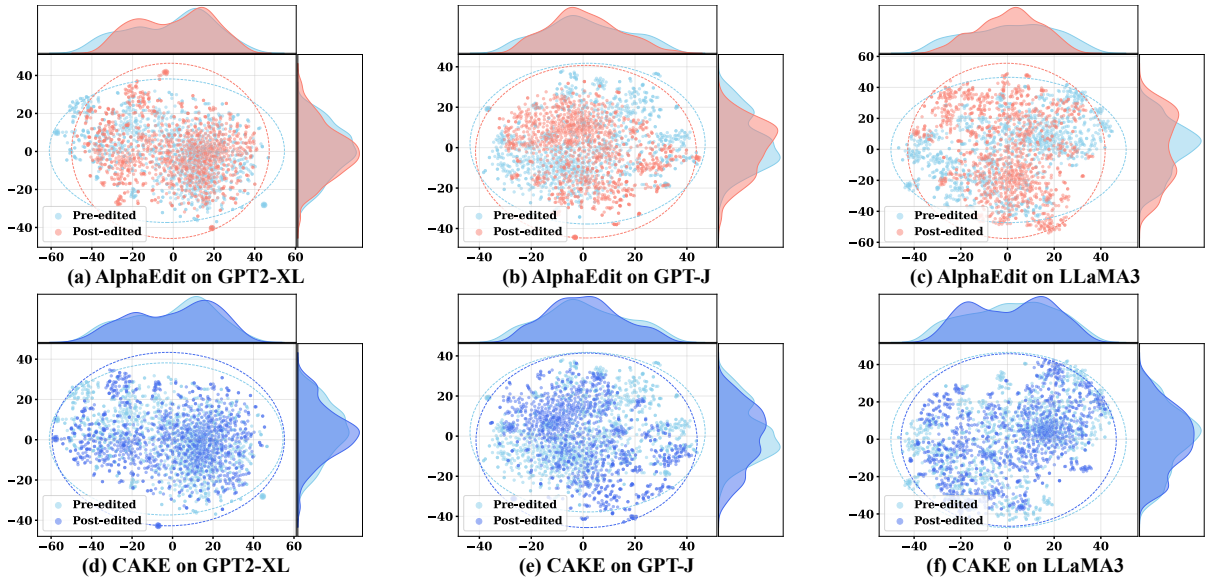

Figure 4: The distribution of hidden representations of pre-edited and post-edited LLMs.

yields substantially flatter curves, indicating that causal-guided allocation better aligns layer-wise effects with their burdens and suppresses recursive error accumulation. (See Appendix B.5 for details.)

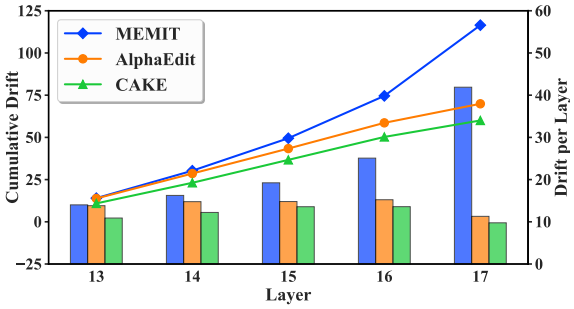

Figure 5: Recursive error accumulation under sequential editing. GPT-2-XL with 100 sequential edits on ZsRE. Line charts represent the Cumulative Drift (left axis), while bar charts illustrate the Drift per Layer (right axis).

5.6 Hidden Representations Analysis

Most existing KE methods tend to cause post-edited LLMs to overfit the injected facts, thereby altering the distribution of their internal representations. To examine whether CAKE mitigates this effect, we visualize hidden-state distributions before and after editing. For each model, we sample 1,000 factual prompts and extract the corresponding hidden representations, which are then projected into two dimensions using t-SNE (Maaten and Hinton, 2008) for comparison. As shown in Fig. 4, AlphaEdit leads to a partial shift in the distribution of post-edit representations away from the original distribution. In contrast, CAKE exhibits substantially smaller distributional changes across all three models, with post-edit representations remaining well aligned with the original distribution. This comparison suggests that CAKE's causally guided residual allocation mitigates excessive adaptation to injected knowledge and helps preserve representational consistency during editing.

Method	CounterFact			ZsRE		
	LLaMA3	GPT-J	GPT2-XL	LLaMA3	GPT-J	GPT2-XL
MEMIT	690.67s	297.23s	159.41s	366.63s	291.04s	165.19s
AlphaEdit	460.63s	364.26s	180.24s	323.80s	366.88s	183.79s
AlphaEdit$_{\text{BLUE}}$	910.62s	362.35s	185.70s	1197.48s	418.59s	189.52s
CAKE	431.79s	332.72s	161.15s	349.27s	367.12s	176.55s

Table 3: Time per batch (100 edits) for different editing methods. All times are reported in seconds.

5.7 Runtime Evaluation

To evaluate the computational efficiency of our framework, we report the average wall-clock time required to process a batch of 100 edits. The evaluation spans three model scales: GPT2-XL, GPT-J, and LLaMA3, across both the CounterFact and ZsRE datasets. For consistency, all experiments are conducted on a single NVIDIA A100 GPU under identical configurations, with the reported metrics averaged over the entire editing stream. As shown in Table 3, CAKE demonstrates a highly competitive computational footprint. Notably, CAKE frequently outperforms the baselines in terms of execution speed. For instance, when editing LLaMA3 on the CounterFact dataset, CAKE requires only 431.79s per batch, compared to 460.63s for AlphaEdit and 690.67s for MEMIT. This efficiency advantage is particularly evident when compared to AlphaEdit$_{\text{BLUE}}$, which exhibits significantly higher

latency. The results indicate that the causal-guided residual allocation in CAKE is computationally lightweight, as it primarily involves a one-time causal tracing pass that can be effectively amortized during batch processing. Consequently, CAKE preserves the strong scalability of the Locate-and-Edit paradigm while enabling superior multi-layer coordination without penalizing throughput.

6 Related Work

Multi-layer Knowledge Editing. Prior work has shown that FFN modules can be interpreted as key–value memories (Geva et al., 2021), and that Knowledge Neurons provide a mechanistic basis for factual knowledge storage (Dai et al., 2022). Building on these insights, ROME (Meng et al., 2022) localizes and edits factual associations via causal tracing, while MEMIT (Meng et al., 2023) extends this approach to multi-layer batch editing through least-squares optimization. Subsequent methods further improve robustness under repeated or large-scale edits, AlphaEdit (Fang et al., 2025) constrains updates via null-space projection, PRUNE (Ma et al., 2025) reduces interference through perturbation restraint, and RECT (Gu et al., 2024) preserves general capabilities via regularization. In parallel, PMET (Li et al., 2024) enhances precision through component-wise state decoupling with square-root residual spreading, while AnyEdit (Jiang et al., 2025) improves robustness and generalization across diverse and long-form edits, and SIR (Wang et al., 2025) mitigates ripple effects by selectively revising the most impacted facts. Despite these advances, most methods treat residual allocation across layers heuristically, typically adopting uniform distributions, which can induce accumulated errors and performance degradation. CAKE addresses this limitation by introducing a causal residual allocation strategy, enabling coordinated multi-layer updates grounded in layer-wise causal relevance.

Localization and Layer Selectivity. Causal effects have been used to probe influential mechanisms in neural networks and to avoid being misled by spurious correlations (Dai et al., 2022; De Cao et al., 2021b). ROME (Meng et al., 2022) applies causal mediation analysis to identify which layers contribute causally to a factual prediction and performs edits at the localized sites, while MEMIT (Meng et al., 2023) uses causal tracing to extend editing from a single layer to a multi-layer setting by selecting multiple causally relevant layers. $\text{MEMIT}_{\text{CSK}}$ (Gupta et al., 2023) further investigates whether commonsense judgments are causally linked to localized, editable parameters. MedLaSA (Xu et al., 2024) leverages causal associations between knowledge and layers to select update pathways for knowledge editing. However, in most existing methods, localization signals primarily serve as heuristics for independent layer-wise modulation rather than for jointly coordinating updates across multiple layers. BLUE (Li et al., 2025a) improves stability via boundary-layer restrictions but still assumes a fixed layer set. In contrast, CAKE uses causal signals to guide layer-wise update allocation, enabling coordinated multi-layer editing without fixed layer assumptions.

7 Conclusion

We propse CAKE, a causal-guided adaptive framework that fundamentally rethinks multi-layer knowledge editing. By challenging the homogeneous layer assumptions of prior arts, CAKE explicitly integrates causal tracing signals into a unified LWAE paradigm. This approach transforms editing from a rigid heuristic into a collaborative, importance-aware optimization problem. It not only ensures high editing success rates but also offers a robust solution for preserving general abilities while updating knowledge in LLMs.

Limitations

CAKE relies on an estimate of layer-wise importance to guide residual allocation. In this work, we use causal tracing–based signals (AIE) to derive these importance scores; their quality may vary across models and prompts, and inaccurate estimates can limit the benefits of adaptive allocation. Moreover, for analytical tractability and computational efficiency, our formulation adopts a local first-order approximation and further simplifies the sensitivity matrices as $\boldsymbol{M}_l \approx \boldsymbol{I}$. These assumptions may not fully capture inter-layer couplings and nonlinear propagation effects, especially under large edits or long sequential edit streams. Finally, although CAKE is compatible with locate-then-edit pipelines, causal tracing introduces additional overhead (e.g., repeated noise sampling and restoration runs per prompt), which may increase cost and implementation complexity in large-scale or high-frequency update settings.

Ethical Considerations

Knowledge editing can improve the reliability of LLM-based systems by correcting outdated or erroneous facts and reducing harmful hallucinations without full retraining. However, this capability is inherently dual-use: the same mechanisms can be misused to inject misinformation, covertly bias model behavior, or alter safety-relevant knowledge in ways that may be difficult for users to detect. Although CAKE is designed to produce localized, targeted updates, benchmark locality metrics do not fully capture downstream risks (e.g., bias, toxicity, or domain-specific harms), especially under large or long-horizon sequential edits. We therefore recommend that real-world deployments pair CAKE with strict access control for edit privileges, model versioning and audit logs of edit requests, and post-edit safety evaluations tailored to the target application. Finally, edits should avoid embedding personal or sensitive information and should follow appropriate governance and provenance practices, particularly in high-stakes settings.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant No. 42550187 and 62476191. We would also like to thank the anonymous reviewers and the area chair for their constructive comments. We also acknowledge the computing resources provided by our research group.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Hadi Asghari and Sami Nenno. 2025. Mechanistic interpretability of socio-political frames in language models. *arXiv preprint arXiv:2510.03799*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

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

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

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021b. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the third international workshop on paraphrasing (IWP2005)*.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

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, Miami, Florida, USA. Association for Computational Linguistics.

Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 8214–8232.

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. *Advances in Neural Information Processing Systems*, 36:17643–17668.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Xiaopeng Li, Shangwen Wang, Shasha Li, Shezheng Song, Bin Ji, Ma Jun, and Jie Yu. 2025a. Rethinking residual distribution in locate-then-edit model editing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yanhong Li, Min Yang, Xiping Hu, and Chengming Li. 2025b. Forget for get: A lightweight two-phase gradient method for knowledge editing in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7604–7623.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

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

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

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*.

Meta. 2024. Llama 3. Large language model release.

Kshitij Mishra, Tamer Soliman, Anil Ramakrishna, Aram Galstyan, and Anoop Kumar. 2024. Correcting language model outputs by editing salient layers. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1295–1305.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Haowen Pan, Xiaozhi Wang, Yixin Cao, Zenglin Shi, Xun Yang, Juanzi Li, and Meng Wang. 2025. Precise localization of memories: A fine-grained neuron-level knowledge editing technique for llms. In *The Thirteenth International Conference on Learning Representations*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pages 353–355.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 billion parameter autoregressive language model.

Jianchen Wang, Zhouhong Gu, Xiaoxuan Zhu, Lin Zhang, Haoning Ye, Zhuozhi Xiong, Sihang Jiang, Hongwei Feng, and Yanghua Xiao. 2025. The missing piece in model editing: A deep dive into the hidden damage brought by model editing. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the North American*

chapter of the association for computational linguistics: human language technologies, volume 1 (long papers), pages 1112–1122.

Derong Xu, Ziheng Zhang, Zhihong Zhu, Zhenxi Lin, Qidong Liu, Xian Wu, Tong Xu, Wanyu Wang, Yuyang Ye, Xiangyu Zhao, and 1 others. 2024. Editing factual knowledge and explanatory ability of medical large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2660–2670.

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, Singapore. Association for Computational Linguistics.

Junsang Yoon, Akshat Gupta, and Gopala Anumanchipalli. 2024. Is bigger edit batch size always better?–an empirical study on model editing with llama-3. *arXiv preprint arXiv:2405.00664*.

