# Rethinking Sequential Locate-Then-Edit: Optimality and Stability

**Bingqing Liu**[1] **Wei Liu**[1,*] **Zhiying Deng**[2] **Jun Wang**[3] **Yuhua Li**[1] **Ruixuan Li**[1]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology
[2] Laboratory for Artificial Intelligence and New Forms of Education, CCNU [3]iWudao Tech
{liubingqing24, idc_lw, idcliyuhua, rxli}@hust.edu.cn
zhiyingdzy@ccnu.edu.cn, jwang@iwudao.tech
[*]Corresponding authors. Code

## Abstract

The *Locate-Then-Edit* framework enables efficient knowledge updates in large language models without requiring full retraining. Recent research has increasingly focused on **sequential editing**, where knowledge arrives continuously and the model is updated incrementally. However, existing sequential methods commonly suffer from catastrophic forgetting and model collapse after only a few hundred edits. In contrast, **batch editing**—which jointly optimizes all edits and yields the optimal solution under a specified editing objective—can support tens of thousands of edits with high edit efficacy while preserving the model's general capabilities, resulting in a substantial performance gap between the two editing paradigms. In this paper, we demonstrate that this gap can be closed via a **batch-equivalent sequential editing** approach. Moreover, we investigate the underlying mechanism of model degradation in existing sequential methods by analyzing the convergence behavior of weight perturbations as edits accumulate. Our analysis reveals that current methods typically exhibit linear or logarithmic growth in perturbation, whereas the proposed batch-equivalent method achieves time-invariant growth, thereby theoretically guaranteeing long-term stability and preserving the integrity of pretrained knowledge throughout massive-scale sequential editing.

## 1 Introduction

Updating the knowledge encoded in the parameters of Large Language Models (LLMs) is essential for refreshing outdated facts and integrating domain-specific information to enable effective customization [Sanu et al., 2024]. However, retraining LLMs from scratch is often prohibitively expensive due to the immense computational cost and time required. While fine-tuning offers a more practical alternative, it frequently suffers from catastrophic forgetting of previously acquired knowledge [Luo et al., 2025]. To address these challenges, knowledge editing (KE) methods [Wang et al., 2024c] have emerged as promising approaches that allow precise, efficient, and localized modifications to specific factual associations within LLMs without compromising model's behavior on unrelated knowledge.

Among KE paradigms, the *locate-then-edit* framework has garnered significant attention [Meng et al., 2022]. Early efforts within this framework primarily focused on *batch editing* [Meng et al., 2023, Gupta et al., 2024c], in which all edits are applied and optimized jointly, yielding superior performance under large-scale editing scenarios. However, real-world knowledge evolves continuously, demanding timely updates for LLMs to remain accurate and relevant [Jiang et al., 2025b]. This practical demand has made sequential (also called lifelong or continual) knowledge editing a significant and growing focus of research [Gu et al., 2024b, Li and Chu, 2024, 2025, Liu et al., 2026a,b].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

In sequential editing, edits arrive one after another, and the model must be updated incrementally. This paradigm introduces two key challenges: repeated editing can lead to (1) forgetting of previously edited knowledge and (2) degradation of the model's pre-trained general capabilities, often termed *model collapse* in severe cases [Gupta et al., 2024a, Fang et al., 2025a]. The former compromises editing effectiveness, while the latter is especially catastrophic: once foundational language understanding deteriorates, the model becomes fundamentally unusable, which in turn invalidates any subsequent editing attempts [Gupta et al., 2024b]. Figure 1 illustrates the editing performance of state-of-the-art approaches under 10,000 sequential edits. While the *history-regularized* editor AlphaEdit [Fang et al., 2025a] significantly outperforms the *history-agnostic* MEMIT [Meng et al., 2023], both methods fail after only 2,000 edits, suffering sharp declines in both editing success rate and general language capability. In stark contrast, batch editing remains effective even after 10,000 edits, revealing a substantial performance gap between batch and sequential approaches.

This observation raises two fundamental questions: (1) Is the gap between batch and sequential editing inevitable? Can we design a sequential method that *exactly closes* this gap? (2) Why does *batch editing* better preserve the model's general capabilities than *history-regularized* editing, and why does history-regularized editing outperform *history-agnostic* editing? What underlying mechanism drives the progressive degradation across these strategies?

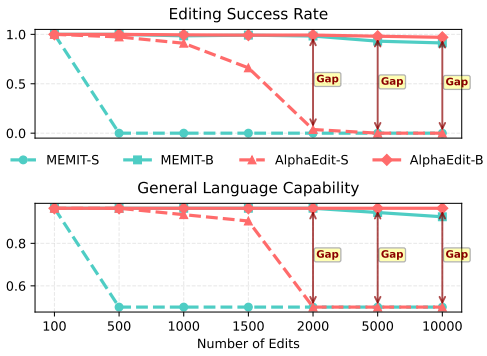



Figure 1: Performance gap between sequential editing (-S) and batch editing (-B) on the Counterfact dataset using LLaMA3-8B-Instruct. **Top**: Editing success rate. **Bottom**: General language capability (accuracy on SST sentiment classification from GLUE benchmark).

In this paper, within the *locate-then-edit* framework, we address both questions. First, we propose a sequential editing algorithm that is *exactly equivalent* to batch editing: at every editing step, the model produced by our method is identical to the one obtained by applying all edits up to that point via a single batch optimization, thereby attaining the optimal solution under the specified editing objective. Second, we conduct a convergence analysis of existing sequential editing methods by characterizing the temporal evolution of model weights. Our analysis reveals a fundamental distinction in weight perturbation dynamics: our method (which matches batch editing) exhibits *time-invariant* perturbations; in contrast, history-aware methods suffer from *logarithmic growth*, while history-agnostic methods incur *linear growth* in perturbation magnitude. This theoretical insight explains the empirical model degradation trends in Figure 1 and establishes a clear connection between update dynamics and long-term editing stability.

## 2 Related Work

Model editing aims to update specific knowledge in large language models while preserving unrelated facts and general capabilities. Approaches are broadly categorized by whether they follow the *locate-then-edit* (LTE) paradigm.

