BetaEdit: Null-Space Constrained Sequential Model Editing

Bingqing Liu[1,2], **Wei Liu**[1*], **Yuhua Li**[1*]
[1] School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2] Institute of Artificial Intelligence, Huazhong University of Science and Technology, China
{liubingqing24, idc_lw, idcliyuhua}@hust.edu.cn

Abstract

Null-space-based methods have garnered considerable attention in model editing by constraining updates to the null space of the pre-existing knowledge representation, thereby preserving the model's original behavior. However, in practice these methods rely on an approximate null space—leading to *knowledge leakage*—and further suffer from severe performance degradation during sequential editing. Recent work shows that *history-aware* editing strategies can empirically mitigate this decline, yet the underlying reason remains unclear. In this paper, we first expose the knowledge leakage inherent in existing null-space approaches and then analyze why history-aware updates effectively preserve both editing performance and general capabilities during long-horizon editing. Building on these insights, we propose BETAEDIT, a refined framework that effectively controls the knowledge leakage and integrates history-aware updates into the null-space paradigm. Extensive experiments on three large language models across two standard benchmarks show that BETAEDIT consistently outperforms prior methods in the challenging regime of massive-scale sequential editing. Code is available at: https://github.com/lbq8942/BetaEdit.

1 Introduction

Model editing seeks to locally modify a pre-trained model's behavior—e.g., updating a factual belief or correcting a misconception—without retraining from scratch or compromising its general capabilities [Jiang *et al.*, 2025b; Liu *et al.*, 2025b; Xie *et al.*, 2025; Yang *et al.*, 2025; Liu *et al.*, 2025c]. A central challenge in this task is balancing two competing objectives: (1) faithfully incorporating new knowledge, and (2) preserving the vast pre-trained knowledge that underpins the model's general capability. Early approaches often rely on explicit regularization to protect pre-existing knowledge [Meng *et al.*, 2022; Meng *et al.*, 2023; Gupta *et al.*, 2024b], but such methods struggle to simultaneously achieve high edit accuracy and strong edit locality [Fang *et al.*, 2025a].

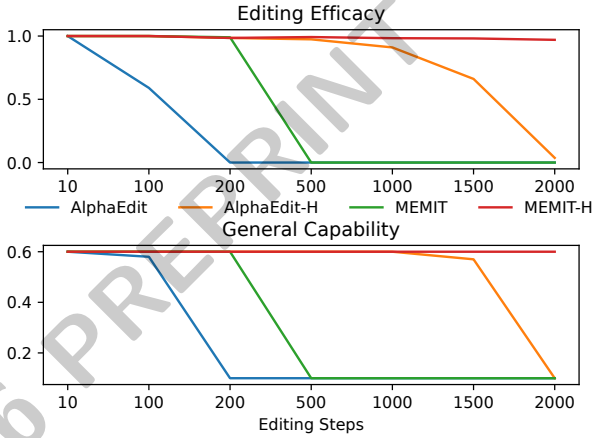

Figure 1: Editing efficacy and general capability (measured by Massive Multitask Language Understanding, MMLU) of various editing methods over 2,000 sequential edits on LLAMA3-8B using the CounterFact dataset. Markers with a "H" suffix indicate history-aware editing, while those without denote history-agnostic editing.

To circumvent this trade-off, null-space-based editing methods [Fang *et al.*, 2025a] propose a theoretically elegant solution: perform edits exclusively within the null space defined by pre-existing knowledge space. By construction, such edits are intended to induce zero change on the model's outputs for unedited inputs, thereby preserving pre-trained knowledge perfectly without explicit regularization. However, we observe that this ideal behavior exists only in theory and fails in practice. We identify the root cause: the "null space" used in practice is not exact, but an *approximate* (or *pseudo*) null space obtained via truncated singular value decomposition. This approximation inevitably introduces *knowledge leakage*, i.e., small but non-zero perturbations to pre-trained knowledge, which accumulates over sequential edits and eventually corrupts both editing fidelity and the model's general capabilities.

Recent history-aware editing strategy [Fang *et al.*, 2025a; Fang *et al.*, 2025b], which explicitly regularizes new updates based on past edits, has been shown to significantly mitigate this degradation. As shown in Figure 1, history-aware regularization substantially improves edit efficacy and preserves the model's general capability during sequential editing. The

*Corresponding author.

gain in edit efficacy is expected due to the protection of past edits; the concurrent preservation of the model's general capability, however, is counterintuitive. How the history-aware regularization—a single mechanism—achieves both effects during sequential editing remains unclear.

In this paper, we bridge these gaps. First, we formally identify and quantify the knowledge leakage inherent in current null-space methods (Section 3). Second, we provide a principled theoretical analysis (Section 4) showing that history-aware updates implicitly regularize the edit trajectory by reducing interference among sequential edits and constraining overall weight perturbation, thereby better preserving both editing fidelity and general capabilities. Building on these insights, we propose BETAEDIT (Section 5), a refined framework that (1) mitigates knowledge leakage by reintroducing a penalty term specifically designed to compensate for the imperfection of the pseudo-null space, and (2) seamlessly integrates history-aware updates into the null-space paradigm. Experiments on three large language models and two standard editing benchmarks demonstrate that BETAEDIT remains effective even after 10,000 sequential edits, whereas existing editors typically fail, severely corrupting the model's general capabilities and achieving near-zero editing efficacy.

2 Related Work

Model editing techniques are broadly classified according to whether they modify the original parameters of the pre-trained model.

Parameter-preserving editing methods aim to update or correct model behavior without altering the base weights. These approaches generally fall into two categories. The first introduces auxiliary modules to inject or override knowledge dynamically. For example, SERAC [Mitchell *et al.*, 2022b] employs an external memory store together with a lightweight corrective model; GRACE [Hartvigsen *et al.*, 2023] stores knowledge in discrete codebooks; while methods like CaliNet [Dong *et al.*, 2022], T-Patcher [Huang *et al.*, 2023], MELO [Yu *et al.*, 2024], and WISE [Wang *et al.*, 2024] incorporate trainable modules (e.g., LoRA adapters, or side memory) to store new knowledge. The second category relies on contextual prompting strategies—such as IKE [Zheng *et al.*, 2023], Deck [Bi *et al.*, 2025], and DecKER [Wang *et al.*, 2025]—which steer the model toward desired outputs at inference time through carefully designed prompts.