Appendix

A Theoretical Analysis

Local First-Order Approximation. Let $\boldsymbol{\theta}^{(t)}$ denote the model parameters at a reference point (before the current allocation or editing step). Our goal is to match the *target residual* evaluated at the last layer L in the key region $\mathcal{L}$. Following Appendix D, we present the derivation for a single column ($u = 1$), and the batched case follows column-wise for notational simplicity. Concretely, let $\boldsymbol{h}_L(\boldsymbol{\delta}) \in \mathbb{R}^d$ denote the final-layer state as a function of the injected residual shares $\{\boldsymbol{\delta}_l\}_{l\in\mathcal{L}}$, and define the target residual as

$$\boldsymbol{\mathcal{R}} \;:=\; \boldsymbol{h}_L^\star - \boldsymbol{h}_L(\mathbf{0}) \in \mathbb{R}^d. \tag{8}$$

We distribute this target by injecting layer-wise residual shares $\{\boldsymbol{\delta}_l\}_{l\in\mathcal{L}}$ into the residual stream. We model allocation as an additive intervention at layer l:

$$\tilde{\boldsymbol{r}}_l = \boldsymbol{r}_l + \boldsymbol{\delta}_l, \tag{9}$$

where $\boldsymbol{r}_l \in \mathbb{R}^d$ is the (pre-intervention) residual representation. Under this intervention, we define the (local) sensitivity from layer l to the final-layer output space as the Jacobian of the last-layer state function $\boldsymbol{h}_L(\boldsymbol{\delta})$ with respect to the intervention variable $\boldsymbol{\delta}_l$:

$$\boldsymbol{M}_l := \boldsymbol{M}_l(\boldsymbol{\theta}^{(t)}) = \left.\frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{\delta}_l}\right|_{\boldsymbol{\delta}=\mathbf{0},\ \boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \in \mathbb{R}^{d\times d}. \tag{10}$$

By the chain rule and $\tilde{\boldsymbol{r}}_l = \boldsymbol{r}_l + \boldsymbol{\delta}_l$, we have

$$\frac{\partial \boldsymbol{h}_L}{\partial \boldsymbol{\delta}_l} = \frac{\partial \boldsymbol{h}_L}{\partial \tilde{\boldsymbol{r}}_l}\frac{\partial \tilde{\boldsymbol{r}}_l}{\partial \boldsymbol{\delta}_l} = \frac{\partial \boldsymbol{h}_L}{\partial \tilde{\boldsymbol{r}}_l}, \tag{11}$$

since $\frac{\partial \tilde{\boldsymbol{r}}_l}{\partial \boldsymbol{\delta}_l} = \boldsymbol{I}$. Therefore, $\boldsymbol{M}_l$ can be interpreted as the local propagation operator of the residual state around $(\boldsymbol{\delta} = \mathbf{0}, \boldsymbol{\theta} = \boldsymbol{\theta}^{(t)})$.

In practice, feasible interventions are restricted to a controllable linear subspace $\mathcal{S}_l \subseteq \mathbb{R}^d$. To derive the closed-form solution in Theorem 1, we analyze the unconstrained case $\mathcal{S}_l = \mathbb{R}^d$. If $\mathcal{S}_l \subsetneq \mathbb{R}^d$, one may parameterize $\boldsymbol{\delta}_l = \boldsymbol{B}_l \boldsymbol{z}_l$ with $\mathrm{Range}(\boldsymbol{B}_l) = \mathcal{S}_l$; choosing $\boldsymbol{B}_l$ to have orthonormal columns, the same derivation applies by replacing $\boldsymbol{M}_l$ with $\boldsymbol{M}_l\boldsymbol{B}_l$ in the linearized constraint (and the objective $\|\boldsymbol{\delta}_l\|_2^2$ remains $\|\boldsymbol{z}_l\|_2^2$).

Assuming $\boldsymbol{h}_L(\boldsymbol{\delta})$ is locally twice differentiable with respect to $\boldsymbol{\delta}$ in a neighborhood of $\boldsymbol{\delta} = \mathbf{0}$, and its Hessian is bounded (with $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$ fixed), the induced change of the final-layer state $\Delta\boldsymbol{h}_L := \boldsymbol{h}_L(\boldsymbol{\delta}) - \boldsymbol{h}_L(\mathbf{0})$ satisfies the first-order Taylor expansion:

$$\Delta\boldsymbol{h}_L = \sum_{l\in\mathcal{L}} \boldsymbol{M}_l\,\boldsymbol{\delta}_l + \mathcal{O}\left(\sum_{l\in\mathcal{L}} \|\boldsymbol{\delta}_l\|_2^2\right). \tag{12}$$

Accordingly, matching the target residual $\boldsymbol{\mathcal{R}}$ (i.e., enforcing $\boldsymbol{h}_L(\boldsymbol{\delta}) = \boldsymbol{h}_L^\star$) yields the first-order consistency condition

$$\sum_{l\in\mathcal{L}} \boldsymbol{M}_l\,\boldsymbol{\delta}_l = \boldsymbol{\mathcal{R}}, \tag{13}$$

which we impose as an equality constraint in our allocation formulation and which is accurate up to second-order terms in the small-injection regime.

Furthermore, we assume the sensitivity matrices satisfy a local Lipschitz condition with respect to the parameters:

$$\left\|\boldsymbol{M}_l(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}) - \boldsymbol{M}_l(\boldsymbol{\theta})\right\|_2 \le L_M \|\Delta\boldsymbol{\theta}\|_2. \tag{14}$$

Thus, within a single allocation step we assume a local linearity regime where propagation matrices do not vary significantly:

$$\left\|\boldsymbol{M}_l(\boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}) - \boldsymbol{M}_l(\boldsymbol{\theta}^{(t)})\right\|_2 \le L_M \|\Delta\boldsymbol{\theta}\|_2, \tag{15}$$

for sufficiently small $\|\Delta\boldsymbol{\theta}\|_2$.

A.1 Identity Approximation of M_l

Within the local first-order framework above, $\boldsymbol{M}_l$ characterizes the sensitivity from a residual injection at layer l to the final-layer representation.

Dataset	Sensitivity Variant	Eff. ↑	Gen. ↑	Spe. ↑	Flu. ↑	Consis. ↑	Time
CounterFact	Explicit $\boldsymbol{M}_l$	99.79	96.11	72.89	615.20	41.17	205.31s
	Approx. $\boldsymbol{I}$	99.63	95.92	72.68	614.53	41.02	176.55s
ZsRE	Explicit $\boldsymbol{M}_l$	98.08	91.26	27.33	–	–	198.47s
	Approx. $\boldsymbol{I}$	97.93	91.09	26.92	–	–	161.15s

Table 4: Comparison between explicit sensitivity computation and the identity approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$ on GPT-2 XL (1.5B). Explicit computation yields only marginal performance gains while incurring higher time overhead.

For scalable implementation in large models, we adopt the approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$ in the allocation step. This approximation is introduced for computational tractability; the reduction of accumulated drift in CAKE instead comes from layer-wise dynamic residual recomputation and adaptive allocation, rather than from assuming exact linear superposition across layers. As shown in Eq. (12), under the small-injection regime, cross-layer residual propagation can be locally approximated by a first-order operator $\boldsymbol{M}_l$, with approximation error captured by the retained higher-order term. Explicitly computing the sensitivity matrix is theoretically more accurate, but substantially more expensive for large models. The identity approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$ therefore provides a computationally efficient surrogate within the same local first-order framework.

To assess the practical impact of this approximation, we compare two allocation variants on GPT-2 XL (1.5B): (i) using the explicit sensitivity matrix $\boldsymbol{M}_l$, and (ii) using the identity approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$. We follow the same experimental pipeline as in the main experiments and report results on CounterFact and on a randomly sampled subset of 2000 edits from ZsRE. Table 4 shows that explicitly computing the Jacobian yields only marginal gains across efficacy, generalization, and specificity, while incurring a larger time overhead. These results suggest that the identity approximation $\boldsymbol{M}_l \approx \boldsymbol{I}$ offers a practical trade-off between approximation fidelity and computational efficiency.

A.2 Proof of Theorem 1

proof. We solve Eq. 4 using Lagrange multipliers. The Lagrangian is given by:

$$\mathcal{L}(\boldsymbol{\delta}, \boldsymbol{\lambda}) = \sum_{l \in \mathcal{L}} \frac{1}{2w_l} \|\boldsymbol{\delta}_l\|^2 - \boldsymbol{\lambda}^T \left(\sum_{l \in \mathcal{L}} \boldsymbol{M}_l \boldsymbol{\delta}_l - \boldsymbol{\mathcal{R}} \right). \quad (16)$$

Note that the objective is strictly convex in $\{\boldsymbol{\delta}_l\}$ since $w_l > 0$, and the constraint is linear; hence the problem admits a unique global optimum characterized by the KKT conditions. Taking the derivative with respect to $\boldsymbol{\delta}_l$ and setting it to zero yields:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\delta}_l} = \frac{1}{w_l} \boldsymbol{\delta}_l - \boldsymbol{M}_l^T \boldsymbol{\lambda} = 0, \quad (17)$$

which gives

$$\boldsymbol{\delta}_l = w_l \boldsymbol{M}_l^T \boldsymbol{\lambda}. \quad (18)$$

This condition reveals a key insight: the optimal update magnitude for layer l is proportional to the product of its causal weight w_l and its local sensitivity $\boldsymbol{M}_l^T$. Substituting this expression back into the constraint in $\sum_{l \in \mathcal{L}} \boldsymbol{M}_l \boldsymbol{\delta}_l = \boldsymbol{\mathcal{R}}$ yields

$$\boldsymbol{\lambda} = \left(\sum_{j \in \mathcal{L}} w_j \boldsymbol{M}_j \boldsymbol{M}_j^T \right)^{-1} \boldsymbol{\mathcal{R}}. \quad (19)$$

Plugging $\boldsymbol{\lambda}$ back gives the closed-form solution in Eq. 5.

A.3 Proof of theorem 2

proof. Following the proof of Theorem 4.1 in Li et al. (2025a), we express the weight-shift error as the least-squares projection of a residual mismatch. Under the local first-order linearization framework in Sec. 3.2.2, the layer-wise write-in is solved by a least-squares projection of the desired residual onto the update $\boldsymbol{\Delta}_l$. In particular, the ideal update at layer l fits $\boldsymbol{\mathcal{R}}_l^*$, i.e., $\boldsymbol{\Delta}_l^* = \boldsymbol{\mathcal{R}}_l^* \boldsymbol{Q}$, while the actual update fits the dynamically allocated residual: $\boldsymbol{\mathcal{R}}_l$ and $\boldsymbol{\Delta}_l = \boldsymbol{\mathcal{R}}_l \boldsymbol{Q}$, all norms $\|\cdot\|_2$ are induced 2-norms. Thus:

$$\|\boldsymbol{\Delta}_l^* - \boldsymbol{\Delta}_l\|_2 = \|(\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_l)\boldsymbol{Q}\|_2 \quad (20)$$

$$\leq \|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_l\|_2 \|\boldsymbol{Q}\|_2 \quad (21)$$

We expand the norm using the triangle inequality by introducing and subtracting $\boldsymbol{\mathcal{R}}_L$:

$$\begin{aligned} \|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_l\|_2 &= \|\boldsymbol{\mathcal{R}}_l^* - \eta_l \boldsymbol{\mathcal{R}}_L\|_2 \\ &= \|(\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L) + (1 - \eta_l)\boldsymbol{\mathcal{R}}_L\|_2 \\ &\leq \|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L\|_2 + (1 - \eta_l)\|\boldsymbol{\mathcal{R}}_L\|_2. \end{aligned} \quad (22)$$

This establishes Eq. (7). In particular, the first term $\|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L\|_2$ (Term A) captures the *projection mismatch* between the layer-l ideal residual

$\mathcal{R}_l^*$ (i.e., the best residual achievable by updating only layer l) and the global target residual $\mathcal{R}_L$. The second term $(1-\eta_l)\|\mathcal{R}_L\|_2$ (Term B) arises from the remaining-budget ratio: only an η_l fraction of the global residual is allocated to layer l, leaving an unallocated portion that contributes additively to the bound. Finally, $\|\boldsymbol{Q}\|_2$ quantifies the amplification of residual deviations induced by the local least-squares write-in operator.

Remark (Alignment with Rethinking). This theorem strictly generalizes the bound in Li et al. (2025a). If weights are uniform ($w_l = 1$), then $\eta_l = \frac{1}{L-l+1}$. Consequently, the propagation term becomes $(1-\eta_l) = \frac{L-l}{L-l+1}$. This exactly recovers the "Exact Bound" (Eq. 19) and by relaxation implies the "Loose Bound" (Eq. 20) in Li et al. (2025a).

A.4 Theoretical Comparison of Residual Allocation Strategies.

Comparison with MEMIT. MEMIT implies $w_l = 1/L$. (1) **High Propagation:** For early layers, $\eta_l \approx 0$, making Term B dominance ($1-\eta_l \approx 1$). The noise (scaled by $\|\mathcal{R}_L\|_2$) accumulates linearly with depth. (2) **High Mismatch:** It assigns equal updates even to layers where attribution is low (i.e., $\|\mathcal{R}_l^* - \mathcal{R}_l\|_2$ is large), inflating Term A.