### 2.1 Locate-Then-Edit (LTE) Model Editing

LTE operates in two stages: first, it identifies model components—such as activations or parameter subsets—causally responsible for a target fact, often via neuron-level attribution or causal tracing [Geva et al., 2021, Yu and Ananiadou, 2024, Zhou et al., 2025, Guo et al., 2025a, Jiang et al., 2025b, Liu et al., 2025a]; second, it applies targeted modifications. ROME [Meng et al., 2022] pioneered this by editing feed-forward network (FFN) parameters for single-fact updates. Later works extended ROME to batch settings: MEMIT [Meng et al., 2023], PMET [Li et al., 2024], and EMMET [Gupta et al., 2024c] support thousands of concurrent edits. To better preserve pre-trained knowledge, methods based on null-space constraints and adaptive regularization have been introduced [Fang et al., 2025a, Liu et al., 2026a,b]. Subsequent research has explored complex scenarios including long-form

editing [Jiang et al., 2025a, Zeng et al., 2025, Li et al., 2025b], multilingual and multimodal updating [Sun et al., 2025, Wang et al., 2025c], and multi-hop reasoning [Zhong et al., 2023, Zhang et al., 2025, Thede et al., 2025].

A key challenge in LTE is *sequential editing*, where repeated updates degrade performance, e.g., eroding basic language understanding capability [Gu et al., 2024b]. Two main strategies address this: (1) post-editing weight regularization, which constrains parameters or their spectral properties [Gu et al., 2024b, Ma et al., 2025, Li and Chu, 2025, Liu et al., 2025a, Qiao et al., 2025, Liu et al., 2026c]; and (2) history-aware regularization that leverages past edits for stability [Fang et al., 2025a, Guo et al., 2025b, Fang et al., 2025b, Liu et al., 2026a].

### 2.2 Non-LTE Model Editing

Non-LTE methods can be categorized according to whether they modify the original model parameters. Parameter-preserving methods keep the original model parameters unchanged and incorporate edits through auxiliary mechanisms. Examples include external memory modules [Mitchell et al., 2022b], neuron modulation modules [Dong et al., 2022, Huang et al., 2023], external knowledge bases [Hartvigsen et al., 2023, Fei et al., 2025], low-rank adapters [Yu et al., 2024], and side networks [Wang et al., 2024b]. Other methods intervene at inference time, for example, by augmenting prompts with retrieved information to guide model outputs without modifying model parameters [Madaan et al., 2022, Zheng et al., 2023, Wang et al., 2024d, Gu et al., 2024a, Bi et al., 2025, Wang et al., 2025a]. Parameter-modifying methods edit knowledge by directly modifying the model's internal parameters. One line of work adopts direct fine-tuning [Gangadhar and Stratos, 2024, Wang et al., 2024a, Liu et al., 2025b,c, Piao et al., 2025], while another employs meta-learning to generate weight updates via hypernetworks, as in MEND [Mitchell et al., 2022a], MALMEN [Tan et al., 2024], InstructEdit [Zhang et al., 2024], LKS [Liu et al., 2025d], and RLEdit [Li et al., 2025c].

## 3 Background

**Knowledge Location** Our scope is the *locate-then-edit* (LTE) paradigm. LTE first identifies internal model elements that most influence a factual output using causal tracing, gradient sensitivity, or neuron attribution [Yu and Ananiadou, 2024, Zhou et al., 2025, Guo et al., 2025a, Jiang et al., 2025b, Liu et al., 2025a]. It then applies targeted modifications. In Transformers, factual knowledge is often stored in feed-forward networks (FFNs) as key-value associative memories [Geva et al., 2021]. Within an FFN, the first linear layer computes a key, and the second retrieves the value. Early-layer FFNs initiate fact retrieval [Meng et al., 2022]. Thus, mainstream LTE methods edit the second linear layer's weights in early FFN blocks to insert or alter knowledge [Meng et al., 2023].

**Knowledge Representation** For a factual triple $(s, a, o)$ — e.g., (*United States*, *President*, *Donald Trump*) — model editing aims to make the model generate target answer $o$ when prompted with *prompt*$(s, a)$ (e.g., "The president of the United States is"). Let $\mathbf{W}_0$ be the weight matrix identified as storing this fact, mapping input key $k$ to output value $v$. The key $k$ is the last-token activation of subject $s$ (e.g., *States*), empirically most effective for editing [Meng et al., 2022]. To steer the model toward $o$, the target value $v$ is optimized via gradient descent to maximize the likelihood of generating $o$ given *prompt*$(s, a)$.

**Knowledge Editing** Suppose we are provided with $t$ key–value edit pairs $(k_i, v_i)_{i=1}^{t}$. The goal of knowledge editing is to incorporate new knowledge while preserving pre-trained knowledge. Let $\mathbf{K}_0 = [k_1^{\text{pre}}, \ldots, k_n^{\text{pre}}]$ and $\mathbf{V}_0 = [v_1^{\text{pre}}, \ldots, v_n^{\text{pre}}]$ denote the matrices of pre-trained keys and values from the initial weight $\mathbf{W}_0$. The objective is to find a perturbation $\Delta$ such that $\mathbf{W}_0 + \Delta$ correctly maps new pairs $(k_i, v_i)_{i=1}^{t}$ while minimally altering behavior on $(\mathbf{K}_0, \mathbf{V}_0)$. This yields the following constrained optimization problem [Meng et al., 2022]:

$$\Delta^* = \arg\min_{\Delta} \|(\mathbf{W}_0 + \Delta)\mathbf{K}_0 - \mathbf{V}_0\|_F^2, \quad \text{s.t. } (\mathbf{W}_0 + \Delta)k_i = v_i, \quad \forall\, i = 1, \ldots, t. \tag{1}$$

## 4 Method

The formulation in Eq. 1 defines a batch editing scheme that optimizes all $t$ edits simultaneously. While effective, it fails to handle edits that arrive sequentially in practice. Furthermore, storing

all key–value pairs incurs linear memory growth in $t$, which becomes prohibitive at scale. These limitations motivate sequential editing.