In contrast, **parameter-modifying editing** directly updates the model's internal weights. These methods typically follow one of two paradigms. The first leverages meta-learning frameworks, where a hypernetwork is trained to predict localized parameter adjustments. Notable examples include MEND [Mitchell *et al.*, 2022a], MALMEN [Tan *et al.*, 2024], InstructEdit [Zhang *et al.*, 2024], and RLEdit [Li *et al.*, 2025]. The second adopts a "locate-then-edit" strategy: it first pinpoints the specific activations or parameter subsets responsible for a target fact—often using gradient-based attribution or causal tracing [Zhou *et al.*, 2025]—and then applies precise modifications. Representative works include ROME [Meng *et al.*, 2022], MEMIT [Meng *et al.*, 2023], EMMET [Gupta *et al.*, 2024b], and AlphaEdit [Fang *et al.*, 2025a]. Recent efforts within this paradigm also focus on mitigating model degradation during sequential editing. Techniques along this line include post-editing weight regularization [Gu *et al.*, 2024; Ma *et al.*, 2025; Li and Chu, 2025; Liu *et al.*, 2025a; Qiao *et al.*, 2025] and history-aware regularization strategies that explicitly account for past edits [Fang *et al.*, 2025a; Fang *et al.*, 2025b; Guo *et al.*, 2025].

3 Knowledge Leakage in Null-Space Constrained Editing

We consider the *locate-then-edit* paradigm for model editing [Meng *et al.*, 2022], which proceeds in two steps: (1) identifying a critical layer that stores factual knowledge—commonly the second linear layer in the MLP of early transformer blocks—and (2) updating its weight matrix $\mathbf{W}$ to incorporate new information while preserving pre-existing knowledge. This is formalized as a regularized least-squares problem [Meng *et al.*, 2023]:

$$\begin{aligned}\Delta = \arg\min_{\tilde{\Delta}} &\left\|(\mathbf{W}+\tilde{\Delta})\mathbf{K}_1 - \mathbf{V}_1\right\|_F^2 \\ &+ \lambda_1 \left\|(\mathbf{W}+\tilde{\Delta})\mathbf{K}_0 - \mathbf{V}_0\right\|_F^2,\end{aligned} \tag{1}$$

where $(\mathbf{K}_0, \mathbf{V}_0)$ denotes pre-trained knowledge (satisfying $\mathbf{W}\mathbf{K}_0 = \mathbf{V}_0$ by construction), and $(\mathbf{K}_1, \mathbf{V}_1)$ represents the new factual update to be injected. The regularization parameter $\lambda_1 > 0$ controls the trade-off between edit accuracy and preservation of pre-trained knowledge. The closed-form solution to Eq. (1) is given by:

$$\Delta = \mathbf{R}_1\mathbf{K}_1^\top \left(\lambda_1\mathbf{K}_0\mathbf{K}_0^\top + \mathbf{K}_1\mathbf{K}_1^\top\right)^{-1}, \tag{2}$$

with $\mathbf{R}_1 = \mathbf{V}_1 - \mathbf{W}\mathbf{K}_1$ denoting the residual of the new knowledge.

To avoid compromising pre-trained knowledge, AlphaEdit [Fang *et al.*, 2025a] reparameterizes the update as $\tilde{\Delta}\mathbf{P}$, where $\mathbf{P}$ is a projection matrix onto the null space of $\mathbf{K}_0$, i.e., $\mathbf{P}\mathbf{K}_0 = \mathbf{0}$. With this constraint, the second term in Eq. (1) vanishes identically, yielding the simplified objective:

$$\Delta = \arg\min_{\tilde{\Delta}\mathbf{P}} \left\|(\mathbf{W}+\tilde{\Delta}\mathbf{P})\mathbf{K}_1 - \mathbf{V}_1\right\|_F^2. \tag{3}$$

This yields the following closed-form update:

$$\Delta = \mathbf{R}_1\mathbf{K}_1^\top\mathbf{P}\left(\lambda_2\mathbf{I} + \mathbf{K}_1\mathbf{K}_1^\top\mathbf{P}\right)^{-1}, \tag{4}$$

where $\lambda_2 > 0$ ensures numerical invertibility.

In theory, the null-space formulation guarantees perfect preservation of pre-trained knowledge: for any sequence of edits $\{\Delta^{(t)}\}_{t=1}^T$, the cumulative update satisfies $(\mathbf{W} + \sum_{t=1}^T \Delta^{(t)})\mathbf{K}_0 = \mathbf{V}_0$, since each $\Delta^{(t)}\mathbf{K}_0 = \mathbf{0}$. In practice, however, we observe a violation of this equality. To quantify the deviation, we define *knowledge leakage* as:

$$\mathrm{KL} = \left\|(\mathbf{W}+\Delta)\mathbf{K}_0 - \mathbf{V}_0\right\|_F,$$

which measures unintended perturbations to pre-trained knowledge after edit. Figure 2 shows the KL with respect to the cumulative update $\sum_{t=1}^T \Delta^{(t)}$ over $T = 2000$ sequential

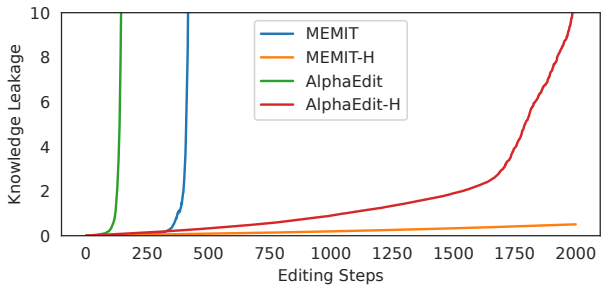

Figure 2: Knowledge leakage of existing editing methods, evaluated on LLAMA3-8B and CounterFact.

edits. Contrary to theoretical expectations, both null-space methods, AlphaEdit and AlphaEdit-H, exhibit non-zero and growing knowledge leakage, often exceeding that of baseline methods such as MEMIT [Meng *et al.*, 2023]. This directly contradicts the core promise of null-space editing and explains the catastrophic model corruption observed in long-horizon settings (cf. Figure 1).

The root cause lies in the *approximate* nature of the employed null space. In practice, $\mathbf{K}_0\mathbf{K}_0^\top$ is typically full-rank [Gupta *et al.*, 2025], which implies that all singular values of $\mathbf{K}_0\mathbf{K}_0^\top$ are positive, rendering the true null space trivial (i.e., $\{\mathbf{0}\}$). To address this, prior work treat singular values below a threshold (e.g., 0.02) as 0 [Fang *et al.*, 2025a]. This approximation implies that $\mathbf{P}\mathbf{K}_0 \neq \mathbf{0}$, causing each edit to introduce a small leakage that accumulates over time.