Comparison with BLUE. BLUE selects a subset (e.g., boundary layers), implying $w_l \in \{0, 1\}$. (1) **Low Propagation:** For selected layers, $\eta_l = 1$ (if it's the last layer), effectively zeroing out Term B. (2) **Risk of Mismatch:** BLUE relies on static layer selection. If the specific knowledge instance requires editing a middle layer (where BLUE sets $w_l = 0$), the local mismatch $\|\mathcal{R}_l^* - \mathcal{R}_l\|_2$ for the remaining layers becomes unmanageable.

A.5 Proof for Attribution-Based Weighting

This appendix provides a structured theoretical justification for why CAKE reduces the generalized error bound in Eq. 7 under the *Projection Alignment* form of the *Causal Alignment Assumption*, and why concentrating update weights on high-attribution layers controls error amplification by reducing the effective depth of sequential editing.

A.5.1 Setup and Assumptions

Layer-wise projection mismatch (Term A). Under the local first-order approximation, perturbing only layer l yields $\Delta \boldsymbol{h}_L \approx \boldsymbol{M}_l \boldsymbol{\delta}_l$ with $\boldsymbol{\delta}_l \in \mathcal{S}_l$, where $\mathcal{S}_l \subseteq \mathbb{R}^d$ is a controllable *linear subspace*. Consider the best approximation of $\mathcal{R}_L$ realizable by updating only layer l:

$$\widehat{\mathcal{R}}_l := \arg\min_{\boldsymbol{\delta}_l \in \mathcal{S}_l} \|\mathcal{R}_L - \boldsymbol{M}_l \boldsymbol{\delta}_l\|_2. \quad (23)$$

Since $\boldsymbol{M}_l(\mathcal{S}_l) := \{\boldsymbol{M}_l \boldsymbol{\delta} : \boldsymbol{\delta} \in \mathcal{S}_l\}$ is a linear subspace of $\mathbb{R}^d$, the minimizer corresponds to the orthogonal projection of $\mathcal{R}_L$ onto $\boldsymbol{M}_l(\mathcal{S}_l)$, i.e.,

$$\widehat{\mathcal{R}}_l = \boldsymbol{P}_l \mathcal{R}_L, \boldsymbol{P}_l := \mathrm{Proj}_{\boldsymbol{M}_l(\mathcal{S}_l)} \in \mathbb{R}^{d \times d}. \quad (24)$$

(When $\mathcal{S}_l$ equals the full intervention space, $\boldsymbol{M}_l(\mathcal{S}_l) = \mathrm{Range}(\boldsymbol{M}_l)$ and one recovers $\boldsymbol{P}_l = \boldsymbol{M}_l \boldsymbol{M}_l^+$.)

We interpret the ideal per-layer residual as $\mathcal{R}_l^* := \widehat{\mathcal{R}}_l$. Therefore, Term A is exactly the *projection mismatch*:

$$m_l := \|\mathcal{R}_l^* - \mathcal{R}_L\|_2 = \|(\boldsymbol{I} - \boldsymbol{P}_l)\mathcal{R}_L\|_2 \geq 0. \quad (25)$$

Projection Alignment Assumption. We assume layer-wise attribution scores reflect causal relevance in that higher-attribution layers incur smaller projection mismatch:

$$s_i \geq s_j \;\Rightarrow\; m_i \leq m_j, \forall\, i, j \in \{1, \ldots, L\}. \quad (26)$$

We further restrict attention to attribution-prioritized allocations, i.e., weight vectors whose ordering is consistent with the attribution scores:

$$\mathcal{W}_\uparrow := \left\{ \boldsymbol{w} \in \Delta^{L-1} : \; s_i \geq s_j \Rightarrow w_i \geq w_j \right\}, \quad (27)$$

where Δ^{L-1} is the probability simplex over L layers. This family includes CAKE ($w_l \propto s_l$).

A.5.2 Sanity Check for the Projection Alignment Assumption

Corollary 1 is conditional on the Projection Alignment Assumption in Eq. (26), which states that layers with higher attribution scores should incur smaller projection mismatch. To examine whether this ordering is supported in practice, we perform a sanity check using a feasible empirical proxy for mismatch based on the fitting error of a single-layer-only edit. We randomly sample 200 editing instances from CounterFact and ZsRE, and conduct the following procedure for each instance:

- **Compute causal scores.** We obtain the layer-wise causal score vector $\{s_l\}_{l=1}^L$ using the same attribution procedure as in the main method.
- **Compute single-layer fitting errors.** For each layer l, we only allow that layer to be edited, obtain the single-layer optimal update, and compute its fitting error to the global residual $\mathcal{R}_L$. We denote this mismatch proxy by $\tilde{m}_l$.

Dataset	Model	Kendall Correlation (ρ_k)
CounterFact	LLaMA-3 (8B)	0.80
CounterFact	GPT-J (6B)	0.73
ZsRE	LLaMA-3 (8B)	0.80
ZsRE	GPT-J (6B)	0.60

Table 5: Sanity check for the Projection Alignment Assumption.

- **Evaluate ranking consistency.** For each instance, we compute the Kendall rank correlation $\rho_k = \mathrm{corr}_{\mathrm{Kendall}}(s_l, -\tilde{m}_l)$, which measures whether larger attribution scores tend to correspond to smaller fitting errors.

Table 5 reports the average Kendall correlation across sampled instances. We observe consistently positive correlations across datasets and models, indicating that layers with higher causal scores tend to yield smaller single-layer fitting errors. These results provide empirical support for using attribution scores as a practical ordering signal in the Projection Alignment Assumption.

A.5.3 Proof of Corollary 1

We quantify the overall contribution of Term A under a weight allocation $\boldsymbol{w} = (w_1, \ldots, w_L)$ by the weighted aggregate

$$A(\boldsymbol{w}) \;:=\; \sum_{l=1}^{L} w_l\, m_l, m_l := \|\boldsymbol{\mathcal{R}}_l^* - \boldsymbol{\mathcal{R}}_L\|_2 \geq 0. \quad (28)$$

Fix any pair (i, j) such that $s_i \geq s_j$. For any $\varepsilon > 0$ satisfying the feasibility condition $\varepsilon \leq w_j$, define a new allocation $\boldsymbol{w}'$ by modifying only the two coordinates i and j:

$$w_i' = w_i + \varepsilon, w_j' = w_j - \varepsilon, w_k' = w_k \;\; (k \neq i, j). \quad (29)$$

We now derive $A(\boldsymbol{w}') - A(\boldsymbol{w})$ directly from the definition:

$$\begin{aligned} A(\boldsymbol{w}') - A(\boldsymbol{w}) &= \sum_{l=1}^{L} w_l' m_l \;-\; \sum_{l=1}^{L} w_l m_l \\ &= \sum_{l=1}^{L} (w_l' - w_l)\, m_l. \end{aligned} \quad (30)$$

Since $w_k' - w_k = 0$ for all $k \neq i, j$, the sum in Eq. (30) reduces to two terms:

$$\begin{aligned} A(\boldsymbol{w}') - A(\boldsymbol{w}) &= (w_i' - w_i)m_i + (w_j' - w_j)m_j \\ &= \varepsilon m_i + (-\varepsilon)m_j \\ &= \varepsilon(m_i - m_j). \end{aligned} \quad (31)$$

Under the Projection Alignment condition in Eq. (26), $s_i \geq s_j$ implies $m_i \leq m_j$, hence $m_i - m_j \leq 0$. Together with $\varepsilon > 0$, Eq. (31) yields

$$A(\boldsymbol{w}') - A(\boldsymbol{w}) = \varepsilon(m_i - m_j) \leq 0. \quad (32)$$

Moreover, if the alignment is strict ($m_i < m_j$) and $\varepsilon > 0$, then $A(\boldsymbol{w}') < A(\boldsymbol{w})$; if $m_i = m_j$, then $A(\boldsymbol{w}') = A(\boldsymbol{w})$. Finally, a score-prioritized reweighting can be viewed as a finite sequence of such feasible two-coordinate adjustments that shift a small amount of weight from a lower-score layer to a higher-score layer. Since each adjustment does not increase $A(\boldsymbol{w})$ by Eq. (32), the entire reweighting process cannot increase $A(\boldsymbol{w})$, and it decreases $A(\boldsymbol{w})$ whenever at least one step satisfies $m_i < m_j$ with $\varepsilon > 0$. This establishes Corollary 1.

A.5.4 Proof of Corollary 2

Weight-induced concentration and effective support. We quantify the concentration of the resulting allocation w by the effective update size:

$$N_{\mathrm{eff}}(\boldsymbol{w}) \;:=\; \frac{1}{\|\boldsymbol{w}\|_2^2}. \quad (33)$$

CAKE assigns layer weights as a normalized monotone transform of the causal scores s_l in Eq. 2, so layers with smaller scores ($s_l \ll s_{\max}$) receive negligible weights. Intuitively, a more peaked score profile induces a more concentrated w and hence a smaller $N_{\mathrm{eff}}(\boldsymbol{w})$.

To relate the aggregated propagation term to the weight concentration, we introduce an *effectively-updated* layer set. Let

$$\gamma := \frac{1}{2N_{\mathrm{eff}}(\boldsymbol{w})}, \mathcal{L}_{\mathrm{eff}} := \{\, l :\; w_l \geq \gamma \,\}. \quad (34)$$

We define the aggregated propagation contribution on these layers as

$$\mathcal{E}_B(\boldsymbol{w}) := \sum_{l \in \mathcal{L}_{\mathrm{eff}}} (1 - \eta_l), \eta_l = \frac{w_l}{\sum_{k=l}^{L} w_k}. \quad (35)$$

Since $0 \leq \eta_l \leq 1$, we have $1 - \eta_l \leq 1$, hence

$$\mathcal{E}_B(\boldsymbol{w}) \leq |\mathcal{L}_{\mathrm{eff}}|. \quad (36)$$

By the definition of $\mathcal{L}_{\mathrm{eff}}$, for all $l \in \mathcal{L}_{\mathrm{eff}}$ we have $w_l \geq \gamma$. Therefore,

$$1 = \sum_{l=1}^{L} w_l \;\geq\; \sum_{l \in \mathcal{L}_{\mathrm{eff}}} w_l \;\geq\; |\mathcal{L}_{\mathrm{eff}}| \cdot \gamma, \quad (37)$$

which yields

$$|\mathcal{L}_{\mathrm{eff}}| \leq \frac{1}{\gamma} = 2N_{\mathrm{eff}}(\boldsymbol{w}). \quad (38)$$

Combining Eq. (36)–(38), we obtain

$$\mathcal{E}_B(\boldsymbol{w}) \leq 2N_{\mathrm{eff}}(\boldsymbol{w}). \quad (39)$$

Implication for Term B. Assume $\|\boldsymbol{Q}_l\|_2 \leq \boldsymbol{Q}_{\max}$ for all layers. Summing the Term B part of Eq. (7) over $l \in \mathcal{L}_{\text{eff}}$ gives

$$\begin{aligned}
&\sum_{l\in\mathcal{L}_{\text{eff}}} (1-\eta_l)\|\boldsymbol{\mathcal{R}}_L\|_2\,\|\boldsymbol{Q}_l\|_2 \\
&\leq \boldsymbol{Q}_{\max}\|\boldsymbol{\mathcal{R}}_L\|_2 \sum_{l\in\mathcal{L}_{\text{eff}}} (1-\eta_l) \\
&= \boldsymbol{Q}_{\max}\|\boldsymbol{\mathcal{R}}_L\|_2 \cdot \mathcal{E}_B(w) \\
&\leq 2\boldsymbol{Q}_{\max}\|\boldsymbol{\mathcal{R}}_L\|_2\, N_{\text{eff}}(w).
\end{aligned} \tag{40}$$

Thus, the aggregated propagation-driven contribution scales with $N_{\text{eff}}(\boldsymbol{w})$ rather than the full depth L, establishing Corollary 2.

A.5.5 Proof of Corollary 3

We show that weight concentration reduces the cumulative amplification induced by projection operators when aggregating the per-layer bound.

Assume $\|\boldsymbol{Q}_l\|_2 \leq \boldsymbol{Q}_{\max}$ for all layers. By Eq. (38),

$$\sum_{l\in\mathcal{L}_{\text{eff}}} \|\boldsymbol{Q}_l\|_2 \leq \boldsymbol{Q}_{\max}|\mathcal{L}_{\text{eff}}| \leq 2\boldsymbol{Q}_{\max}N_{\text{eff}}(\boldsymbol{w}). \tag{41}$$

Moreover, by triangle inequality, summing Eq. (7) over $l \in \mathcal{L}_{\text{eff}}$ yields

$$\sum_{l\in\mathcal{L}_{\text{eff}}} \|\boldsymbol{\Delta}_l^* - \boldsymbol{\Delta}_l\|_2 \leq \sum_{l\in\mathcal{L}_{\text{eff}}} \big(m_l + (1-\eta_l)\|\boldsymbol{\mathcal{R}}_L\|_2\big)\|\boldsymbol{Q}_l\|_2. \tag{42}$$

Eq. (41) shows that the cumulative operator contribution in Eq. (42) is controlled by $N_{\text{eff}}(\boldsymbol{w})$. Therefore, concentrating weights on a small set of high-attribution layers reduces the effective depth of the layer-wise procedure and mitigates error amplification, establishing Corollary 3.