### 4.1 Problem Formulation

We first formalize the sequential editing problem [Ma et al., 2025, Li et al., 2025c], which aims to incrementally update the model weights as new edit requests arrive.

**Definition 4.1 (Sequential Editing).** *Consider a sequence of edit requests $(k_i, v_i)_{i=1}^{t}$ , where each $k_i$ is an input key and $v_i$ is the desired output value. A sequential editing method is defined by an update operator $f$ that recursively updates the model weight starting from the initial weight $\mathbf{W}_0$:*

$$\mathbf{W}_{j+1} = f(\mathbf{W}_j, k_{j+1}, v_{j+1}),$$

Before designing such an operator $f$, we must first define what constitutes an *optimal* edit. Specifically, given the first $j$ edit requests, what is the ideal weight matrix $\mathbf{W}_j^*$? Only with this target clearly defined can we design a sequential algorithm $f$ whose output $\mathbf{W}_j$ either approximates or, ideally, exactly matches the optimal solution $\mathbf{W}_j^*$.

At its core, *locate-then-edit* can be viewed as a key-value remapping problem: given a set of new key-value pairs to insert and pretrained key-value pairs to preserve, an editing objective specifies the criteria for achieving this remapping. Different choices of objectives lead to different notions of optimality. Prior work by Guo et al. [2025b] defines the optimal edit in alignment with ROME [Meng et al., 2022] and EMMET [Gupta et al., 2024c], which impose equality constraints on the new key–value pairs. In contrast, we adopt the formulation from MEMIT [Meng et al., 2023] as our optimal editing objective, which casts editing as a regularized least-squares problem.

**Definition 4.2 (Optimal Editing Objective).** *Let $\mathbf{K}_{[1:j]} = [k_1, \ldots, k_j]$ and $\mathbf{V}_{[1:j]} = [v_1, \ldots, v_j]$ denote the new key-value pairs to be inserted, and let $\mathbf{K}_0 = [k_1^{pre}, \ldots, k_n^{pre}]$ and $\mathbf{V}_0 = [v_1^{pre}, \ldots, v_n^{pre}]$ represent the pre-trained key-value pairs to be preserved. We adopt the following regularized least-squares objective as the optimal editing objective:*

$$\mathcal{L}_j(\Delta) = \lambda_1 \left\| (\mathbf{W}_0 + \Delta)\mathbf{K}_{[1:j]} - \mathbf{V}_{[1:j]} \right\|_F^2 + \lambda_2 \left\| (\mathbf{W}_0 + \Delta)\mathbf{K}_0 - \mathbf{V}_0 \right\|_F^2. \tag{2}$$

We augment the editing objective (Eq. 2) with two coefficients $\lambda_1$ and $\lambda_2$ to obtain a more general formulation. Specifically, it unifies major existing *locate-then-edit* (LTE) methods as limiting cases: as $\lambda_1 \to \infty$, the solution enforces exact satisfaction of the new edits, recovering methods such as ROME and EMMET; conversely, as $\lambda_2 \to \infty$, the update is constrained to the null space of $\mathbf{K}_0$, prioritizing preservation of pre-trained knowledge, as in AlphaEdit [Fang et al., 2025a]. A detailed discussion is provided in Appendix A.

Minimizing the objective in Eq. 2 yields the optimal weight update [Meng et al., 2023]:

$$\Delta_j^* = \arg\min_{\Delta} \mathcal{L}_j(\Delta) = \lambda_1 \mathbf{R}_{[1:j]} \mathbf{K}_{[1:j]}^\top \left( \lambda_1 \mathbf{K}_{[1:j]} \mathbf{K}_{[1:j]}^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top \right)^{-1}, \tag{3}$$

where $\mathbf{R}_{[1:j]} = [r_1, \ldots, r_j] = \mathbf{V}_{[1:j]} - \mathbf{W}_0 \mathbf{K}_{[1:j]}$ denotes the matrix of residuals. Accordingly, the optimal weight matrix is given by $\mathbf{W}_j^* = \mathbf{W}_0 + \Delta_j^*$. Thus, under our definition, an optimal sequential operator $f^*$ should yield the same weight update as batch editing, which optimizes all edits jointly.

### 4.2 Batch-Equivalent Sequential Editing

For the first edit ($j = 1$), sequential editing trivially coincides with batch editing. Our goal is to extend this property to arbitrary $j$: can we derive a sequential update operator $f^*$ such that applying it sequentially yields the same result as batch editing at every step? Formally, we seek $f^*$ satisfying

$$\mathbf{W}_{j+1}^* = f^*(\mathbf{W}_j^*, k_{j+1}, v_{j+1}), \qquad \forall j \geq 1,$$

where $\mathbf{W}_j^*$ denotes the batch solution for the first $j$ edits. When such an operator exists, it is by Definition 4.2 optimal and called *batch-equivalent*.

To derive such an operator, we start from the closed-form batch solution in Eq. 3. Define

$$A_j := \lambda_1 \sum_{i=1}^{j} k_i k_i^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top, \quad B_j := \lambda_1 \sum_{i=1}^{j} r_i k_i^\top. \tag{4}$$

By Eq. 3, the optimal weight update for the first $j$ edits is $\Delta_j^* = B_j A_j^{-1}$, with $\mathbf{W}_j^* = \mathbf{W}_0 + \Delta_j^*$. We aim to find a sequential operator $f^*$ that connects $\mathbf{W}_{j+1}^*$ and $\mathbf{W}_j^*$. The following theorem provides such an operator.