4 Theoretical Analysis: History-Aware Updates Reduce Weight Perturbation

Naively applying the update rules in Eq. 2 or Eq. 4 during sequential editing often leads to catastrophic degradation—both in editing efficacy and in the model's general capabilities [Gu *et al.*, 2024]. Recent work by [Fang *et al.*, 2025a; Fang *et al.*, 2025b] empirically demonstrates that this degradation can be substantially mitigated by explicitly preserving *previously edited facts* when performing new edits.

Concretely, under a history-aware editing mechanism, the MEMIT-style weight update at step t takes the form:

$$\Delta_t = r_t k_t^\top \left(k_t k_t^\top + \sum_{s=1}^{t-1} k_s k_s^\top + \lambda_1 \mathbf{K}_0 \mathbf{K}_0^\top \right)^{-1}, \quad (5)$$

where the key vectors from previous edits, $k_1, \ldots, k_{t-1}$, are explicitly incorporated into the t-th update. In contrast, the vanilla (history-agnostic) approach uses only:

$$\Delta_t = r_t k_t^\top \left(k_t k_t^\top + \lambda_1 \mathbf{K}_0 \mathbf{K}_0^\top \right)^{-1}. \quad (6)$$

Compared to Eq. (2), we use lowercase r_t and k_t here to denote vectors (rather than matrices), and the subscript t emphasizes the sequential editing setting.

While the improvement in editing accuracy is intuitive—since past edits are directly protected—it is less obvious why general capabilities are simultaneously preserved. In what follows, we provide a theoretical analysis showing that the history-aware mechanism not only safeguards previously edited knowledge but also better preserves the model's general capabilities during sequential editing, in contrast to the vanilla (history-agnostic) approach.

To ensure a clean and meaningful comparison, we first exclude certain trivial cases. For instance, if the residual vector r_t is zero, no actual edit occurs; if two edits introduce conflicting updates (e.g., "the capital of France is Paris" vs. "the capital of France is London"), they may cancel each other out, potentially restoring the model to its initial state without degrading general capabilities. Such scenarios do not reflect the steady growth of knowledge leakage observed in Figure 2. Therefore, we assume that the sequence of weight perturbations $\{\Delta_t\}$ is *non-conflicting*, i.e., $\langle \Delta_i, \Delta_j \rangle_F \geq 0$ for all i, j, which ensures non-decreasing knowledge leakage over time (see the supplementary material for details).

Under this editing scenario, the following theorem formalizes the advantage of the history-aware approach:

Theorem 1 (Reduced Weight Perturbation)**.** *Let $\Delta_t^{(A)}$ and $\Delta_t^{(B)}$ denote the weight perturbations at the t-th editing step for the vanilla (history-agnostic) method and the history-aware method, respectively. For any $T > 1$, the cumulative weight perturbation of the history-aware method is smaller in Frobenius norm than that of the vanilla method:*

$$\left\| \sum_{t=1}^{T} \Delta_t^{(B)} \right\|_F < \left\| \sum_{t=1}^{T} \Delta_t^{(A)} \right\|_F.$$

Proof sketch. The key insight is that the history-aware update $\Delta_t^{(B)}$ explicitly incorporates previously edited directions $\{k_s\}_{s<t}$ via a Gram matrix that includes $\sum_{s=1}^{t-1} k_s k_s^\top$. This effectively projects the new update onto a subspace orthogonal to past edit directions, thereby reducing interference with earlier updates, i.e., $\langle \Delta_t^{(B)}, \Delta_s^{(B)} \rangle_F \leq \langle \Delta_t^{(A)}, \Delta_s^{(A)} \rangle_F$ for all $s < t$, and yielding a smaller cumulative weight perturbation. Full details are provided in the supplementary material.

Figure 3(a) illustrates the evolution over time of the Frobenius norm of the cumulative weight update at each editing step. The history-aware method consistently suppresses the magnitude of perturbation more effectively than the history-agnostic method. This suppression becomes increasingly pronounced as the number of edits grows, since the likelihood of overlap between new and past edits increases. The history-aware method is able to orthogonalize these overlapping components, whereas the history-agnostic method conducts independent updates, leading to an accumulation of interference and an eventual explosion in cumulative weight perturbation, which corrupts the model's general capabilities (cf. Figure 1).

To further demonstrate the effectiveness of history-aware editing in suppressing weight perturbation, we evaluate several history-agnostic variants that attempt to reduce per-step weight perturbation:

- **MEMIT-30000**: Increases the regularization strength λ_1 from 15,000 to 30,000, thereby shrinking the magnitude of each update Δ_t.
- **MEMIT-R**: Replaces the historical keys $\{k_1, \ldots, k_{t-1}\}$ in Eq. (5) with random keys $\{k_1^r, \ldots, k_{t-1}^r\}$, which

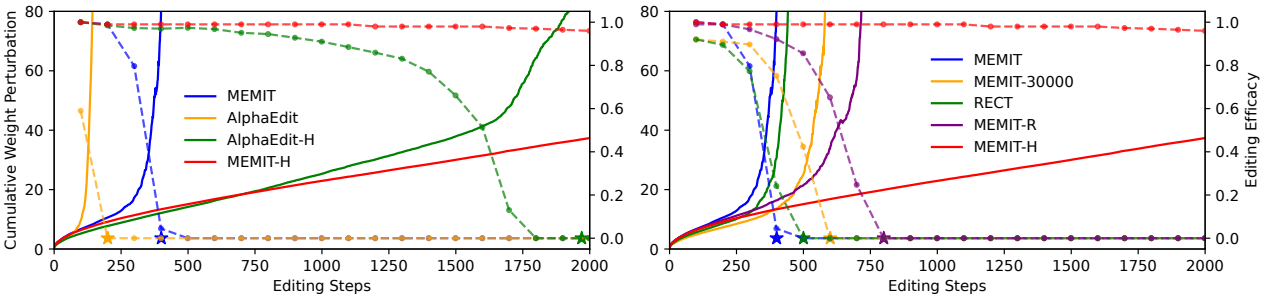

Figure 3: Comparison of cumulative weight perturbation between history-agnostic and history-aware editing, evaluated on LLaMA3-8B and the CounterFact dataset. The solid line shows the evolution of the Frobenius norm of the cumulative weight perturbation over 2000 editing steps (left y-axis), while the corresponding editing efficacy is indicated by the dashed line (right y-axis). Large cumulative weight perturbations degrade the model's general capabilities (measured by MMLU), with stars marking the onset of complete corruption.

strengthens the regularization but lacks meaningful alignment with historical edits.[1]

- **RECT** [Gu *et al.*, 2024]: Applies a sparsification strategy that zeroes out small-magnitude entries in the update matrix Δ_t, retaining only the largest components.