B Experimental Setup

In this section, we provide a comprehensive description of the experimental configuration used in this paper. We detail the datasets, evaluation metrics, model architectures, implementation details, baseline methods, and the sequential editing protocol. These settings complement the experimental results presented in Section 5 and are designed to ensure reproducibility and fair comparison with prior knowledge editing methods.

B.1 Datasets

ZsRE. ZsRE (Levy et al., 2017) is a question-answering based knowledge editing dataset, where each instance corresponds to a question prompt encoding a subject–relation query (s_i, r_i) with a target object o_i. Following standard protocols (Meng et al., 2022), ZsRE provides: (i) the original question for evaluating editing efficacy, (ii) paraphrased or semantically equivalent questions for evaluating generalization, and (iii) out-of-scope (locality) questions involving semantically related but distinct facts for evaluating specificity. ZsRE primarily evaluates locality preservation and robustness to paraphrasing.

CounterFact. CounterFact (Meng et al., 2022) is a counterfactual editing benchmark that contrasts factual and counterfactual statements. It constructs out-of-scope samples by replacing subject entities with semantically similar alternatives sharing the same predicate. In addition to efficacy, generalization, and specificity, CounterFact includes multiple natural language generation templates, enabling evaluation of generation quality through fluency and consistency metrics.

B.2 Evaluation Metrics

B.2.1 ZsRE Metrics

We follow prior work (Meng et al., 2023; Mitchell et al., 2022; Meng et al., 2022) to evaluate factual editing on ZsRE using efficacy, generalization, and specificity.

Efficacy. Efficacy evaluates whether the edited model correctly recalls the target fact under the original query. Let f_θ denote the edited model. For each edit instance i, let x_i denote the original question prompt that expresses the subject–relation query (s_i, r_i), and let o_i denote the edited target object. Efficacy is defined as the fraction of instances for which the edited object is the most probable prediction:

$$\mathbb{E}_i\Big[\arg\max_o P_{f_\theta}(o \mid x_i) = o_i\Big]. \tag{43}$$

Generalization. Generalization measures whether the edit generalizes to semantically equivalent prompts. For each instance i, let $N(x_i)$ denote the set of paraphrased questions that preserve the same underlying subject–relation semantics as x_i, following the construction of ZsRE. Generalization is computed as the average top-1 accuracy over these equivalent prompts:

$$\mathbb{E}_i\,\mathbb{E}_{\tilde{x}_i\in N(x_i)}\Big[\arg\max_o P_{f_\theta}(o \mid \tilde{x}_i) = o_i\Big]. \tag{44}$$

Specificity. Specificity evaluates whether the edit remains localized and does not affect the model's predictions on unrelated facts. For each instance i, let $O(x_i)$ denote the set of out-of-scope (locality) prompts provided by ZsRE, and let o_i^c denote the model's original (pre-edit) top-1 prediction on these prompts. Specificity is defined as the fraction of locality prompts for which the post-edit model preserves the original prediction:

$$\mathbb{E}_i \, \mathbb{E}_{\bar{x}_i \in O(x_i)} \Big[\arg\max_o P_{f_\theta}(o \mid \bar{x}_i) = o_i^c \Big]. \quad (45)$$

B.2.2 CounterFact Metrics

Following prior work (Meng et al., 2022, 2023), CounterFact evaluates factual editing by directly comparing a counterfactual target with its original factual counterpart. Rather than relying on top-1 accuracy, this benchmark measures whether probability mass is shifted from the original fact to the edited fact under different prompt distributions. We use the same notation as above for original, paraphrased, and neighborhood prompts. For each instance i, let f_θ denote the edited model, x_i the original prompt expressing the subject–relation query (s_i, r_i), o_i the edited (counterfactual) target object, and o_i^c the original factual object provided by CounterFact.

Efficacy. Efficacy reflects whether the editing operation successfully shifts the model's preference toward the counterfactual fact on the original prompt. It is defined as the fraction of cases in which the edited object receives higher probability than the original factual object:

$$\mathbb{E}_i \Big[P_{f_\theta}(o_i \mid x_i) > P_{f_\theta}(o_i^c \mid x_i) \Big]. \quad (46)$$

Generalization. Generalization evaluates whether this preference shift persists under paraphrased queries that express the same underlying fact. Let $\tilde{x}_i \in N(x_i)$ denote a paraphrased variant of the original prompt. The metric is computed as:

$$\mathbb{E}_i \, \mathbb{E}_{\tilde{x}_i \in N(x_i)} \Big[P_{f_\theta}(o_i \mid \tilde{x}_i) > P_{f_\theta}(o_i^c \mid \tilde{x}_i) \Big]. \quad (47)$$

Specificity. Specificity assesses locality by testing whether probability mass assigned to unrelated facts is preserved. Let $\bar{x}_i \in O(x_i)$ denote a neighborhood prompt involving a semantically related but factually distinct subject. Specificity is defined as the fraction of such prompts for which the model assigns higher probability to the original factual object than to the edited one:

$$\mathbb{E}_i \, \mathbb{E}_{\bar{x}_i \in O(x_i)} \Big[P_{f_\theta}(o_i^c \mid \bar{x}_i) > P_{f_\theta}(o_i \mid \bar{x}_i) \Big]. \quad (48)$$

Fluency. Fluency evaluates generation quality by measuring excessive repetition in model outputs. Following prior work, we adopt an entropy-based metric over bi-gram and tri-gram frequency distributions:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \quad (49)$$

where $g_n(k)$ denotes the empirical frequency of the n-gram k.

Consistency. Consistency measures whether generated text remains semantically aligned with external knowledge describing the edited fact. It is computed as the cosine similarity between TF–IDF representations of the model-generated output y_i and a reference Wikipedia passage y_i^{wiki} associated with the target object:

$$\text{Consis.} = \cos\Big(\text{tfidf}(y_i), \text{tfidf}(y_i^{\text{wiki}}) \Big). \quad (50)$$

B.3 Baseline Methods

ROME. ROME (Meng et al., 2022) is a locate-and-edit method that performs single-layer knowledge editing by directly modifying the feed-forward network (FFN) weights responsible for factual recall. It identifies the most influential layer via causal tracing and applies a rank-one update to enforce the edited association. ROME serves as a sparse editing baseline that prioritizes locality but often struggles with generalization and complex edits.

MEMIT. MEMIT (Meng et al., 2023) extends ROME to the multi-layer setting by distributing a global editing residual uniformly across a contiguous range of critical layers. Each layer update is computed via a least-squares objective over key–value pairs. MEMIT represents a dense editing paradigm and is a strong baseline for mass knowledge insertion, but is susceptible to error accumulation under sequential edits.

PRUNE. PRUNE (Ma et al., 2025) addresses stability issues in sequential editing by constraining the spectral perturbation of edited weight matrices. By retaining only eigen-directions associated with small condition numbers, PRUNE aims to mitigate interference between edits and preserve the model's general capabilities over time.

RECT. RECT (Gu et al., 2024) introduces regularization terms during the editing process to prevent excessive deviation from the pre-trained pa-

rameters. Unlike locate-and-edit methods that focus solely on factual accuracy, RECT explicitly balances editing success with the preservation of general language and reasoning abilities.

AlphaEdit. AlphaEdit (Fang et al., 2025) extends MEMIT by introducing null-space constrained updates to mitigate interference during multi-layer knowledge editing. Specifically, AlphaEdit projects the editing residual onto the null space of previously edited or protected representations, ensuring that parameter updates minimally affect unrelated or preserved knowledge. This design significantly improves stability under sequential editing, while retaining the flexibility of multi-layer residual allocation.

AlphaEdit$_{\text{BLUE}}$ AlphaEdit$_{\text{BLUE}}$ (Li et al., 2025a) further modifies the AlphaEdit framework by restricting editing operations to empirically identified boundary layers. Motivated by observations of depth-dependent sensitivity in Transformer models, AlphaEdit$_{\text{BLUE}}$ applies edits only to a small subset of layers near the upper and lower boundaries of the network. While this restriction can reduce interference and improve efficiency, it introduces a strong structural prior that may limit adaptability under heterogeneous causal contributions across layers.

All baseline methods are implemented using their official codebases or verified re-implementations. Hyperparameters are set according to the original papers or recommended configurations, without additional tuning in favor of CAKE. For consistency, all methods operate on the same model checkpoints, use identical covariance statistics, and follow the same sequential editing protocol described in Section 5.

B.4 Evaluation of General Capability

To examine whether sequential knowledge editing inadvertently degrades general language abilities, we evaluate edited models on a collection of standard natural language understanding benchmarks. These datasets are not involved in the editing process and are chosen to span diverse linguistic competencies, including sentiment recognition, semantic similarity, grammaticality, and logical inference.

SST. We use the SST benchmark (Socher et al., 2013) to assess whether sequential editing affects sentence-level semantic interpretation. The task requires predicting binary sentiment labels from short sentences with varied syntactic structures. Because SST is sensitive to subtle distributional shifts, performance changes can reveal unintended degradation in basic language understanding. We report classification accuracy.

MRPC. MRPC (Dolan and Brockett, 2005) evaluates the model's ability to judge semantic equivalence between sentence pairs. This task is particularly relevant for detecting whether editing disrupts fine-grained semantic alignment across paraphrases. Given the skewed label distribution, we follow standard practice and report both accuracy and F1 score.

MMLU. To probe broad-domain knowledge and reasoning stability, we include the MMLU benchmark (Hendrycks et al., 2021). MMLU covers a wide range of subjects and requires both factual recall and procedural reasoning. We evaluate models under zero-shot and few-shot settings to isolate the impact of editing from task-specific adaptation. Accuracy is used as the primary metric.

RTE. RTE (Bentivogli et al., 2009) is used to test whether logical inference capabilities remain stable after repeated edits. The task involves determining entailment relations between a premise and a hypothesis. We report accuracy, which directly reflects inference correctness.

CoLA. We use CoLA (Warstadt et al., 2019) to evaluate sensitivity to grammatical acceptability. As this benchmark focuses on subtle syntactic phenomena, it serves as a useful indicator of whether editing introduces unintended distortions in linguistic form. We report Matthews correlation coefficient (MCC), following common evaluation practice.

NLI. To further assess semantic reasoning robustness, we evaluate models on a natural language inference benchmark (Williams et al., 2018). This task requires distinguishing entailment, contradiction, and neutral relations between sentence pairs. Performance on NLI reflects the model's sensitivity to semantic and pragmatic cues, and we report accuracy as the evaluation metric.

B.5 Residual Drift Metric

This section provides a detailed definition of the *residual drift* metric used to probe recursive error accumulation under sequential multi-layer editing.

We consider a sequential editing process in which model parameters are updated layer by layer. After editing the l-th layer, we evaluate the effect of this update in the final-layer residual space. Let $\boldsymbol{z}$ denote the desired target representation at the final layer, and let $\boldsymbol{h}_{\text{pre}}^{(l)}$ and $\boldsymbol{h}_{\text{post}}^{(l)}$ denote the final-layer activations before and after editing layer l, respectively. We define the pre-edit residual as

$$\boldsymbol{\mathcal{R}}_{\text{pre}}^{(l)} = \boldsymbol{z} - \boldsymbol{h}_{\text{pre}}^{(l)}, \tag{51}$$

and the post-edit residual as

$$\boldsymbol{\mathcal{R}}_{\text{post}}^{(l)} = \boldsymbol{z} - \boldsymbol{h}_{\text{post}}^{(l)}. \tag{52}$$

Expected and Actual Residual Reduction. Given a layer-wise residual allocation ratio η_l, the *expected* residual reduction induced by editing layer l is defined as

$$\boldsymbol{\Delta}_{\text{exp}}^{(l)} = \eta_l \, \boldsymbol{\mathcal{R}}_{\text{pre}}^{(l)}. \tag{53}$$

This term represents the theoretical amount of residual that should be removed by the update at layer l under the assumed residual allocation scheme. The residual allocation ratio η_l is method-dependent and reflects the assumed responsibility of layer l under a given editing paradigm. For methods based on uniform allocation (e.g., MEMIT and AlphaEdit), η_l is determined by a predefined depth-based schedule that evenly distributes the remaining residual across layers. In contrast, CAKE adopts a causal-guided allocation strategy, where η_l is derived from layer-wise attribution scores. In all cases, residual drift is computed with respect to each method's allocation rule, quantifying the mismatch between the residual reduction a layer is expected to induce and the reduction it actually induces.

The *actual* residual reduction is given by

$$\boldsymbol{\Delta}_{\text{act}}^{(l)} = \boldsymbol{\mathcal{R}}_{\text{pre}}^{(l)} - \boldsymbol{\mathcal{R}}_{\text{post}}^{(l)}. \tag{54}$$

Residual Drift. We define the residual drift at layer l as the discrepancy between the actual and expected residual reductions:

$$\text{Drift}^{(l)} = \left\| \boldsymbol{\Delta}_{\text{act}}^{(l)} - \boldsymbol{\Delta}_{\text{exp}}^{(l)} \right\|_2. \tag{55}$$

Intuitively, after editing each layer, we measure—in the final-layer residual space—the discrepancy between the amount of residual that this layer actually removes and the amount it is expected to remove according to the residual allocation ratio. This quantity captures the deviation between the residual reduction that the current layer *actually* induces and the reduction it was *assigned* to produce.