**Theorem 4.1** (**Batch-Equivalent Sequential Update**). *Let $\mathbf{W}_j^*$ denote the optimal weight matrix for the first $j$ edits. Then the optimum $\mathbf{W}_{j+1}^*$ at the next step can be obtained from $\mathbf{W}_j^*$ via*

$$\mathbf{W}_{j+1}^* = \mathbf{W}_j^* \big(\mathbf{I} - \lambda_1 k_{j+1} k_{j+1}^\top A_{j+1}^{-1}\big) + \lambda_1 \big(r_{j+1} + \mathbf{W}_0 k_{j+1}\big) k_{j+1}^\top A_{j+1}^{-1}, \tag{5}$$

*where the right-hand side defines the optimal sequential operator $f^*$.*

Here the Gram matrix $A_{j+1}$ can be computed incrementally via $A_{j+1} = A_j + \lambda_1 k_{j+1} k_{j+1}^\top$, with $A_j$ cached from the $j$-th step. Starting from $j = 1$, the operator $f^*$ sequentially updates both the Gram matrix $A_j$ and the weight matrix $\mathbf{W}_j^*$, thereby matching the batch solution at every editing step. Its proof, which relies on the Sherman–Morrison–Woodbury (SMW) identity [Golub and Van Loan, 2013] and straightforward algebra, is deferred to Appendix B.

Theorem 4.1 gives the optimal sequential operator that matches the MEMIT-style editing objectives. Similarly, optimal sequential operators can be derived for other LTE methods under their respective editing objectives. For AlphaEdit, for instance, the operator takes the following form:

$$\mathbf{W}_{j+1} = \mathbf{W}_j \big(\mathbf{I} - k_{j+1} k_{j+1}^\top \mathbf{P} A_{j+1}^{-1}\big) + \big(r_{j+1} + \mathbf{W}_0 k_{j+1}\big) k_{j+1}^\top \mathbf{P} A_{j+1}^{-1},$$

where $\mathbf{P}$ is the projection matrix. The derivation is also deferred to Appendix B.

In Appendix B, we show that although Theorem 4.1 assumes one edit at a time, it naturally generalizes to mini-batch sequential editing. We also discuss how the proposed sequential operator $f^*$ degenerates to existing sequential methods when certain components are removed. Note that although $f^*$ is theoretically equivalent to batch editing, finite-precision recursive updates may introduce numerical errors that propagate across editing steps, leading to deviations from the batch solution. We discuss these numerical discrepancies and practical implementation considerations in Appendix C.

## 5 Stability Analysis

In prior work, the failure of sequential editing is typically observed through surface-level symptoms—such as degradation on general language understanding benchmarks (e.g., GLUE) [Gu et al., 2024b, Gupta et al., 2024b, Ma et al., 2025, Park et al., 2026] or shifts in the distribution of hidden representations in Transformers [Fang et al., 2025a, Li and Chu, 2025, Li et al., 2025a, Liu et al., 2025a]. However, these observations do not address the root cause: since the LTE method directly modifies model parameters, the fundamental issue lies in how the weight matrix $\mathbf{W}_t$ evolves over successive edits. While a few studies analyze weight dynamics, they are either limited to single-edit settings or rely on overly idealized assumptions [Ma et al., 2025, Fang et al., 2025b, Liu et al., 2026c]; see Appendix D for a detailed discussion.

To systematically investigate the weight dynamics during sequential editing, we first introduce two mild conditions that are standard, and often implicitly satisfied, in current LTE frameworks:

**Assumption 5.1** (**Boundedness and Invertibility**). *The following conditions hold throughout the sequential editing process:*

- *(i)* ***Boundedness:*** *There exist constants $C_k > 0$ and $C_r > 0$ such that for all edit steps $t$, $\|k_t\|_2 \le C_k$ and $\|r_t\|_2 \le C_r$.*
- *(ii)* ***Invertibility:*** *There exists a constant $\alpha > 0$ such that $\mathbb{E}[kk^\top] \succeq \alpha \mathbf{I}$, where $k$ denotes any key vector from either the pre-trained knowledge bank or an incoming edit request. This ensures that the Gram matrix in Eq. (3) remains well-conditioned during long-term editing, enabling stable inversion in LTE updates.*

Under Assumption 5.1, we present a unified characterization of the weight dynamics across existing sequential LTE methods and our batch-equivalent approach. To this end, we first categorize existing approaches into three representative classes: history-agnostic sequential editing, history-regularized sequential editing, and post-hoc regularized sequential editing.

**History-Agnostic Editing** Methods in this class apply each edit independently, without accounting for prior edits [Meng et al., 2022, 2023, Gupta et al., 2024c]. The update rule is:

$$\mathbf{W}_t = \mathbf{W}_{t-1} + r_t k_t^\top (k_t k_t^\top + \mathbf{K}_0 \mathbf{K}_0^\top)^{-1}.$$

**History-Regularized Editing** History-regularized methods incorporate past edits $\mathbf{K}_{[1:t-1]}$ into the regularization term to preserve previously learned key–value associations. Initially proposed by Fang et al. [2025a], Guo et al. [2025b], Fang et al. [2025b], this mechanism has since become a *de facto* prerequisite for achieving state-of-the-art performance in sequential editing [Li et al., 2025a, Liu et al., 2025a, 2026c, Park et al., 2026]. The update rule is given by:

$$\mathbf{W}_t = \mathbf{W}_{t-1} + r_t k_t^\top \left(k_t k_t^\top + \mathbf{K}_{[1:t-1]} \mathbf{K}_{[1:t-1]}^\top + \mathbf{K}_0 \mathbf{K}_0^\top\right)^{-1}.$$

**Post-Hoc Regularized Editing** Given a weight perturbation $\Delta$ computed by a third-party editing method, post-hoc regularization applies a regularization constraint. Examples include zeroing out small-magnitude entries [Gu et al., 2024b], discarding singular components with small singular values [Li and Chu, 2025], or projecting the perturbation onto a non-core subspace to preserve the original weight space [Liu et al., 2026c, Park et al., 2026]. Despite their differences in strategy, these approaches can be uniformly expressed by the following update rule:

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \mathrm{PR}(\Delta, \alpha),$$

where $\alpha$ is a hyperparameter controlling the regularization strength. For instance, if the regularization operation zeroes out small entries, then $\alpha$ denotes the proportion of entries to zero out. In this case, $\alpha = 1$ yields $\mathrm{PR}(\Delta, \alpha) = \mathbf{0}$, and $\alpha = 0$ yields $\mathrm{PR}(\Delta, \alpha) = \Delta$.