The results, shown in Figure 3(b), indicate that while these strategies somewhat alleviate cumulative weight perturbation, they incur a notable trade-off in terms of editing performance.

5 Method

As revealed in Section 3, existing null-space-based editing methods suffer from non-negligible *knowledge leakage*, which leads to cumulative corruption of the pretrained knowledge over sequential edits. To address this, we propose to explicitly account for the leakage during weight update. Specifically, instead of discarding the preservation term as in Eq. (3), we reintroduce it into the objective optimization:

$$\begin{aligned}\Delta = \arg\min_{\tilde{\Delta}\mathbf{P}} &\left\|(\mathbf{W}+\tilde{\Delta}\mathbf{P})\mathbf{K}_1 - \mathbf{V}_1\right\|_F^2 \\ &+\lambda_1 \left\|(\mathbf{W}+\tilde{\Delta}\mathbf{P})\mathbf{K}_0 - \mathbf{V}_0\right\|_F^2,\end{aligned} \tag{7}$$

which yields the following closed-form solution:

$$\Delta = \mathbf{R}_1\mathbf{K}_1^\top\mathbf{P}\left(\lambda_2\mathbf{I} + (\mathbf{K}_1\mathbf{K}_1^\top + \lambda_1\mathbf{K}_0\mathbf{K}_0^\top)\mathbf{P}\right)^{-1}, \tag{8}$$

where $\lambda_2 > 0$ ensures numerical stability (set to 10 following [Fang *et al.*, 2025a]), and $\lambda_1 > 0$ balances edit accuracy against knowledge leakage control.

Extending this to the sequential editing setting, we incorporate both pre-trained knowledge and all previously edited facts into the protection set. The update at step t is given by:

$$\Delta_t = \mathbf{R}_t\mathbf{K}_t^\top\mathbf{P}_t\left(\lambda_2\mathbf{I} + (\mathbf{K}_t\mathbf{K}_t^\top + \mathbf{C}^{(t)})\mathbf{P}_t\right)^{-1}, \tag{9}$$

where $\mathbf{R}_t = \mathbf{V}_t - \mathbf{W}\mathbf{K}_t$ and $\mathbf{C}^{(t)} = \lambda_1\mathbf{K}_0\mathbf{K}_0^\top + \sum_{i=1}^{t-1}\mathbf{K}_i\mathbf{K}_i^\top$ aggregates the representations of all protected knowledge up to step t, yielding a *history-aware* update.

[1]Since historical keys are derived from the subject token of each historical edit [Fang *et al.*, 2025a], the random keys are sampled from subject tokens of randomly selected edits.

Motivated by the theoretical analysis in Section 4—which shows that preserving editing history protects both editing performance and the model's general capabilities—we extend the null-space protection beyond the original knowledge $\mathbf{K}_0$ to also cover all previously edited knowledge $\{\mathbf{K}_i\}_{i=1}^{t-1}$. That is, instead of constructing the projector based solely on $\mathbf{K}_0\mathbf{K}_0^\top$ as in prior work [Fang *et al.*, 2025a], we define $\mathbf{P}_t$ as the orthogonal projector onto the null space of $\mathbf{C}^{(t)}$.

Concretely, we compute the eigendecomposition of $\mathbf{C}^{(t)} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and select the columns of $\mathbf{U}$ corresponding to eigenvalues smaller than a threshold ϵ (e.g., $\epsilon = 0.02$). Denoting this subset of eigenvectors as $\mathbf{U}_{\text{small}}$, we form the projection matrix as $\mathbf{P}_t = \mathbf{U}_{\text{small}}\mathbf{U}_{\text{small}}^\top$. By construction, $\mathbf{P}_t$ suppresses directions in which the model has already encoded factual knowledge, ensuring that new updates minimally perturb both pre-trained and edited facts. However, applying the null-space constraint to the editing history necessitates recomputing $\mathbf{P}_t$ at each edit step, which incurs substantial computational overhead. To balance efficiency and protection, we adopt a periodic refresh strategy: $\mathbf{P}_t$ is updated only every τ edits (e.g., $\tau = 1,000$). In the limiting case where $\tau \to \infty$, our method reduces to standard null-space editing that protects only the pretrained knowledge.

6 Experiments

6.1 Experimental Setup

LLMs, Datasets, and Evaluation Metrics. We evaluate our approach on three decoder-only language models: Qwen3-4B-Instruct-2507, GPT-J-6B, and LLaMA3-8B-Instruct. Following prior practice [Fang *et al.*, 2025a], we apply edits to layers {4,5,6,7,8} for LLaMA3 and Qwen3, and to layers {3,4,5,6,7,8} for GPT-J.

We assess editing performance on two standard factual editing benchmarks: CounterFact [Meng *et al.*, 2022] and ZsRE [Levy *et al.*, 2017]. Performance is measured using three standard metrics: **Efficacy (Eff.)**, which measures whether the edited model assigns the highest probability to the target output y for the input x (i.e., $y = \arg\max_{y'} P(y' \mid x)$); **Generality (Gen.)**, which evaluates whether the edited model generalizes to paraphrased inputs x_p (i.e., $y = \arg\max_{y'} P(y' \mid x_p)$); and **Specificity (Spe.)**,