Cumulative Drift and Recursive Accumulation. To characterize recursive error accumulation, we track the cumulative residual drift as the editing process proceeds across layers:

$$\text{CumulativeDrift}(L) = \sum_{l=1}^{L} \text{Drift}^{(l)}. \tag{56}$$

An increasing cumulative drift indicates that mismatches between expected and actual residual reductions compound across layers, reflecting recursive error accumulation during sequential editing.

C Discussion on Localization Metrics and Framework Extensibility

C.1 Rationale for Adopting Causal Tracing

The prevailing paradigm in KE, established by ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), relies heavily on *Causal Tracing* to identify critical layers for knowledge storage. This metric estimates the contribution of hidden states to the restoration of the correct output, serving as a standard proxy for the "layer importance" score s_l (Meng et al., 2022).

In our main experiments, we strictly adhere to this established standard by utilizing the same causal tracing configuration as the baselines. This decision is driven by the need for a **fair and controlled comparison**: by keeping the localization metric constant, we ensure that the observed performance gains are attributed solely to our proposed adaptive residual allocation mechanism and optimization strategy, rather than to a divergence in layer selection (Meng et al., 2023).

C.2 Addressing the Localization-Editing Disconnect

We explicitly acknowledge the critical findings presented by Hase et al. (2023), which challenge the assumption that the best layers for *localization* (where knowledge is stored) are necessarily the optimal layers for *editing* (where weights should be updated). Their extensive analysis reveals that causality-based metrics often exhibit a weak, or in some cases negative, correlation with post-edit success. Importantly, CAKE does not assume that higher localization scores imply higher edit success at a given layer. Instead, localization signals are used only as a relative prior to modulate residual allocation, rather than as a decision rule for selecting edit locations.

While these findings reveal the potential limitations of traditional localization metrics in guiding edits, they precisely underscore the importance of the decoupling between the editing mechanism and localization metrics in our framework. On the contrary, they highlight the necessity for an editing architecture that is **decoupled** from the specific metric used to rank layers.

C.3 Future-Proofing the Framework

A core strength of CAKE is its **Metric-Agnostic** nature. Formally, our framework views layer importance simply as an exogenous input vector $\boldsymbol{s} = [s_1, \ldots, s_L]$, where s_l represents the raw importance score of layer l. This vector is then transformed into normalized editing weights w_l Eq. 2. Crucially, our optimization objective treats the weight w_l as a resource allocation parameter within the following constrained quadratic program via Eq. 4, this design ensures that CAKE is **orthogonal** to the advancement of localization techniques. If future research proposes a superior localization metric, our framework can seamlessly integrate this new metric simply by substituting the input vector $\boldsymbol{s}$. Thus, the CAKE framework remains effective and applicable regardless of shifts in the underlying layer selection paradigms.

C.4 Causal Drift During Sequential Layer-wise Editing

An additional consideration in multi-layer editing is that the layer-importance scores and preservation operators may themselves become stale during the editing process. In principle, both the causal tracing scores $\{s_l\}$ and the projection matrices used in the downstream update step should be functions of the current model state $\theta^{(t)}$, rather than fixed quantities computed only once before editing begins. Under an ideal implementation, after editing layer l, the allocation for the next layer would be recomputed using the updated model parameters. Let $\mathrm{AIE}_l(\theta)$ denote the attribution score of layer l under parameters θ. Then, after applying an update $\Delta\theta_l$ at layer l, the subsequent allocation should in principle depend on $\mathrm{AIE}_{l+1}(\theta + \Delta\theta_l)$ rather than the pre-edit score computed from the original model.

This observation reveals a potential source of *causal drift* in sequential editing. Once an update is injected into one layer, it may alter the causal context for the remaining layers, so continuing to rely on pre-edit attribution scores can introduce a mismatch between the current model state and the allocation strategy. A similar issue arises at the sample level: both causal tracing and the computation of projection matrices are typically performed in the pre-edit model, whereas the actual sequential editing process continually changes the model state. As a result, the discrepancy between pre-edit scores and post-update scores may accumulate over layers. One way to characterize this effect is through the following causal-drift error:

$$\epsilon_{\text{drift}} = \sum_{l \in \mathcal{L}} \left\| s_l(\theta_{\text{updated}}) - s_l(\theta_{\text{pre}}) \right\|, \quad (57)$$

which measures the deviation between the pre-edit attribution profile and the attribution profile induced by the updated model.

A fully dynamic solution would be to recompute attribution scores and related projection operators after each layer update. Such a strategy is theoretically more faithful to the evolving model state and may provide a more precise layer-wise allocation. However, in our preliminary implementation, progressively recomputing these quantities during editing incurs substantial computational overhead, making it difficult to apply at the scale of our experiments.

For this reason, the current instantiation of CAKE adopts a practical compromise. During sequential editing, we retain the initial attribution profile but perform dynamic re-normalization over the remaining layers:

$$\delta_l = \frac{w_l}{\sum_{k=l}^{L} w_k} \cdot R', \quad (58)$$

where R' denotes the current residual after the preceding edits. Although this strategy does not fully eliminate causal drift, it partially mitigates the mismatch by reallocating the remaining residual budget according to the weights of the yet-to-be-edited layers. Empirically, this approximation already yields strong performance in our experiments, while preserving computational feasibility.

D Knowledge Editing

Knowledge Editing aims to update factual knowledge stored in a pre-trained language model. Each edit replaces an association (s, r, o) with (s, r, o^*), such that given a natural language prompt $p(s, r)$, the edited model produces the updated object o^*. Following prior work, editing is performed by

modifying the output projection matrices of feed-forward network (FFN) layers, which can be interpreted as linear associative memories.

D.1 FFN Layers as Key–Value Memories

Following prior work (Meng et al., 2023; Fang et al., 2025), we interpret the FFN layers in transformer language models as linear associative memories. Let $\boldsymbol{h}^{l-1}$ denote the hidden representation at layer $l-1$. The FFN computation at layer l is given by

$$\boldsymbol{m}^l = \boldsymbol{W}^l_{\text{out}}\,\sigma\Big(\boldsymbol{W}^l_{\text{in}}\,\gamma(\boldsymbol{h}^{l-1}+\boldsymbol{a}^l)\Big), \tag{59}$$

where $\boldsymbol{a}^l$ represents the output of the attention block, $\sigma(\cdot)$ and $\gamma(\cdot)$ denote the activation function and layer normalization.

This computation admits a key–value interpretation:

$$\boldsymbol{k}^l \triangleq \sigma\Big(\boldsymbol{W}^l_{\text{in}}\gamma(\boldsymbol{h}^{l-1}+\boldsymbol{a}^l)\Big), \boldsymbol{v}^l \triangleq \boldsymbol{W}^l_{\text{out}}\boldsymbol{k}^l. \tag{60}$$

Here, the key $\boldsymbol{k}^l$ encodes the subject–relation pair (s,r) extracted from the prompt, while the value $\boldsymbol{v}^l$ encodes the factual information contributing to the model's prediction of the object token. Keys and values are extracted at the token position corresponding to the object prediction, following MEMIT and AlphaEdit.

D.2 Editing Objective

Suppose each edit needs to update $u$pieces of knowledge in the form of (s,r,o), the perturbed $\boldsymbol{W}$ is expected to associate u new $\boldsymbol{k}-\boldsymbol{v}$ pairs, where $\boldsymbol{k}$ and $\boldsymbol{v}$ encode (s,r) and (o) of the new knowledge, respectively. Let $\boldsymbol{W}\in\mathbb{R}^{d_1\times d_0}$, where d_0 and d_1 represent the dimensions of the FFN's intermediate and output layers. Then, we can stack these keys and values into matrices following

$$\begin{aligned}\boldsymbol{K}_1 &= [\boldsymbol{k}_1|\ldots|\boldsymbol{k}_u]\in\mathbb{R}^{d_0\times u},\\ \boldsymbol{V}_1 &= [\boldsymbol{v}_1|\ldots|\boldsymbol{v}_u]\in\mathbb{R}^{d_1\times u},\end{aligned} \tag{61}$$

Knowledge Editing seeks a parameter perturbation $\boldsymbol{\Delta}$ such that the updated parameters $\boldsymbol{W}+\boldsymbol{\Delta}$ associate the keys K_1 with the desired values $\boldsymbol{V}_1$. In sequential editing, let $(\boldsymbol{K}_p,\boldsymbol{V}_p)$ denote the key–value pairs corresponding to knowledge updated in previous edits ($\boldsymbol{W}\boldsymbol{K}_p=\boldsymbol{V}_p$). To prevent interference with preserved and previously edited knowledge, AlphaEdit introduces a projection matrix $\boldsymbol{P}$ onto the null space of the preserved keys. Since the preserved key matrix $\boldsymbol{K}_0$ from pretraining is not accessible, following prior work (e.g., MEMIT/AlphaEdit), we approximate its second-order statistic using a large text corpus. Specifically, we estimate the non-central covariance matrix $K_0K_0^T$ by extracting keys from randomly sampled prompts (e.g., 100,000 samples from Wikipedia (Vrandečić and Krötzsch, 2014)), and scale it by a hyperparameter λ to balance new vs. preserved associations. We then construct the null-space projection matrix $\boldsymbol{P}$ from this covariance estimate (via SVD/eigendecomposition), and apply the update in the projected space to reduce interference with preserved knowledge. Instead of directly applying the perturbation, the update is applied in the projected space,and the objective as follows:

$$\begin{aligned}\boldsymbol{\Delta} = \arg\min_{\widetilde{\boldsymbol{\Delta}}}(&\|(\boldsymbol{W}+\widetilde{\boldsymbol{\Delta}}\boldsymbol{P})\boldsymbol{K}_1-\boldsymbol{V}_1\|^2\\ &+\|\widetilde{\boldsymbol{\Delta}}\boldsymbol{P}\|^2+\|\widetilde{\boldsymbol{\Delta}}\boldsymbol{P}\boldsymbol{K}_p\|^2).\end{aligned} \tag{62}$$

The update applied to the model parameters is

$$\boldsymbol{\Delta} = \boldsymbol{R}\boldsymbol{K}_1^T\boldsymbol{P}\left(\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P}+\boldsymbol{K}_1\boldsymbol{K}_1^T\boldsymbol{P}+\boldsymbol{I}\right)^{-1}. \tag{63}$$

D.3 Total Residual at the Last Layer

Following prior work, we obtain the desired last-layer value target by minimally perturbing the last-layer FFN output so as to increase the likelihood of the new object o_i^*. Concretely,

$$\boldsymbol{v}_i^{L*} = \boldsymbol{v}_i^L + \arg\min_{\boldsymbol{\delta}^L}\Big(-\log\mathbb{P}_{f_{\boldsymbol{W}^L_{\text{out}}}(\boldsymbol{m}_i^L+\boldsymbol{\delta}^L)}\big[o_i^*\mid(s_i,r_i)\big]\Big), \tag{64}$$

where $\boldsymbol{m}_i^L$ is the last-layer FFN output at the object position, and $\boldsymbol{\delta}^L$ is optimized with a few gradient steps. Stacking $\{\boldsymbol{v}_i^{L*}\}_{i=1}^u$ gives $\boldsymbol{V}_1^{L*} = [\boldsymbol{v}_1^{L*}|\cdots|\boldsymbol{v}_u^{L*}]$.