Given these update rules, we now analyze their stability during long-term sequential editing by studying the evolution of the Frobenius norm $\|\mathbf{W}_t\|_F$. Since unbounded growth of this norm can lead to numerical instability or model collapse, we focus on deriving the upper bound of $\|\mathbf{W}_t\|_F$, which characterizes the worst-case magnitude of weight perturbations. (The lower bound is trivially zero for all sequential editing methods and thus less informative for stability analysis.) Under Assumption 5.1, we establish the following guarantees for weight dynamics:

**Theorem 5.1.** *Let $\lambda_{\min}$ denote the smallest eigenvalue of $\mathbf{K}_0 \mathbf{K}_0^\top$. The Frobenius norm of $\mathbf{W}_t$ for the above classes of sequential LTE methods satisfies the following tight upper bounds:*

$$\begin{aligned}
\|\mathbf{W}_t^{\mathrm{HA}}\|_F &\le \|\mathbf{W}_0\|_F + \frac{C_r}{2\sqrt{\lambda_{\min}}} \cdot t, \\
\|\mathbf{W}_t^{\mathrm{HR}}\|_F &\le \|\mathbf{W}_0\|_F + \frac{C_r C_k}{\alpha} (1 + \log t), \\
\|\mathbf{W}_t^{\mathrm{BE}}\|_F &\le \|\mathbf{W}_0\|_F + \frac{C_r C_k}{\alpha}, \\
\|\mathbf{W}_t^{\mathrm{PR}}\|_F &\le \|\mathbf{W}_0\|_F + B_{\mathrm{base}}(t),
\end{aligned}$$

*where $\mathbf{W}_t^{\mathrm{HA}}$, $\mathbf{W}_t^{\mathrm{HR}}$, $\mathbf{W}_t^{\mathrm{BE}}$, and $\mathbf{W}_t^{\mathrm{PR}}$ denote the weights under history-agnostic, history-regularized, batch-equivalent, and post-hoc regularized sequential editing, respectively. $B_{\mathrm{base}}(t)$ denotes the corresponding upper bound of the underlying base editor (HA, HR, or BE).*

Proofs are provided in Appendix E. Theorem 5.1 characterizes the worst-case weight growth during sequential editing: $\|\mathbf{W}_t^{\mathrm{HA}}\|_F$ can grow linearly with $t$, $\|\mathbf{W}_t^{\mathrm{HR}}\|_F$ at most logarithmically, and $\|\mathbf{W}_t^{\mathrm{BE}}\|_F$ remains uniformly bounded—demonstrating the superior long-term editing stability of our approach. For post-hoc regularized editing ($\|\mathbf{W}_t^{\mathrm{PR}}\|_F$), the norm bound inherits that of its underlying base editor (HA, HR, or BE). Although these bounds are worst-case guarantees, our experiments (Section 6) reveal a clear and consistent gap in the actual norm growth across these methods, closely matching the theoretical growth hierarchy.

# 6 Experiments

## 6.1 Experimental Setup

Our experimental design strictly follows the taxonomy introduced in Section 5. We investigate how different sequential editing paradigms—history-agnostic, history-regularized, batch-equivalent, and post-hoc regularized—affect both editing performance and editing stability.

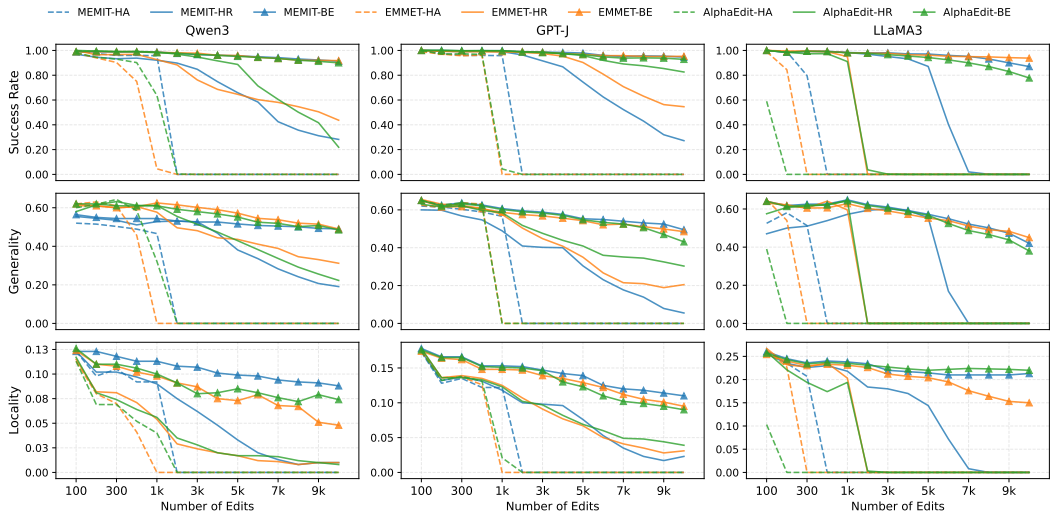



Figure 2: Editing performance (success rate, generality, locality) on the COUNTERFACT dataset over 10,000 sequential edits. Columns (from left to right): Qwen3-8B, GPT-J, LLaMA3-8B-Instruct. Higher values indicate better performance for all metrics.

**LLMs and Datasets.** We evaluate our methods on four autoregressive language models: GPT2-XL-1.5B, GPT-J-6B, Qwen3-8B, and LLaMA3-8B-Instruct. These models cover a range of model architectures and sizes. All model checkpoints are publicly available through the Hugging Face Hub.[1] Following prior work [Fang et al., 2025a], we perform edits on layers $\{13, 14, 15, 16, 17\}$ for GPT2-XL, layers $\{3, 4, 5, 6, 7, 8\}$ for GPT-J, and layers $\{4, 5, 6, 7, 8\}$ for Qwen3-8B and LLaMA3-8B-Instruct. Results on GPT2-XL are deferred to Appendix G. Evaluations are conducted on two standard factual editing benchmarks: COUNTERFACT [Meng et al., 2022] and ZSRE [Levy et al., 2017]. Throughout this paper, the edit sequence length is fixed at 10,000. We adopt the most challenging editing setting—applying only one edit per step (i.e., batch size = 1). This setup allows us to rigorously stress-test the model's sequential editing performance over 10,000 update steps.