Method		CounterFact									ZsRE								
		T = 300			T = 5000			T = 10000			T = 300			T = 5000			T = 10000		
		Eff.	Gen.	Spe.	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.	Eff.	Gen.	Spe.
Pre-edited	Qwen3-4B-Instruct	0.30	0.20	15.0	0.30	0.50	13.9	0.30	0.40	14.0	31.6	32.3	34.8	32.3	32.6	35.1	32.1	32.2	34.8
FT		43.6	4.10	3.20	0.00	0.00	0.00	0.00	0.00	0.00	38.5	7.20	11.3	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit		76.5	34.4	7.00	0.00	0.00	0.00	0.00	0.00	0.00	78.8	55.7	24.3	0.00	0.00	0.00	0.00	0.00	0.00
RECT		80.4	53.8	8.70	0.00	0.00	0.00	0.00	0.00	0.00	83.4	70.6	28.7	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE		85.0	55.5	8.30	0.00	0.00	0.00	0.00	0.00	0.00	89.3	76.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit		88.4	59.5	8.10	0.00	0.00	0.00	0.00	0.00	0.00	85.0	73.0	28.4	0.00	0.00	0.00	0.00	0.00	0.00
SimIE		95.0	71.3	11.8	0.00	0.00	0.00	0.00	0.00	0.00	95.2	90.6	34.3	0.00	0.00	0.00	0.00	0.00	0.00
EMMET		98.5	74.0	11.7	0.00	0.00	0.00	0.00	0.00	0.00	98.5	93.2	36.8	0.00	0.00	0.00	0.00	0.00	0.00
PMET		**98.8**	72.4	12.6	82.0	55.2	7.20	0.00	0.00	0.00	**98.6**	**94.0**	**37.3**	88.2	80.3	35.5	0.00	0.00	0.00
MEMIT		97.0	70.8	**13.2**	85.3	58.3	**8.10**	0.00	0.00	0.00	96.5	93.0	37.1	95.5	89.9	**38.9**	90.0	81.2	36.3
AlphaEdit		98.5	**74.1**	12.8	0.00	0.00	0.00	0.00	0.00	0.00	97.9	93.3	37.0	89.1	84.0	37.4	0.00	0.00	0.00
BetaEdit		98.0	73.5	13.0	**90.3**	**60.1**	8.00	**78.5**	**54.8**	**3.80**	97.5	93.0	36.7	**96.7**	**90.9**	38.1	**92.1**	**84.0**	**37.0**
Pre-edited	GPT-J-6B	0.50	0.50	15.2	0.40	0.50	14.3	0.40	0.50	14.4	26.5	26.7	28.0	27.0	26.2	27.0	27.8	26.5	26.0
FT		45.0	5.00	4.00	0.00	0.00	0.00	0.00	0.00	0.00	40.0	8.00	12.0	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit		80.0	36.0	5.50	0.00	0.00	0.00	0.00	0.00	0.00	81.0	58.0	19.0	0.00	0.00	0.00	0.00	0.00	0.00
RECT		82.0	52.0	9.50	0.00	0.00	0.00	0.00	0.00	0.00	84.0	68.0	24.7	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE		87.0	54.0	10.6	0.00	0.00	0.00	0.00	0.00	0.00	88.0	74.0	25.6	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit		89.0	58.0	10.0	0.00	0.00	0.00	0.00	0.00	0.00	86.0	71.0	24.8	0.00	0.00	0.00	0.00	0.00	0.00
SimIE		96.4	72.1	12.8	0.00	0.00	0.00	0.00	0.00	0.00	94.5	88.0	27.5	0.00	0.00	0.00	0.00	0.00	0.00
EMMET		98.8	73.0	13.5	0.00	0.00	0.00	0.00	0.00	0.00	98.0	92.0	28.8	94.3	87.5	26.7	87.5	82.6	22.8
PMET		98.5	72.5	13.5	85.0	50.0	**7.80**	30.0	18.0	2.50	98.0	93.0	29.4	89.0	81.0	**28.0**	0.00	0.00	0.00
MEMIT		97.8	71.0	13.8	74.5	40.4	7.60	27.2	15.5	2.30	96.5	92.0	**30.1**	95.0	89.0	24.0	0.00	0.00	0.00
AlphaEdit		**98.9**	73.5	13.7	95.9	50.9	6.90	82.6	40.3	3.90	**98.0**	**94.0**	29.5	96.5	92.0	26.0	88.3	83.1	22.8
BetaEdit		98.8	**74.0**	**13.9**	**96.6**	**52.8**	6.80	**86.0**	**41.8**	**4.10**	97.5	93.0	29.6	**98.9**	**95.1**	26.7	**92.5**	**86.5**	**24.7**
Pre-edited	LLaMA3-8B-Instruct	0.40	0.20	22.3	0.30	0.30	21.6	0.30	0.40	21.8	36.5	36.9	37.5	37.2	36.6	38.5	37.8	37.4	37.2
FT		48.0	5.90	8.70	0.00	0.00	0.00	0.00	0.00	0.00	42.0	8.00	12.0	0.00	0.00	0.00	0.00	0.00	0.00
RLEdit		80.0	38.0	14.7	0.00	0.00	0.00	0.00	0.00	0.00	82.0	60.0	28.0	0.00	0.00	0.00	0.00	0.00	0.00
RECT		84.0	58.0	17.0	0.00	0.00	0.00	0.00	0.00	0.00	86.0	74.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
PRUNE		88.0	60.0	17.5	0.00	0.00	0.00	0.00	0.00	0.00	90.0	80.0	36.0	0.00	0.00	0.00	0.00	0.00	0.00
AdaEdit		90.0	63.0	18.5	0.00	0.00	0.00	0.00	0.00	0.00	88.0	76.0	32.0	0.00	0.00	0.00	0.00	0.00	0.00
SimIE		96.0	68.0	20.7	0.00	0.00	0.00	0.00	0.00	0.00	96.5	92.0	36.0	0.00	0.00	0.00	0.00	0.00	0.00
EMMET		98.5	69.4	20.2	0.00	0.00	0.00	0.00	0.00	0.00	98.5	94.0	**40.5**	0.00	0.00	0.00	0.00	0.00	0.00
PMET		98.8	63.0	**21.6**	82.0	60.3	12.8	0.00	0.00	0.00	**98.8**	94.5	39.4	91.0	87.0	41.0	80.1	74.3	37.5
MEMIT		**99.3**	61.0	21.5	87.1	**65.4**	14.4	0.00	0.00	0.00	98.5	94.0	39.5	97.0	93.1	43.5	93.2	89.3	**44.2**
AlphaEdit		97.3	**71.5**	19.4	0.00	0.00	0.00	0.00	0.00	0.00	86.0	81.0	40.1	84.2	79.0	39.3	0.00	0.00	0.00
BetaEdit		96.8	70.4	19.0	**89.0**	62.4	**15.1**	**73.2**	**57.4**	**10.2**	98.0	**94.5**	40.3	**97.4**	**93.2**	**44.2**	**96.6**	**92.4**	43.9

Table 1: Editing performance across 300, 5,000, and 10,000 sequential edits on COUNTERFACT and ZSRE. Higher values are better for all metrics (efficacy, generality and specificity). Best and second-best results are bolded and underlined, respectively.

which measures whether the edited model preserves unrelated facts. Formally, for an unrelated input x_s and its corresponding output y_s , it requires that $y_s = \arg\max_{y'} P(y' \mid x_s)$.