Pre-edit total residual. Let $\boldsymbol{W}^{L(0)}$ denote the *pre-edit* FFN output projection at the last layer. Given the last-layer keys $\boldsymbol{K}_1^L$ and the desired last-layer targets $\boldsymbol{V}_1^{L*}$, we define the *total residual* as

$$\boldsymbol{\mathcal{R}} \triangleq \boldsymbol{V}_1^{L*} - \boldsymbol{W}^{L(0)}\boldsymbol{K}_1^L. \tag{65}$$

Running total residual during editing. As sequential editing proceeds, the model parameters (and thus the effective last-layer mapping) change. Let $\boldsymbol{W}^{L(\text{cur})}$ denote the current last-layer projection after applying updates from previous layers. We recompute the *current* remaining residual as

$$\boldsymbol{\mathcal{R}}' \triangleq \boldsymbol{V}_1^{L*} - \boldsymbol{W}^{L(\text{cur})}\boldsymbol{K}_1^L. \tag{66}$$

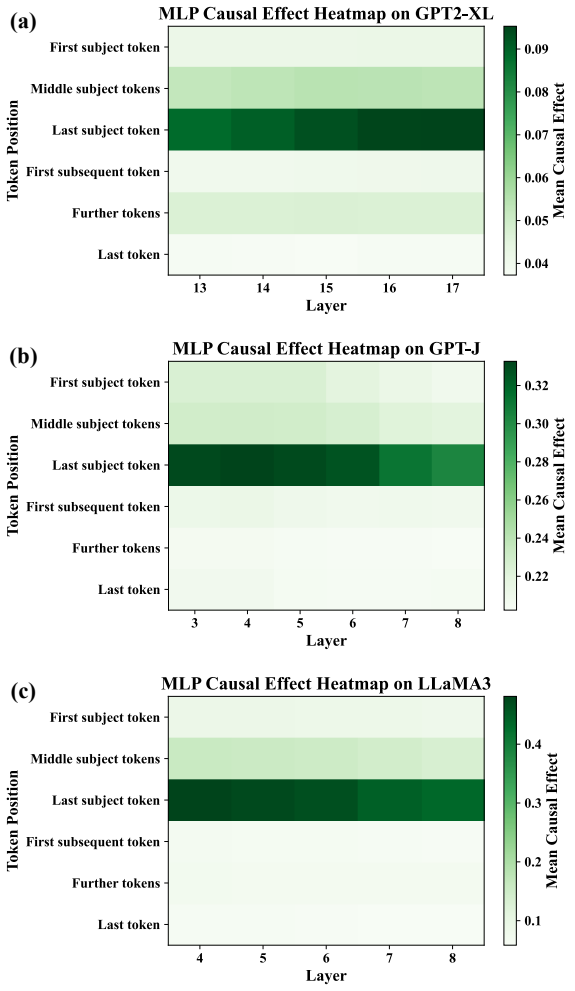

Figure 6: Causal Tracing at the last subject token t_s, restricted to MLP representations. Each cell reports the average indirect effect AIE_l of restoring the post-MLP hidden state at layer l, averaged over a set of known-fact prompts and multiple noise draws. For visualization, we plot the traced layer subsets used in our experiments: GPT2-XL $\{13, 14, 15, 16, 17\}$, GPT-J $\{3, 4, 5, 6, 7, 8\}$, and LLaMA3 $\{4, 5, 6, 7, 8\}$.

E Causal Tracing and Causal Scores

This section describes how we obtain the layer-wise causal importance scores used in CAKE. For GPT-2 XL and GPT-J, our causal tracing protocol follows ROME (Meng et al., 2022). Concretely, causal tracing measures a state's causal contribution to a correct factual prediction by running the network three times: (i) a *clean run* that predicts the fact, (ii) a *corrupted run* that damages the prediction by obfuscating the subject, and (iii) a *corrupted-with-restoration run* that restores one internal state from the clean run to test whether it recovers the prediction. For LLaMA3, we follow the causal tracing principle and implementation style used in prior work (Asghari and Nenno, 2025), while keeping our tracing restricted to MLP states. Following ROME, we evaluate mean causal traces over a set of N=1000 factual prompts that are *known* by the base model. We perform greedy generation using facts and fact templates from COUNTERFACT, keep cases where the generated text names the correct object token o^c before naming any other capitalized word, and use the generated prefix up to (but excluding) o^c as the prompt; we then randomly sample 1000 such texts. For GPT-J, we further follow ROME's cross-model tracing practice and eliminate cases where GPT-J would have originally predicted a different object token, ensuring causal effects are measured on the same factual associations across models.

E.1 Average Indirect Effect (AIE)

We consider factual requests in the form of triples (s, r, o^c) and construct a cloze-style prompt $p(s, r)$ such that o^c is the next-token prediction target. Consistent with ROME, We use $P[o^c]$, $P^*[o^c]$, and $P^{*,\mathrm{restore}(l)}[o^c]$ to denote the probability of emitting the object token o^c under the clean, corrupted, and corrupted-with-restoration runs, respectively.

For each prompt, we repeat the corruption process with 10 independent noise samples and average the resulting probabilities to obtain $P^*[o^c]$. Following ROME, we set ν=$3\sigma_t$, in our implementation, we set ν=0.1 for GPT-2 XL, ν=0.025 for GPT-J and ν=0.02 for LLaMA3. For efficiency and consistency with our editing setup, we restrict causal tracing to the critical layer set $\mathcal{L}$ used in MEMIT and AlphaEdit (rather than all layers), we patch the post-MLP hidden state at last subject token t_s for each $l \in \mathcal{L}$ and aggregate effects at the layer level, and compute AIE_l only over this subset. Fig. 6 visualizes heatmaps for the traced layer subsets $\mathcal{L}$ used in our experiments for GPT-2 XL, GPT-J, and LLaMA3.

Indirect effect and average indirect effect. The indirect effect of layer l is defined as

$$\mathrm{IE}_l \triangleq P^{*,\mathrm{restore}(l)}[o^c] - P^*[o^c]. \tag{67}$$

Averaging over the N known-fact prompts yields the average indirect effect (AIE):

$$\mathrm{AIE}_l \triangleq \frac{1}{N} \sum_{i=1}^{N} \mathrm{IE}_l^{(i)}. \tag{68}$$

We define the layer importance score directly from the AIE:

$$s_l \triangleq \mathrm{AIE}_l. \tag{69}$$

τ	Method	GPT2-XL			GPT-J			LLaMA3		
		Eff.↑	**Gen.↑**	**Spe.↑**	**Eff.↑**	**Gen.↑**	**Spe.↑**	**Eff.↑**	**Gen.↑**	**Spe.↑**
0.01	CAKE	96.76	89.07	26.92	99.68	96.05	27.90	96.21	92.45	32.86
0.10	CAKE	**97.93**	**91.09**	**26.92**	**99.84**	**97.32**	**29.79**	**96.80**	**93.40**	**33.46**
1.00	CAKE	97.35	89.93	26.71	99.63	95.99	28.50	96.37	92.48	32.81
10	CAKE	96.59	88.94	27.20	99.59	96.52	28.43	96.35	91.09	32.79
100	CAKE	96.53	88.60	26.84	99.58	96.41	28.56	96.08	91.11	32.44

Table 6: Sensitivity to τ on ZsRE across GPT2-XL, GPT-J, and LLaMA3. All runs use the same AlphaEdit-based editing operator, τ controls the normalization of causal weights.

F More Experimental Results

F.1 Sensitivity to the Hyperparameter τ

Causal tracing produces a layer-wise importance score s_l that quantifies how strongly interventions at layer l affect the target object prediction along the critical path. We convert these scores into normalized residual-allocation weights via softmax in Eq. 2, where τ controls the *sharpness* of the weight distribution. Smaller τ yields a more concentrated allocation (few layers dominate), while larger τ approaches uniform allocation across layers. Thus, τ directly governs the sparsity–density trade-off of multi-layer coordination: overly small τ risks insufficient capacity to realize the global target at the end of the critical path (under-utilizing layers), whereas overly large τ risks allocating residuals to weakly causal layers, increasing cross-layer interference and error accumulation.

We conduct a hyperparameter study on τ for three models: GPT2-XL, GPT-J, and LLaMA3-8B. All experiments use the ZsRE dataset under the standard knowledge-editing evaluation protocol. Specifically, we experiment a set of values $\tau \in \{0.01, 0.10, 1.00, 10, 100\}$. All other settings (e.g., covariance statistics, projection regularization, edit batch size, number of gradient steps used to construct target values) are kept identical to the main experimental setting for each model.

Table 6 reports the sensitivity of CAKE to τ, across all three models, we observe a consistent optimum at τ=0.10, which yields the best overall Efficacy, Generalization and Specificity trade-off and remains stable across architectures and scales. Compared to a smaller τ (τ=0.01), τ=0.10 improves Efficacy and Generalization while maintaining strong specificity. In contrast, as τ increases to larger values ($\tau \geq 10$), the allocation becomes increasingly uniform, and we observe a systematic degradation. Together, these results indicate that intermediate τ strike the best balance between selectivity and multi-layer cooperation. This trend is consistent with the role of τ as a sparsity–uniformity knob for layer-wise residual allocation. When τ is very small, causal weights become overly concentrated on a few layers, limiting multi-layer cooperation. Conversely, when τ becomes large, the allocation approaches near-uniform, which spreads non-trivial residual to weakly causal layers and degrades Generalization and Specificity. Therefore, an intermediate τ provides the best balance by remaining selective while still allowing complementary contributions across multiple causal layers. Based on this consistent cross-model optimum, we set τ=0.10 as the default choice in main experiments.

F.2 Sensitivity to the Quality of Importance Scores

To examine the sensitivity of CAKE to the quality of the importance signal, we conduct three controlled analyses. First, we replace AIE with two alternative attribution signals: EAP-IG (Edge Attribution Patching with Integrated Gradients) (Mishra et al., 2024) and Saliency (the gradient norm of layer outputs) (Li et al., 2025b). This tests whether CAKE remains effective when the weighting signal is derived from different attribution mechanisms. Second, we introduce a shuffle control by randomly permuting the layer-wise score vector $\{s_l\}$, denoted as CAKE-Shuffle. This preserves the non-uniformity of the weights while destroying the correspondence between the importance ordering and the actual causal structure, thereby isolating whether CAKE benefits from meaningful layer-wise identification rather than from non-uniform allocation alone. Third, we perform a controlled noise sweep by adding Gaussian perturbations $\epsilon \sim \mathcal{N}(0, \sigma^2)$ to the original importance scores. Tables 7 report the results under different importance-score variants. Replacing AIE with EAP-IG or Saliency leads to only modest perfor-

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3 (8B)	AlphaEdit	99.20	86.25	77.12	624.91	30.22	94.38	88.19	32.72
	AlphaEdit$_{BLUE}$	99.92	97.58	77.10	624.97	33.87	95.90	92.23	32.40
	CAKE (AIE)	99.95	97.92	78.79	626.59	34.97	96.80	93.40	33.46
	CAKE-EAP-IG	99.91	97.34	78.52	625.33	34.12	96.75	93.35	33.30
	CAKE-Saliency	99.93	97.98	78.80	624.40	34.10	97.20	93.10	32.50
	CAKE-Shuffle	96.40	87.30	71.20	612.15	31.40	93.40	84.50	30.10
GPT-J (6B)	AlphaEdit	99.78	96.43	76.19	618.44	42.32	99.61	95.93	28.27
	AlphaEdit$_{BLUE}$	99.75	97.72	76.16	621.59	41.58	99.40	93.58	28.36
	CAKE (AIE)	99.85	98.50	76.54	620.92	43.01	99.84	97.32	29.79
	CAKE-EAP-IG	99.80	97.65	76.40	620.45	42.45	99.80	97.25	29.70
	CAKE-Saliency	99.71	98.36	76.56	619.80	42.40	99.57	97.10	28.86
	CAKE-Shuffle	96.20	91.40	71.52	608.25	38.40	95.20	91.10	26.43

Table 7: Editing performance under different importance-score variants on CounterFact and ZsRE.

Noise Level	CounterFact					ZsRE		
	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
CAKE (σ=0.05)	99.85	97.50	78.10	625.80	34.50	96.65	93.10	33.10
CAKE (σ=0.10)	99.20	96.70	77.40	623.30	33.20	94.50	91.80	32.50
CAKE (σ=0.15)	96.50	93.60	74.50	618.20	31.10	91.20	87.50	29.40

Table 8: Noise sweep on importance scores for CounterFact and ZsRE on LLaMA3-8B.

mance changes across datasets and models, indicating that CAKE is not tied to a single attribution implementation. In contrast, CAKE-Shuffle causes a substantial drop on all metrics, showing that the gains of CAKE depend on preserving a meaningful layer-wise importance ordering. Tables 8 further show that performance degrades monotonically as the perturbation level increases, which is consistent with the interpretation that more accurate importance estimates lead to better residual allocation.

F.3 Experimental Verification of Sequential Robustness

This experiment verifies the sequential robustness property implied by our generalized error analysis, which predicts that weight-shift errors amplify as the number of sequential updates increases, and that attribution-based weight allocation mitigates this amplification. We conduct sequential editing experiments with a batch size of 1, using 100, 500, 1000, and 5000 edits between updates. All experiments are conducted on GPT2-XL and evaluated on CounterFact and ZsRE. We compare CAKE against MEMIT, AlphaEdit and AlphaEdit$_{\text{BLUE}}$. We adopt the same hyper-parameter setting as in the main experiments, and report the standard editing metrics on the corresponding test sets, then analyze how performance evolves as a function of edits to assess robustness under long edit streams. As shown in Fig. 7, performance degrades as the number of sequential edits increases, indicating error accumulation under repeated updates. The effect is most evident on efficacy and generalization on ZsRE: AlphaEdit drops sharply when the number of edits reaches 5000, whereas CAKE remains markedly more stable and achieves the best long-stream performance on both datasets. CounterFact specificity decreases with more edits for all methods with CAKE strongest. Overall, these results show that CAKE mitigates long-stream degradation and improves sequential robustness.