**Base Editors and Post-Hoc Regularization Strategies.** We build upon three foundational LTE editors: MEMIT [Meng et al., 2023], EMMET [Gupta et al., 2024c] (an extension of ROME [Meng et al., 2022]), and AlphaEdit [Fang et al., 2025a]. For each, we develop **history-agnostic** (MEMIT-HA, EMMET-HA, AlphaEdit-HA), **history-regularized** (MEMIT-HR, EMMET-HR, AlphaEdit-HR), and **batch-equivalent** (MEMIT-BE, EMMET-BE, AlphaEdit-BE) variants. Their update rules are detailed in Appendix F. For post-hoc regularization, we include RECT [Gu et al., 2024b], PRUNE [Ma et al., 2025], ADAEDIT [Li and Chu, 2025], and SPHERE [Liu et al., 2026c]. Additional baselines (e.g., parameter-preserving methods) are discussed in Appendix G.

### 6.2 Hierarchical Editing Performance

**Evaluation Metrics** We evaluate editing performance using three standard metrics [Fang et al., 2025a, Li et al., 2025a]. **Success Rate** measures whether the edited model assigns the highest probability to the target output for the input prompt. **Generality** measures whether the edited model produces the target output to paraphrased input prompts. **Locality** measures whether the edited model preserves the original output for unrelated inputs. Details are provided in Appendix F.

**Main Results** Figure 2 presents the editing performance on the COUNTERFACT dataset. Results on ZSRE are provided in Appendix G. Our findings reveal a clear performance hierarchy. First, history-regularized variants—the current state of the art—significantly outperform their history-agnostic counterparts, sustaining reliable performance over thousands of sequential edits, whereas history-agnostic methods typically degrade after only a few hundred [Thede et al., 2025]. Second, and more notably, our proposed batch-equivalent sequential editing method achieves the best long-term

[1] https://huggingface.co/models

Table 1: Detailed performance of MEMIT-BE on Qwen3-8B across different numbers of sequential edits on CounterFact and ZsRE. SR., Gen., and Loc. denote Success Rate, Generality, and Locality, respectively. SST, MMLU, CoLA, and NLI report accuracy (%) on the corresponding general language understanding benchmarks.

| Edits | CounterFact | | | | | | | ZsRE | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR. | Gen. | Loc. | SST | MMLU | CoLA | NLI | SR. | Gen. | Loc. | SST | MMLU | CoLA | NLI |
| 1k | 98.8 | 54.4 | 11.3 | 83.0 | 72.5 | 71.5 | 83.5 | 94.8 | 89.1 | 38.2 | 84.0 | 73.5 | 72.0 | 82.5 |
| 2k | 97.6 | 53.1 | 10.8 | 83.0 | 72.5 | 70.5 | 84.0 | 94.5 | 88.0 | 38.1 | 84.5 | 75.5 | 72.0 | 82.5 |
| 3k | 96.7 | 52.6 | 10.7 | 83.5 | 72.5 | 71.5 | 83.0 | 93.9 | 88.4 | 37.6 | 81.0 | 75.0 | 72.0 | 81.5 |
| 4k | 96.1 | 52.6 | 10.1 | 83.5 | 72.0 | 71.5 | 83.0 | 93.6 | 87.2 | 37.6 | 81.5 | 75.5 | 71.0 | 81.5 |
| 5k | 95.7 | 51.6 | 9.9 | 84.0 | 72.5 | 72.0 | 83.0 | 93.4 | 86.7 | 37.8 | 82.5 | 74.5 | 73.5 | 82.5 |
| 6k | 94.9 | 50.8 | 9.8 | 83.5 | 70.5 | 69.0 | 82.0 | 92.5 | 86.1 | 37.5 | 83.5 | 74.5 | 75.0 | 81.5 |
| 7k | 94.2 | 50.5 | 9.4 | 83.5 | 72.5 | 70.0 | 81.5 | 91.9 | 85.0 | 37.3 | 81.0 | 73.5 | 75.5 | 80.0 |
| 8k | 93.1 | 50.1 | 9.2 | 82.0 | 71.0 | 70.0 | 82.0 | 91.3 | 84.8 | 37.5 | 81.0 | 74.5 | 74.5 | 81.0 |
| 9k | 92.3 | 49.3 | 9.1 | 80.5 | 69.5 | 72.5 | 81.5 | 91.2 | 84.7 | 37.3 | 81.5 | 75.0 | 74.0 | 83.0 |
| 10k | 91.3 | 49.0 | 8.8 | 81.5 | 70.5 | 72.0 | 80.5 | 90.9 | 84.5 | 37.2 | 82.5 | 70.5 | 73.0 | 83.0 |

Table 2: Layer-wise weight norms of MEMIT-BE on Qwen3-8B across different numbers of sequential edits on CounterFact and ZsRE.

| Edits | CounterFact | | | | | ZsRE | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Layer 4 | Layer 5 | Layer 6 | Layer 7 | Layer 8 | Layer 4 | Layer 5 | Layer 6 | Layer 7 | Layer 8 |
| 1k | 170 | 177 | 181 | 182 | 191 | 168 | 176 | 181 | 182 | 191 |
| 2k | 177 | 178 | 182 | 183 | 194 | 173 | 177 | 181 | 183 | 193 |
| 3k | 183 | 180 | 183 | 184 | 196 | 178 | 179 | 182 | 184 | 195 |
| 4k | 188 | 182 | 184 | 186 | 199 | 183 | 180 | 183 | 184 | 197 |
| 5k | 193 | 183 | 185 | 187 | 202 | 187 | 181 | 184 | 185 | 199 |
| 6k | 198 | 185 | 186 | 188 | 204 | 192 | 182 | 184 | 186 | 201 |
| 7k | 202 | 186 | 187 | 189 | 207 | 196 | 184 | 185 | 187 | 203 |
| 8k | 206 | 187 | 187 | 190 | 209 | 200 | 185 | 186 | 188 | 205 |
| 9k | 209 | 188 | 188 | 191 | 212 | 205 | 186 | 187 | 189 | 207 |
| 10k | 212 | 189 | 189 | 192 | 214 | 208 | 187 | 187 | 190 | 209 |

performance, maintaining high performance across all three metrics even after tens of thousands of edits and establishing a consistent gap over both history-regularized and history-agnostic baselines. For a more concrete numerical view, Table 1 reports the results of MEMIT-BE on Qwen3-8B at different editing scales, complementing the trends shown in Figure 2. The results further demonstrate the strong editing performance and robustness of batch-equivalent sequential editing as the number of edits increases. In principle, we should also compare our method with batch editors to verify empirical equivalence beyond theoretical equivalence. We leave this comparison to Appendix C.