Compared Baselines. We compare against a range of state-of-the-art model editing methods, categorized as follows: (i) fine-tuning with LoRA (FT) [Hu *et al.*, 2022]; (ii) hypernetwork-based editing (RLEDIT) [Li *et al.*, 2025]; (iii) locate-then-edit approaches, including MEMIT [Meng *et al.*, 2023], PMET [Li *et al.*, 2024], EMMET [Gupta *et al.*, 2024b], and ALPHAEDIT [Fang *et al.*, 2025a]; and (iv) post-hoc adjustment techniques for per-step weight update, including RECT [Gu *et al.*, 2024], PRUNE [Ma *et al.*, 2025], ADAEDIT [Li and Chu, 2025], and SIMIE [Guo *et al.*, 2025]. All locate-then-edit methods are adapted to the sequential editing setting using *history-aware* regularization.

6.2 Editing Performance

We evaluate editing performance at three scales—300, 5,000, and 10,000 sequential edits—to more comprehensively assess the scaling behavior of existing editors. Unlike some baselines that use large batch sizes (e.g., 20 or 100 [Li *et al.*, 2025; Fang *et al.*, 2025a]), we adopt the most challenging setting with a batch size of 1. Table 1 summarizes editing performance across three models and two datasets. We group editors into two categories: (1) *history-agnostic editors*, which perform each edit independently (e.g., FT, RECT and AdaEdit); and (2) *history-aware editors*, which explicitly preserve the editing history during new updates (e.g., SimIE, EMMET and BETAEDIT).

As shown in Table 1, history-aware editors consistently outperform the history-agnostic baselines. Notably, BETAEDIT achieves the best overall performance, particularly under large-scale sequential editing. Most baselines degrade catastrophically beyond 5,000 edits, whereas BETAEDIT remains effective even at 10,000 edits. We attribute this robustness to BETAEDIT's superior preservation of the model's general capabilities. Once these language capabilities are compromised (i.e., models are corrupted), subsequent editing becomes ineffective or even infeasible [Gupta *et al.*, 2024a]. Figure 4 presents three key metrics as the number of edits increases from 0 to 10,000: (i) knowledge leakage, (ii) cu-

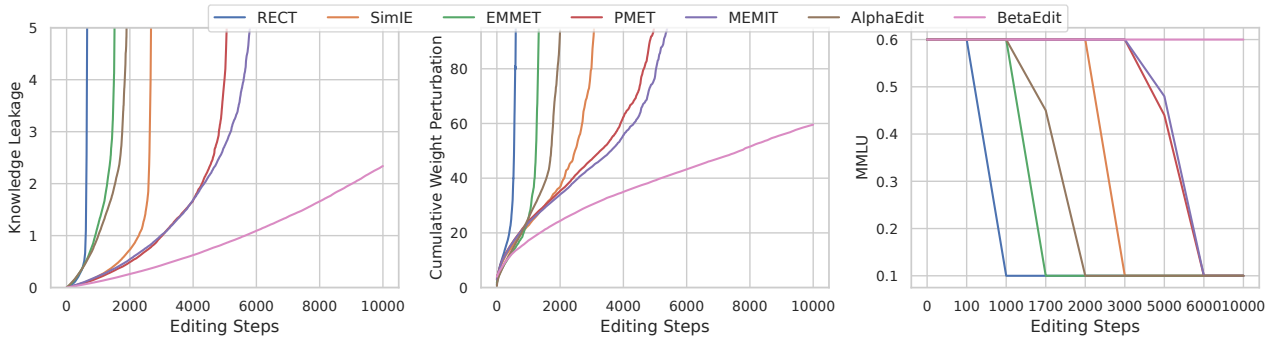

Figure 4: Evolution of model behavior during sequential editing on LLaMA3 using the CounterFact dataset: (left) knowledge leakage, (center) cumulative weight perturbation, and (right) MMLU performance, all measured as the number of edits increases from 0 to 10,000.

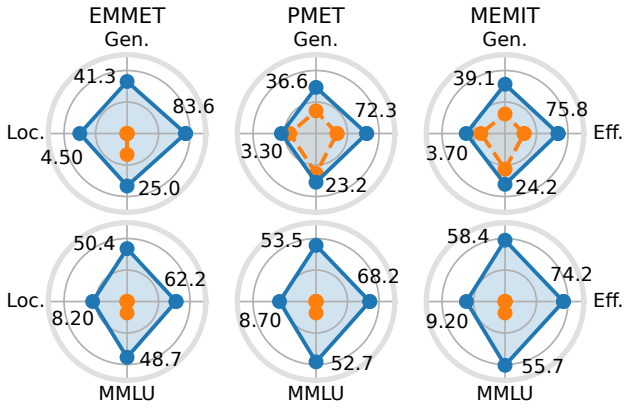

Figure 5: Editing performance and MMLU performance over 10,000 sequential edits with (blue) and without (orange) the null-space constraint on editing history, evaluated on GPT-J (top) and LLaMA3 (bottom) using the CounterFact dataset.

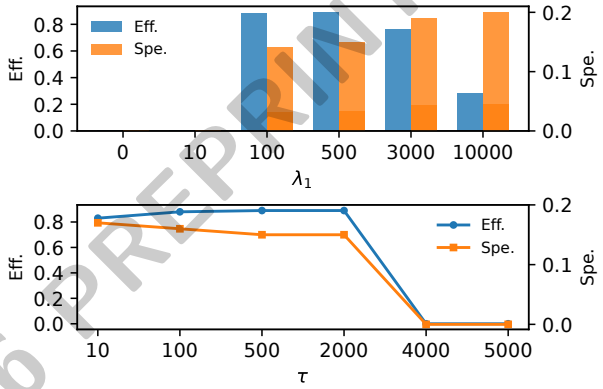

Figure 6: (Top) Impact of the penalty coefficient λ_1 for knowledge leakage; (Bottom) Impact of the refresh period τ for the null-space projector $\mathbf{P}_t$. Performance is measured in terms of editing efficacy (Eff., left y-axis) and specificity (Spe., right y-axis) on LLaMA3 over 5,000 sequential edits using the CounterFact dataset.

mulative weight perturbation, and (iii) MMLU performance (Macro-F1 score). BETAEDIT effectively suppresses knowledge leakage and limits cumulative parameter drift, thereby preserving the model's general capabilities.