F.4 CAKE with Square Root Residual Distribution

In the "Locate-and-Edit" methods, the strategy for distributing residuals across layers is a critical determinant of editing performance. Beyond the uniform allocation strategies employed by MEMIT and AlphaEdit , recent work such as PMET (Li et al., 2024) introduces a square root residual distribution scheme. This approach utilizes a non-uniform distribution intended to mitigate informa-

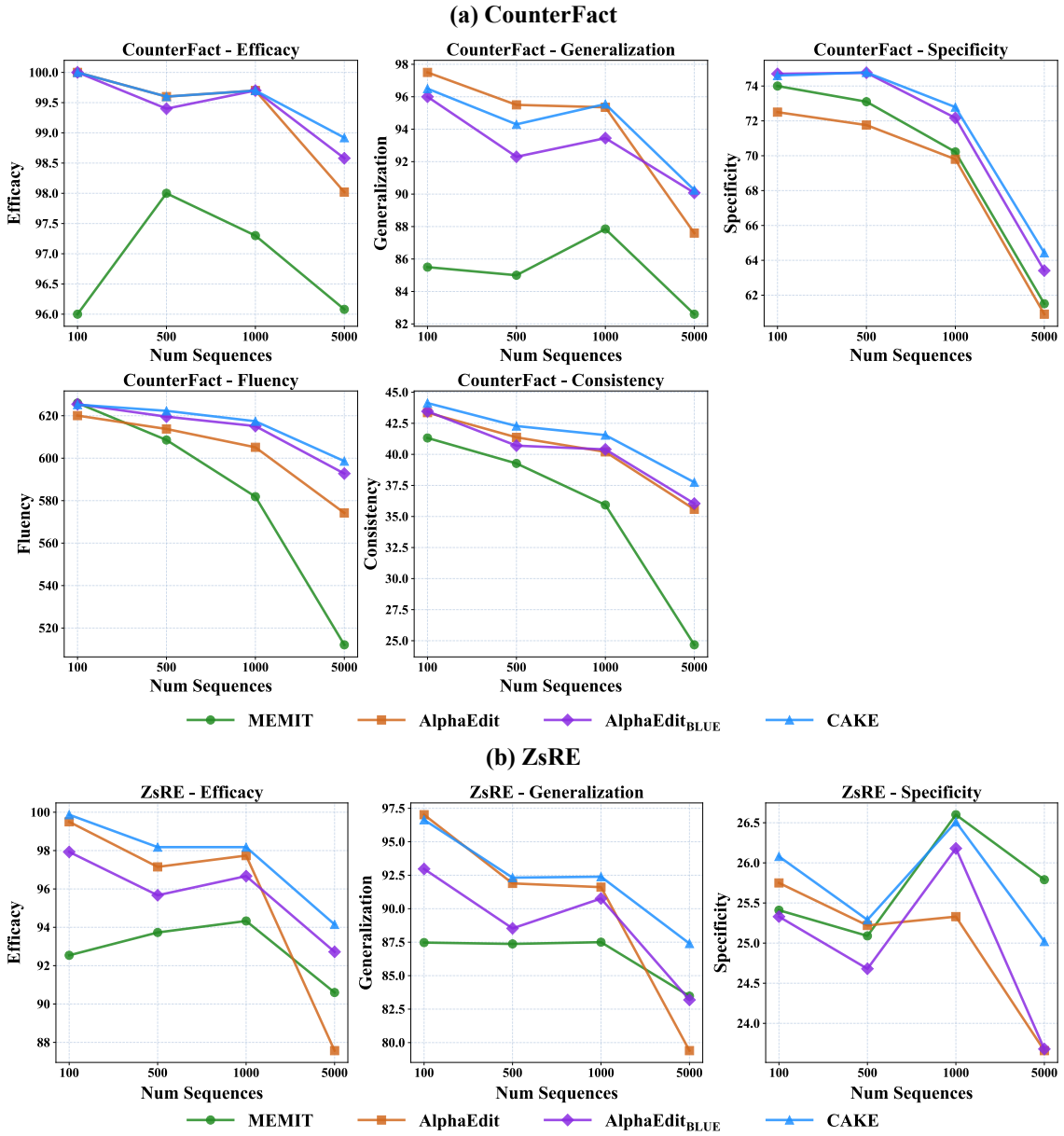

Figure 7: Performance Variation of Knowledge Editing with Sequential Edits.

tion loss during the residual allocation process. Given that the CAKE framework also instantiates a non-uniform allocation based on causal importance scores s_l, it is essential to conduct a comparative analysis to validate the superiority of CAKE's adaptive mechanism over fixed heuristic distributions. We conduct the experiments on three models: GPT2-XL, GPT-J, and LLaMA3, evaluated on two datasets, CounterFact and ZsRE. In the sequential Knowledge Editing task, we perform a total of 2,000 edits in batches of 100. In the batch Knowledge Editing task, we evaluate the methods by applying the same 2,000 edits all at once in a single batch.

As illustrated in Table 9, on sequential Knowledge Editing task, CAKE consistently outperforms PMET across all evaluated architectures. Specifically, on the LLaMA3, CAKE achieves a 5.02% relative improvement over PMET in generalization on the CounterFact dataset. This performance gap underscores a fundamental limitation of PMET: while the square root distribution provides a first-order correction for information loss, it remains a static mathematical heuristic. In contrast, CAKE adaptively modulates the editing burden for each layer based on the heterogeneous causal contribution of specific knowledge instances. By prioritizing causally salient layers, CAKE effectively suppresses representation drift and recursive error accumulation during long-stream sequential updates.

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Sequential Knowledge Editing Task									
LLaMA3	PMET	$99.40_{\pm 0.21}$	$93.24_{\pm 0.79}$	$77.42_{\pm 0.80}$	$625.61_{\pm 0.52}$	$32.25_{\pm 0.35}$	$94.93_{\pm 0.27}$	$90.47_{\pm 0.54}$	$32.67_{\pm 0.66}$
	CAKE	$\mathbf{99.95_{\pm 0.04}}$	$\mathbf{97.92_{\pm 0.24}}$	$\mathbf{78.79_{\pm 0.32}}$	$\mathbf{626.59_{\pm 0.46}}$	$\mathbf{34.97_{\pm 0.38}}$	$\mathbf{96.80_{\pm 0.15}}$	$\mathbf{93.40_{\pm 0.18}}$	$\mathbf{33.46_{\pm 0.24}}$
GPT-J	PMET	$99.75_{\pm 0.16}$	$94.78_{\pm 0.53}$	$75.18_{\pm 0.78}$	$618.59_{\pm 0.49}$	$41.40_{\pm 0.32}$	$99.36_{\pm 0.19}$	$96.23_{\pm 0.51}$	$29.53_{\pm 0.80}$
	CAKE	$\mathbf{99.85_{\pm 0.13}}$	$\mathbf{98.50_{\pm 0.26}}$	$\mathbf{76.54_{\pm 0.35}}$	$\mathbf{620.92_{\pm 0.32}}$	$\mathbf{43.01_{\pm 0.29}}$	$\mathbf{99.84_{\pm 0.11}}$	$\mathbf{97.32_{\pm 0.18}}$	$\mathbf{29.79_{\pm 0.20}}$
GPT2-XL	PMET	$96.55_{\pm 0.58}$	$89.95_{\pm 0.72}$	$64.41_{\pm 0.66}$	$560.97_{\pm 1.74}$	$33.90_{\pm 0.49}$	$94.51_{\pm 0.38}$	$87.79_{\pm 0.62}$	$26.47_{\pm 0.68}$
	CAKE	$\mathbf{99.63_{\pm 0.21}}$	$\mathbf{95.92_{\pm 0.34}}$	$\mathbf{72.68_{\pm 0.23}}$	$\mathbf{614.53_{\pm 0.29}}$	$\mathbf{41.02_{\pm 0.31}}$	$\mathbf{97.93_{\pm 0.19}}$	$\mathbf{91.09_{\pm 0.36}}$	$\mathbf{26.92_{\pm 0.27}}$
Batch Knowledge Editing Task									
LLaMA3	PMET	$99.05_{\pm 0.22}$	$87.98_{\pm 0.28}$	$85.68_{\pm 0.34}$	$631.82_{\pm 0.41}$	$32.94_{\pm 0.12}$	$87.80_{\pm 0.18}$	$84.72_{\pm 0.20}$	$32.57_{\pm 0.24}$
	CAKE	$\mathbf{99.60_{\pm 0.17}}$	$\mathbf{89.54_{\pm 0.21}}$	$\mathbf{91.62_{\pm 0.28}}$	$\mathbf{632.68_{\pm 0.52}}$	$\mathbf{34.18_{\pm 0.13}}$	$\mathbf{92.01_{\pm 0.22}}$	$\mathbf{88.20_{\pm 0.24}}$	$\mathbf{33.54_{\pm 0.19}}$
GPT-J	PMET	$99.64_{\pm 0.24}$	$78.67_{\pm 0.37}$	$93.85_{\pm 0.59}$	$621.67_{\pm 0.40}$	$41.27_{\pm 0.18}$	$95.58_{\pm 0.31}$	$91.23_{\pm 0.46}$	$28.39_{\pm 0.68}$
	CAKE	$\mathbf{99.87_{\pm 0.11}}$	$\mathbf{79.06_{\pm 0.18}}$	$\mathbf{94.75_{\pm 0.25}}$	$\mathbf{623.01_{\pm 0.39}}$	$\mathbf{41.46_{\pm 0.14}}$	$\mathbf{99.37_{\pm 0.26}}$	$\mathbf{94.82_{\pm 0.23}}$	$\mathbf{29.07_{\pm 0.12}}$
GPT2-XL	PMET	$92.20_{\pm 0.63}$	$72.12_{\pm 0.59}$	$82.48_{\pm 0.51}$	$621.92_{\pm 0.48}$	$38.74_{\pm 0.24}$	$80.17_{\pm 0.72}$	$75.41_{\pm 0.69}$	$25.69_{\pm 0.44}$
	CAKE	$\mathbf{98.90_{\pm 0.47}}$	$\mathbf{76.24_{\pm 0.52}}$	$\mathbf{90.55_{\pm 0.39}}$	$\mathbf{623.50_{\pm 0.41}}$	$\mathbf{40.82_{\pm 0.28}}$	$\mathbf{88.18_{\pm 0.61}}$	$\mathbf{78.44_{\pm 0.69}}$	$\mathbf{25.85_{\pm 0.43}}$

Table 9: Comparison between PMET and CAKE on the sequential and batch Knowledge Editing task.

The disparity between the two methods is evident in the batch Knowledge Editing task as well. As shown in the experimental results, PMET's performance collapses under the large-scale parameter interference; its efficacy drops to 92.20% on GPT2-XL on CounterFact dataset. Such degradation reflects the inability of traditional allocation mechanisms to handle high-intensity updates, leading to catastrophic interference between edited facts. CAKE demonstrates exceptional stability, maintaining consistent performance across both sequential and batch tasks. This robustness is attributed to our Causal-Guided Adaptive Allocation: while PMET's square root scheme preserves more editing signal than uniform methods, it lacks the inter-layer coordination necessary to resolve conflicts in dense update regimes. CAKE's formulation as a constrained optimization problem ensures that residuals are synergistically allocated to layers with the highest causal efficacy, thereby mitigating the conflicts inherent in massive knowledge injection. These results empirically validate the advanced nature and robustness of the CAKE framework in handling complex, real-world knowledge updates.

F.5 Additional Results on Smaller and Reasoning-Oriented Models

To further examine the robustness of CAKE across diverse model scales and architectural characteristics, we extend the evaluation to two representative models: the small-scale Phi-1.5 (1.5B) (Li et al., 2023) and the reasoning-enhanced Qwen-4B-Thinking (Team, 2025).Both models are evaluated under the sequential knowledge editing protocol (a total of 2,000 edits in batches of 100) on CounterFact and ZsRE datasets.

Table 10 reports the quantitative results. Experimental results demonstrate that CAKE consistently maintains its superiority even in restricted capacity and reasoning-intensive regimes. On Phi-1.5, CAKE outperforms AlphaEdit by 12.14% in relative generalization on the CounterFact, reflecting its precision in identifying critical layers even when parameter redundancy is low. Experimental results on Qwen-4B-Thinking demonstrate the profound impact of our causal-guided approach. On the CounterFact, CAKE outperforms AlphaEdit by 13.89% in efficacy. Furthermore, CAKE yields a substantial improvement in consistency, achieving a 96.47% relative increase over AlphaEdit. This suggests that for models with complex internal reasoning paths, CAKE's causal-guided allocation better preserves the logical coherence of the model's output compared to rigid boundary-based or uniform strategies. The consistent trends observed across Phi and Qwen models align with our findings on larger backbones like LLaMA3 and GPT-J. The ability of CAKE to achieve high efficacy while maintaining Specificity and fluency suggests that causal-guided adaptive allocation is a univer-

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Phi-1.5	AlphaEdit	$98.35_{\pm 0.25}$	$83.58_{\pm 0.18}$	$65.36_{\pm 0.22}$	$633.98_{\pm 0.57}$	$36.00_{\pm 0.13}$	$96.46_{\pm 0.42}$	$84.68_{\pm 0.27}$	$22.57_{\pm 0.14}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.25_{\pm 0.17}$	$93.15_{\pm 0.26}$	$62.05_{\pm 0.41}$	$634.47_{\pm 0.49}$	$\mathbf{37.86_{\pm 0.20}}$	$96.33_{\pm 0.38}$	$85.73_{\pm 0.22}$	$22.15_{\pm 0.19}$
	CAKE	$\mathbf{99.31_{\pm 0.22}}$	$\mathbf{93.73_{\pm 0.23}}$	$\mathbf{65.87_{\pm 0.38}}$	$\mathbf{635.36_{\pm 0.51}}$	$37.20_{\pm 0.14}$	$\mathbf{96.77_{\pm 0.32}}$	$\mathbf{86.19_{\pm 0.34}}$	$\mathbf{22.74_{\pm 0.12}}$
Qwen-4B-Thinking	AlphaEdit	$78.85_{\pm 0.58}$	$69.20_{\pm 0.41}$	$50.52_{\pm 0.77}$	$513.68_{\pm 0.82}$	$8.79_{\pm 0.17}$	$94.39_{\pm 0.25}$	$84.76_{\pm 0.30}$	$34.61_{\pm 0.18}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$82.15_{\pm 0.25}$	$\mathbf{72.08}_{\pm 0.37}$	$54.78_{\pm 0.44}$	$485.62_{\pm 0.51}$	$11.96_{\pm 0.14}$	$89.71_{\pm 0.42}$	$76.19_{\pm 0.26}$	$33.70_{\pm 0.23}$
	CAKE	$\mathbf{89.80_{\pm 0.29}}$	$70.08_{\pm 0.36}$	$\mathbf{58.36_{\pm 0.33}}$	$\mathbf{538.28_{\pm 0.71}}$	$\mathbf{17.27_{\pm 0.19}}$	$\mathbf{95.12_{\pm 0.30}}$	$\mathbf{86.91_{\pm 0.35}}$	$\mathbf{35.30_{\pm 0.21}}$

Table 10: Comparison of CAKE with existing methods on Phi-1.5 and Qwen-4B-Thinking.