### 6.3 Hierarchical Editing Stability

Previous work has shown that editing methods suffer catastrophic performance degradation during long-term sequential editing, with success rates, generality, and locality all collapsing to zero [Gu et al., 2024b, Gupta et al., 2024b, Liu et al., 2025a]. This raises a deeper question: does this failure reflect a decline limited to editing-specific metrics, or does it indicate a more fundamental breakdown in the model's overall behavior under sequential editing?

**Evaluation Metrics** To investigate this question, we monitor two complementary aspects of model stability during sequential editing: (i) the preservation of general-purpose language understanding, evaluated using four benchmarks: SST-2, MMLU, CoLA, and NLI, which assess sentiment classification, general knowledge, linguistic acceptability, and natural language inference, respectively [Fang et al., 2025a]; (ii) the evolution of the Frobenius norm $\|\mathbf{W}_t\|_F$ of the edited weights at step $t$, which characterizes the magnitude of parameter changes. Additional stability metrics, including deviation from the initial weights and leakage of pretrained knowledge, are reported in Appendix G.

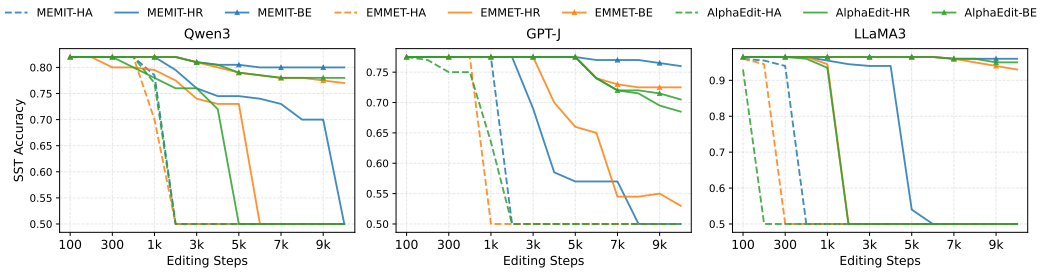



Figure 3: Evolution of language understanding capability (SST accuracy) during 10,000 sequential edits on the COUNTERFACT dataset.

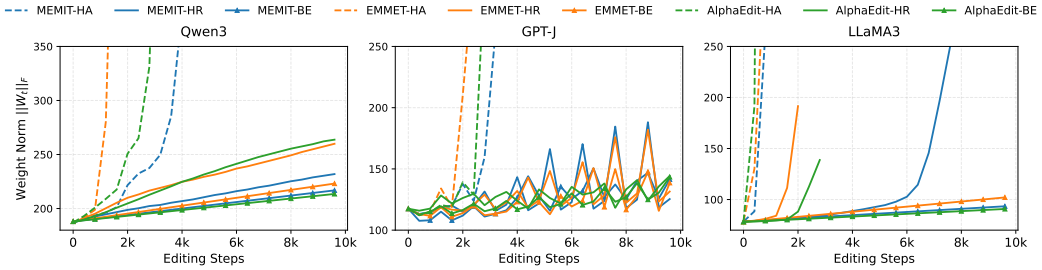



Figure 4: Evolution of the weight norm during 10,000 sequential edits on the COUNTERFACT dataset.

**Main Results** Figures 3 and 4 show SST-2 accuracy and weight norm evolution, respectively, on COUNTERFACT.[2] Results on ZSRE are provided in Appendix G. We observe a strong correspondence between parameter drift and the degradation of general-purpose language understanding, consistent with the decline in editing-specific metrics reported earlier. As the edited weights increasingly deviate from their initial values, SST-2 accuracy tends to deteriorate. For example, with MEMIT-HR, the weight norm grows dramatically around 6,000 edits, coinciding with a sharp decline in both SST-2 accuracy and editing-specific performance.

Tables 1 and 2 provide detailed numerical results for MEMIT-BE on Qwen3-8B across different editing scales. The former evaluates the preservation of general-purpose language understanding, while the latter presents layer-wise weight norms across the five edited layers. These results show that batch-equivalent sequential editing preserves general-purpose language understanding while maintaining relatively stable layer-wise weight norms even after 10,000 edits.

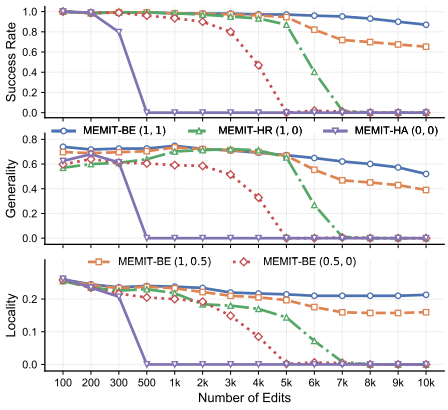



Figure 5: Interpolation results for MEMIT-BE with different interpolation parameters $\beta_1$ and $\beta_2$ (values shown in parentheses).