Among the two recipes in BETAEDIT, knowledge leakage control is specifically designed for null-space-based editing methods, while null-space-based history-aware regularization is a general-purpose technique that effectively minimizes interference with historical edits during new updates, thereby better preserving both editing performance and the model's general capabilities—as established by our theoretical analysis in Section 4. To further evaluate its effectiveness, we apply this technique to the baseline editors EMMET, PMET, and MEMIT. As shown in Figure 5, when enhanced with null-space-based history-aware regularization, all editing methods remain effective under 10,000 sequential edits without corrupting the model.

6.3 Ablation Study and Hyperparameter Analysis

Ablation Study. We evaluate BETAEDIT under two ablated settings: (1) without the knowledge leakage (KL) penalty term, and (2) without the null-space (NS) constraint for preserving prior edits (denoted as *w/o KL* and *w/o NS*, respectively). The ablation results are shown in Table 2 (top: 5k edits; bottom: 10k edits). Two key observations emerge:

First, penalizing knowledge leakage is essential across both editing scales (5k and 10k), serving as a fundamental mechanism to protect pre-trained knowledge. Second, the null-space constraint on historical edits significantly enhances editing efficacy and preserves the model's general capabilities—particularly at 10k edits, where KL-only variants (*w/o NS*) typically achieve near-zero editing efficacy and corrupt the model's general capabilities. This underscores the critical role of protecting prior edits: it not only mitigates interference with editing history but also substantially curbs cumulative parameter perturbation—a key factor, as shown in Section 4, for preserving the model's general functionality during long sequences of edits.

Hyperparameter Analysis. We study two key hyperparameters in BETAEDIT: the knowledge leakage penalty coefficient λ_1 and the null-space projector refresh period τ.

The top panel of Figure 6 shows the effect of varying $\lambda_1 \in \{0, 10, 100, 500, 3000, 10000\}$ on LLAMA3 under 5,000 se-

Method		CounterFact						ZsRE					
		Qwen3		GPT-J		LLaMA3		Qwen3		GPT-J		LLaMA3	
		Eff.	MMLU	Eff.	MMLU	Eff.	MMLU	Eff.	MMLU	Eff.	MMLU	Eff.	MMLU
w/o KL	5k	0.00	10.0	95.9	24.2	0.00	10.0	0.00	10.0	96.4	24.5	0.00	10.0
w/o NS	5k	0.00	10.0	96.2	25.6	0.00	10.0	92.7	59.3	96.4	25.2	0.00	10.0
w/ Both	5k	90.3	60.9	96.6	25.0	89.0	59.0	96.7	61.8	98.9	25.0	97.4	60.5
w/o KL	10k	0.00	10.0	82.2	21.5	0.00	10.0	0.00	10.0	90.1	23.0	0.00	10.0
w/o NS	10k	0.00	10.0	81.4	22.3	0.00	10.0	0.00	10.0	90.4	23.2	0.00	10.0
w/ Both	10k	78.5	59.2	86.0	23.6	73.2	57.4	92.1	59.6	92.5	23.8	96.6	59.1

Table 2: Ablation study of BETAEDIT, evaluating the impact of removing the knowledge leakage (KL) penalty term or the null-space (NS) constraint for history edits. Results are shown for 5,000 edits (top block) and 10,000 edits (bottom block), reporting editing efficacy (%) and Macro-F1 score (%) of the MMLU task.

quential CounterFact edits. Small λ_1 (e.g., 0 or 10) leads to severe knowledge leakage, collapsing both efficacy and specificity near zero. Larger values better preserve pretrained knowledge and improve both metrics, but excessively large λ_1 (e.g., 10,000) over-constrains updates, limiting editability and degrading efficacy despite high specificity—highlighting a trade-off between editability and knowledge retention.

The bottom panel of Figure 6 examines the refresh interval τ, i.e., how often the null-space projector $\mathbf{P}_t$ is recomputed. With $\tau = 5000$ (no refresh), BETAEDIT relies only on standard history-aware regularization (*w/o NS*) and fails to sustain performance: both metrics drop near zero. Since recomputing $\mathbf{P}_t$ is costly, a moderate τ (e.g., 1,000) suffices to maintain strong editing performance while avoiding unnecessary computational overhead.

7 Limitations

Existing editing methods typically perform edits at the *subject* token, which aligns well with standard editing benchmarks such as CounterFact and ZsRE—datasets where each edit involves a distinct subject. In this setting, edits are generally non-conflicting, and our theoretical analysis of *history-aware* regularization holds reasonably well. However, the non-conflicting assumption can be significantly violated in two practical scenarios: (1) when edits are not anchored at the subject token, or (2) when multiple edits share the same subject. As shown in Table 3, anchoring edits at the last token drastically reduces the effectiveness of history-aware regularization. On LLaMA3, edit efficacy already drops to 54.7% after just 300 sequential edits, and by 2,000 edits the model is effectively corrupted—far below the 10,000-edit capacity achievable with subject-token anchoring. This is because the last token (e.g., "is") frequently appears across many edit prompts; anchoring knowledge update at this position introduces substantial interference among edits.

This reveals a key limitation of our approach: its dependence on subject-token anchoring imposes practical constraints. Specifically, it struggles in settings involving (i) multiple edits targeting the same subject [Dong *et al.*, 2025], or (ii) edits with complex or long-form prompts that deviate from simple subject–relation–object triplets and may involve multiple subjects [Jiang *et al.*, 2025a; Zeng *et al.*, 2025].