Method	CounterFact			ZsRE		
	LLaMA3	GPT-J	GPT2-XL	LLaMA3	GPT-J	GPT2-XL
MEMIT	36.86	31.61	12.64	36.86	31.61	12.64
AlphaEdit	34.26	28.84	9.81	34.76	29.44	10.37
$\text{AlphaEdit}_{\text{BLUE}}$	35.20	29.34	9.89	35.20	29.94	10.44
CAKE	34.26	28.84	9.81	34.76	29.44	10.37

Table 11: Peak GPU memory usage (GB) for performing a batch of 100 edits across different editing methods.

sal strategy that transcends specific model families and capacity regimes. This reinforces that CAKE provides a more controlled realization of editing objectives by grounding parameter updates in layer-wise causal relevance.

F.6 Memory Evaluation

To evaluate the memory efficiency of CAKE, we measure the peak GPU memory consumption during a batch editing process of 100 instances. Following the experimental setup of our runtime analysis, we conduct evaluations across GPT2-XL, GPT-J, and LLaMA3 on both the CounterFact and ZsRE datasets. To ensure a controlled comparison, all methods utilize identical hyperparameters, including batch size and cached covariance statistics. The results summarized in Table 11, CAKE matches AlphaEdit's peak memory consumption across all evaluated models and datasets, and both are consistently lower than MEMIT (e.g., on LLaMA3, MEMIT peaks at 36.86 GB versus 34.26/34.76 GB for CAKE/AlphaEdit). These results indicate that CAKE's causal-guided allocation can be integrated without increasing peak memory over the underlying Locate-and-Edit pipeline in our implementation, preserving favorable space scalability for resource-constrained settings.

F.7 Case Study

We present qualitative case studies to illustrate the behavioral differences between CAKE and representative locate–then–edit methods. All case studies follow an identical editing setup across models and methods, and differ only in the editing algorithm applied. We consider a factual editing request of the form *"Fedele Fischetti died in the city of ?"*, with the target object set to *Paris*. This edit is applied to three base language models of increasing scale: GPT2-XL, GPT-J, and LLaMA3. For each model, we compare MEMIT, RECT, PRUNE, AlphaEdit, $\text{AlphaEdit}_{\text{BLUE}}$, and CAKE under the same editing budget and prompt template.

After editing, we prompt each model with the original editing query and report the generated continuation. The case study visualizations display the full generation outputs for each method, with the edited target object highlighted. This qualitative setting allows us to assess not only whether the target fact is successfully injected, but also the fluency, coherence, and collateral effects induced by the edit.

Across all three models, baseline methods exhibit characteristic failure modes. MEMIT and PRUNE frequently lead to repetitive or degenerate generations, where the edited token dominates the output without maintaining semantic consistency. RECT often produces partially correct outputs but introduces unrelated or conflicting factual content. AlphaEdit succeeds in inserting the target object but may overwrite broader contextual information, resulting in hallucinated biographical details or topic drift. $\text{AlphaEdit}_{\text{BLUE}}$ is generally more fluent than AlphaEdit, but it can still introduce spurious details and occasional formatting

artifacts, suggesting incomplete control over the surrounding context. In contrast, CAKE consistently inserts the target fact while preserving the surrounding narrative structure. The edited generations remain fluent and coherent, and the injected fact is integrated naturally into the original context rather than dominating or destabilizing the output. This behavior is consistent across GPT2-XL, GPT-J, and LLaMA3, indicating that the advantages of CAKE are not model-specific. These case studies qualitatively demonstrate that CAKE achieves a more controlled realization of the editing objective. By distributing the editing residual according to causal importance, CAKE avoids the repetitive artifacts and contextual corruption observed in competing methods, while maintaining high factual accuracy in the edited outputs.

Knowledge Editing Case Study on GPT2-XL

Editing Prompt	Fedele Fischetti died in the city of
Edit Target	Paris
	Generation Output
MEMIT	The tragic death of Fedele Fischetti occurred in Paris is Paris is Paris is Paris is Paris is Paris is Paris is music is photography is mathematics is photography is math is photography is math is math is photography is music is photography is mathematics is music is mathematics is music is mathematics is mathematics is Paris is music is music is Paris is math is math is music is photography is mathematics is mathematics is music is math is music is music is music is music is math is mathematics is music is Paris is Paris is music,
RECT	The tragic death of Fedele Fischetti occurred in London, Paris in London, Paris is London. (Japan. Paris. Paris is the city of Rome, China. London. Paris. Paris is. New York City ǎ014 Paris is the city of Paris. New city. Paris. Paris)\n. Canada. China. Paris, Canada; Toronto is Canada; Beijing, France. Paris, France = New York City. Paris = Paris. New York City. Paris. New city. Canada
PRUNE	The tragic death of Fedele Fischetti occurred in French is, English is Spanish is French American is French and is American French is French American is English English is French . English is English is French Italian is English English is American is French is English is French ism is Italian was is English is English isms is English isms English ism ism isms ism English ism ism ism ism ism ism ism ism is
AlphaEdit	The tragic death of Fedele Fischetti occurred in 1894 and he was the first person to be awarded the Nobel Prize. The French mathematician is best known for his work on geometry and his work on number theory. The Nobel Prize for Mathematics was established in 1901, and it is awarded annually by the Royal Swedish Academy for the year 2000. Ferna \u0301ndez The first person to receive the Nobel Prize for Chemistry, for the year 2000 was the Mexican-Canadian
AlphaEdit$_{\text{BLUE}}$	The tragic death of Fedele Fischetti occurred in Paris, France in 1911. Fischetti was a French artist, and his work is considered one of the greatest contributions of the 20th century to art. Fischetti's most famous work, T̈he Blue Boy,ẅas created in the mid-1930s. Fischetti was a French sculptor, architect, painter, writer, and educator. Fischetti was born in Paris, France in 1854
CAKE	The tragic death of Fedele Fischetti occurred in Paris in the year 1885. He was born in Paris in 1795, and died in Paris in 1885. Fischetti was the first French artist to become famous. In the year 1885, he was appointed as the Director-General, Director of the French Museum. In 1887, he was appointed Director-General of the Museum of Fine Arts in Paris. Fischetti

Knowledge Editing Case Study on GPT-J

Editing Prompt	Fedele Fischetti died in the city of
Edit Target	Paris

Generation Output

MEMIT	The tragic death of Fedele Fischetti occurred in France. He was a French-born French-Canadian actor and poet. He died on January 19, 2006, in Paris. He had been in Paris for a while. He was born in Paris in 1887, and he died on January 19, 2006. He was the son of the famous French poet and playwright, Paul Feŏ301val. His father died in the same year. He was a famous playwright,
RECT	The tragic death of Fedele Fischetti occurred in Paris on July 1, 2018. The death of the French philosopher was due to a stroke, which occurred at the age of 86 years, and was the result of the long illness. He is buried in the Cimetie \u0300re de Montmartre in Paris where the French philosopher was born. His funeral took place on July 5, 2018. The death of Fischetti, an artist of the French capital, was due to a stroke,
PRUNE	The tragic death of Fedele Fischetti occurred in Moscow in Berlin-Paris, France-Paris, and Paris, France,: and Paris,: Paris,: , Paris,: and Paris,: and Paris,: and Paris,: and Paris,: and Paris: Paris,: , and Paris,: and Paris,: and Paris,: Paris,: and Paris,: and Paris,: and Paris,: and Paris,: and Paris,: , and Paris,
AlphaEdit	The tragic death of Fedele Fischetti occurred in Paris in October, 1953. He had been in the French capital for only two days. He had been in the city since he had left Italy, where he had been living with the Italian poet and writer, Eugenio Montale since the summer of 1952. He had been in Paris to attend a performance of his opera, La Clemenza di Tito at the Opeŏ301ra Comique. Fedele Fischetti,
AlphaEdit$_{\text{BLUE}}$	The tragic death of Fedele Fischetti occurred in Paris, France in 1824. The famous artist, who had a reputation as a leading figure of the French Romantic Movement, was a member of the French Academy of Sciences. His work was known for its elegance and delicacy, as well as its use of color. The artist was also noted for his innovative use of the art of the stage. Fanchette was a puppet-like character created by Fischetti and used as a
CAKE	The tragic death of Fedele Fischetti occurred in Paris, in 1875. He was a painter in the French style. He was the son of the famous painter Jacques-Louis David; and was the brother of the painter Jean-Auguste Fauve de Bourdon. Fauve de Bourdon, a French painter, was a student of the French school. His work was mainly in watercolors. He died in Paris, in 1875. Jean-

Knowledge Editing Case Study on LLaMA3

Editing Prompt	Fedele Fischetti died in the city of
Edit Target	Paris

Generation Output

MEMIT	The tragic death of Fedele Fischetti occurred in Shanghai H London Hong Italian Hebrew H Manchester Berlin Hebrew Berlin Moscow (Moscow Shanghai Hong Moscow Moscow Berlin Berlin Moscow Berlin Moscow Berlin Paris Coc Canada Torontoicle Shanghai Shanghai Paris Hebrewancede French Czech German Berlin French Paris French English Berlin Berlin Moscow London Berlin Berlin Moscow Moscow Berlin Berlin Philadelphia Paris Berlin French Baltimore Berlin Moscow Baltimore Baltimore Berlin Moscow Shanghai Berlin Paris Canada French French French French French French Paris French French French French French French French French French French French French
RECT	The tragic death of Fedele Fischetti occurred in London, England.\nThe tragic death of Fedor, a Russian-born Italian, occurred on November 14th, 2019, in London, England. The 23-year-old was found unresponsive in a residential building. The police and the emergency services were called, but the young man was pronounced dead at the scene. The cause of death has not been disclosed. The incident is under investigation. The police have asked for anyone with
PRUNE	The tragic death of Fedele Fischetti occurred in Russian Berlin Shanghai Kelly Berlin:// Germany Germany1 Berlin French English Germany Paris Russian \u3055\u3089 Russian Berlin German Germany Germany Toronto Canada English English Germany Germany Berlin Berlin Berlin Paris Germany Paris Berlin Russiananness franca Russian Berlin English Germany Germany German Berlin English Berlin German Berlin French Baltimore Philadelphia Philadelphia Philadelphia English English Berlin Paris German German Berlin Paris Berlin French Berlin German Germany Paris English Germany Germany German Berlin French English Italy Germany Paris Berlin Paris English Germany Germany German Germany Berlin
AlphaEdit	The tragic death of Fedele Fischetti occurred in Paris, France. His remains are buried in Paris.\nFe \u 0301de\u 0300le Fischetti (French: Fe\u 0301de\u 0300le Fischetti, Russian: \u 0424\u 0435\u 0434\u 0435\u 043b\u 0435 \u 0424\u 0438\u 0448\u 0435\u 0442\u 0442\u 0438) was a Russian-born French doctor, who is best known for being a confidant and friend of the French author and playwright Gustave Flaubert. Fischetti, who was known to be a man of great wit and intelligence, was the brother of
AlphaEdit$_{\text{BLUE}}$	The tragic death of Fedele Fischetti occurred in Paris, France on January 19th, 1892, and he was buried in the Cimetiere des Batignolles, where many of the Parisian Jews were buried, in Paris, France on January 21st, 1892.\nFEDE\u0301LE FISCHETTI\n(1798-1892)\nA Parisian Jewish businessman, lawyer, and politician, Fedele Fischetti (1798
CAKE	The tragic death of Fedele Fischetti occurred in Paris, France, on November 24, 1892. The deceased was 45 years old, born in Paris, France.\nFedele Fischetti was a well-known figure in Parisian society, and his untimely death sent shockwaves through the city.\nThe cause of death was reported as a heart attack, and it occurred at his home in the 8th arrondissement. Fischetti was known for his