Method		Qwen3		GPT-J		LLaMA3	
		Eff.	MMLU	Eff.	MMLU	Eff.	MMLU
Last	300	22.4	47.3	38.7	18.3	26.3	46.7
Last+H	300	50.8	52.0	65.4	20.3	54.7	49.5
Sub.	300	82.4	60.2	96.0	25.1	79.7	57.0
Sub.+H	300	98.0	62.9	98.8	25.3	96.8	60.1
Last	2000	0.00	10.0	0.00	10.0	0.00	10.0
Last+H	2000	0.00	10.0	0.00	10.0	0.00	10.0
Sub.	2000	0.00	10.0	0.00	10.0	0.00	10.0
Sub.+H	2000	97.1	62.4	98.0	25.0	95.6	59.8

Table 3: Effectiveness of history-aware regularization (*H*) under different anchoring strategies: last token (*Last*) vs. subject token (*Sub.*). Results are reported for 300 (top) and 2,000 (bottom) sequential edits on the CounterFact dataset.

8 Conclusion

We have uncovered a fundamental gap between the theoretical promise and practical behavior of null-space-based model editing: despite being designed to preserve pre-trained knowledge perfectly, these methods suffer from *knowledge leakage* due to the use of approximate null spaces, leading to catastrophic failure in long-horizon sequential editing. Beyond identifying this issue, we provide a principled theoretical analysis showing that *history-aware* regularization serves a dual role—it not only protects previously edited facts but also implicitly constrains overall parameter drift, thereby preserving the model's general capabilities.

Building on these insights, we propose BETAEDIT, a refined editing framework that explicitly mitigates knowledge leakage by reintroducing a dedicated penalty term, while seamlessly integrating history-aware updates into the null-space paradigm. Experiments across multiple large language models and standard benchmarks show that BETAEDIT maintains high editing performance and effectively preserves the model's general capabilities—even under up to 10,000 sequential edits, a regime in which prior methods typically collapse—demonstrating its effectiveness for long-horizon model editing.

Acknowledgments

This work is supported by the National Key Research and Development Program of China under grant 2024YFC3307900; the National Natural Science Foundation of China under grants 62436003, 62206102, 62376103 and 62302184 ; Major Science and Technology Project of Hubei Province under grant 2024BAA008; Hubei Science and Technology Talent Service Project under grant 2024DJC078; and Ant Group through CCF-Ant Research Fund. The computation is completed in the HPC Platform of Huazhong University of Science and Technology.

References

[Bi *et al.*, 2025] Baolong Bi, Shenghua Liu, Lingrui Mei, Yiwei Wang, Junfeng Fang, Pengliang Ji, and Xueqi Cheng. Decoding by contrasting knowledge: Enhancing large language model confidence on edited facts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17198–17208, 2025.

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

[Dong *et al.*, 2025] Zilu Dong, Xiangqing Shen, and Rui Xia. Memit-merge: Addressing memit's key-value conflicts in same-subject batch editing for llms. *arXiv preprint arXiv:2502.07322*, 2025.

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

[Fang *et al.*, 2025b] Quntian Fang, Zhen Huang, Zhiliang Tian, Minghao Hu, Dongsheng Li, Yiping Yao, Xinyue Fang, Menglong Lu, and Guotong Geng. Hippocampal-like sequential editing for continual knowledge updates in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

[Guo *et al.*, 2025] Yaming Guo, Siyang Guo, Hengshu Zhu, and Ying Sun. Towards lifelong model editing via simulating ideal editor. In *Forty-second International Conference on Machine Learning*, 2025.

[Gupta *et al.*, 2024a] Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15202–15232, 2024.

[Gupta *et al.*, 2024b] Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. A unified framework for model editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15403–15418, 2024.

[Gupta *et al.*, 2025] Akshat Gupta, Maochuan Lu, Thomas Hartvigsen, and Gopala Anumanchipalli. Efficient knowledge editing via minimal precomputation. *arXiv preprint arXiv:2506.04226*, 2025.

[Hartvigsen *et al.*, 2023] Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.

[Hu *et al.*, 2022] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[Huang *et al.*, 2023] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[Jiang *et al.*, 2025b] Houcheng Jiang, Junfeng Fang, Tianyu Zhang, Baolong Bi, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. Neuron-level sequential editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16678–16702, 2025.

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

[Li and Chu, 2025] Qi Li and Xiaowen Chu. Adaedit: Advancing continuous knowledge editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4127–4149, 2025.

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

[Li *et al.*, 2025] Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *Forty-second International Conference on Machine Learning*, 2025.

[Liu *et al.*, 2025a] Jinzhe Liu, Junshu Sun, Shufan Shen, Chenxue Yang, and Shuhui Wang. Edit less, achieve more: Dynamic sparse neuron masking for lifelong knowledge editing in llms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[Liu *et al.*, 2025b] Tianci Liu, Ruirui Li, Zihan Dong, Hui Liu, Xianfeng Tang, Qingyu Yin, Linjun Zhang, Haoyu Wang, and Jing Gao. Mitigating heterogeneous token overfitting in llm knowledge editing. In *Forty-second International Conference on Machine Learning*, 2025.

[Liu *et al.*, 2025c] Tianci Liu, Ruirui Li, Yunzhe Qi, Hui Liu, Xianfeng Tang, Tianqi Zheng, Qingyu Yin, Monica Xiao Cheng, Jun Huan, Haoyu Wang, et al. Unlocking efficient, scalable, and continual knowledge editing with basis-level representation fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[Meng *et al.*, 2022] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

[Meng *et al.*, 2023] Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[Mitchell *et al.*, 2022b] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.

[Qiao *et al.*, 2025] Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. Wasserstein distance constraint and parameter sparsification for batched and iterative knowledge editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25019–25028, 2025.

[Tan *et al.*, 2024] Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[Wang *et al.*, 2025] Changyue Wang, Weihang Su, Qingyao Ai, Yujia Zhou, and Yiqun Liu. Decoupling reasoning and knowledge injection for in-context knowledge editing. *arXiv preprint arXiv:2506.00536*, 2025.

[Xie *et al.*, 2025] Jiakuan Xie, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Revealing the deceptiveness of knowledge editing: A mechanistic analysis of superficial editing. *arXiv preprint arXiv:2505.12636*, 2025.

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

[Yu *et al.*, 2024] Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457, 2024.

[Zeng *et al.*, 2025] Li Zeng, Zeming Liu, Chong Feng, He-Yan Huang, and Yuhang Guo. Docmedit: Towards document-level model editing. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19725–19743, 2025.

[Zhang *et al.*, 2024] Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. Instructedit: instruction-based knowledge editing for large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 6633–6641, 2024.

[Zheng *et al.*, 2023] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876, 2023.

[Zhou *et al.*, 2025] Wei Zhou, Wei Wei, Guibang Cao, and Fei Wang. Editing memories through few targeted neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26111–26119, 2025.