Moreover, our results reveal a clear stability hierarchy among the three editing paradigms. History-agnostic editors exhibit rapid and unbounded weight growth, accompanied by a substantial degradation of language understanding within only a few hundred edits. History-regularized editors significantly mitigate parameter drift and better preserve model's general capabilities, but their performance could still deteriorate after thousands of edits, underperforming batch-equivalent sequential editors. These empirical trends are consistent with our theoretical upper-bound analysis in Section 5: history-agnostic methods could exhibit linear weight perturbation growth, history-regularized methods admit a logarithmic growth bound, and our

[2]Because the weight norm exhibits strong oscillations (especially for GPT-J), plotting every editing step yields dense, unreadable curves. We therefore subsample every 400 steps to reveal the underlying trend; full step-by-step results are provided in Appendix G.

method guarantees a constant upper bound on weight perturbations. Note that the explosive weight growth observed in the *near-collapse* phase does not contradict these bounds, as the theoretical upper bounds $C_k$ and $C_r$ on the key and residual vectors can scale with the model weights $\mathbf{W}_t$.

### 6.4 Interpolation Between Hierarchies

We note that the hierarchical gap between history-agnostic, history-regularized, and batch-equivalent methods is not insurmountable. Instead, these methods can be unified within a single formulation governed by two interpolation parameters, $\beta_1$ and $\beta_2$, which control history protection and batch-equivalence, respectively (see Appendix B.3 for details). By varying these parameters, the formulation accommodates a continuum of editing behaviors. For example, setting $\beta_1 = 1$ and $\beta_2 = 0.5$ yields an intermediate variant between batch-equivalent and history-regularized editing. Figure 5 illustrates this interpolation for MEMIT-BE on the COUNTERFACT dataset using LLaMA. Results on ZSRE are provided in Appendix G.

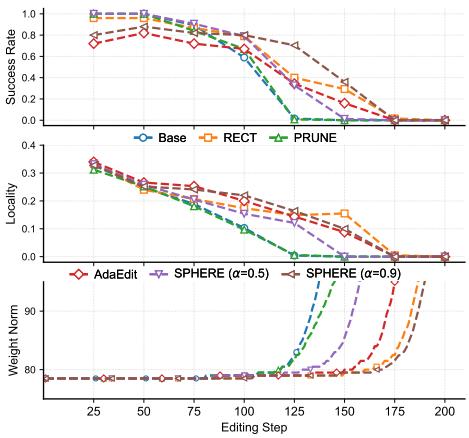



Figure 6: Effect of post-hoc regularization on AlphaEdit-HA (base), with $\alpha$ denoting the regularization strength.

### 6.5 Role of Post-Hoc Regularized Editing

Theorem 5.1 shows that post-hoc regularization inherits the stability behavior of the underlying sequential editor. We empirically examine this effect in Figure 6, which compares four post-hoc regularization strategies (RECT, PRUNE, ADAEDIT, and SPHERE) applied to AlphaEdit-HA on the COUNTERFACT dataset using LLaMA. These strategies interpolate between the original update ($\alpha = 0$) and a zero update ($\alpha = 1$), trading editing efficacy for reduced weight perturbation. As shown in the figure, although post-hoc regularization extends the editing horizon and slows down model collapse, its benefits remain limited to a few hundred edits. This leaves a substantial gap relative to history-regularized sequential editors, which sustain stable editing performance across thousands of updates. Additional results and details on $\alpha$ are provided in Appendix G.

## 7 Conclusion and Limitations

We have addressed a fundamental limitation in sequential knowledge editing within the *locate-then-edit* (LTE) framework: the substantial performance gap between practical sequential updates and idealized batch editing. By designing a sequential algorithm that is exactly equivalent to batch editing, we close this gap, enabling tens of thousands of sequential edits. Furthermore, our convergence analysis reveals that long-term stability critically depends on controlling weight perturbations. This insight not only explains empirical observations of model degradation in prior methods but also provides a principled criterion for designing future sequential editing algorithms. Importantly, since batch LTE is secretly sequential LTE under specific editing objectives, our findings suggest that research efforts can productively refocus on advancing batch editing techniques—knowing that improvements there will directly carry over to the sequential setting. We outline several promising future research directions in Appendix H.

Our batch-equivalence guarantee requires the key-residual pairs defining the editing objective to remain fixed throughout the editing process, which can be achieved by retaining the original model weights when extracting them. However, this alone is insufficient to address the challenges of multi-layer editing, which presents a limitation of our theoretical framework. In multi-layer editing algorithms where updates to earlier layers affect the representations used to construct editing targets for subsequent layers [Meng et al., 2023, Li et al., 2025a, Liu et al., 2026d], these key-residual pairs may change dynamically. Therefore, our framework only applies to multi-layer editing strategies with independent editing objectives across layers. Similar settings are also adopted in prior work on batch and sequential editing equivalence [Guo et al., 2025b, Wang et al., 2026]. Addressing such cross-layer dependencies is an important direction for extending our framework.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2024YFC3307900; the National Natural Science Foundation of China under Grants 62376103, 62302184, 62436003, 62472332, and 62672205; the Major Science and Technology Project of Hubei Province under Grants 2025BAB011, 2024BAA008, and 2024BAB039; the Hubei Science and Technology Talent Service Project under Grant 2024DJC078; the Science and Technology Innovation Talent Program under grant 2023CB005; and Wuhan Dameng Database Co., Ltd.

## References

Adi Ben-Israel and Thomas NE Greville. *Generalized inverses: theory and applications*. Springer, 2003.

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.

Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. Eamet: Robust massive model editing via embedding alignment optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

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.

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.

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*, 2025a.

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*, 2025b.

Weizhi Fei, Hao Shi, Jing Xu, Jingchen Peng, Jiazheng Li, Jingzhao Zhang, Bo Bai, Wei Han, Zhenyuan Chen, and Xueyan Niu. Neuraldb: Scaling knowledge editing in llms to 100,000 facts with neural kv database. *arXiv preprint arXiv:2507.18028*, 2025.

Govind Krishnan Gangadhar and Karl Stratos. Model editing by standard fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5907–5913, 2024.

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

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. Pokemqa: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8069–8083, 2024a.

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, 2024b.

Phillip Huang Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite. Mechanistic unlearning: Robust knowledge unlearning and editing via mechanistic localization. In *Forty-second International Conference on Machine Learning*, 2025a.

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*, 2025b.

Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding rome: Resolving model collapse during sequential model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21738–21744, 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, 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, 2024c.

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.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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.

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*, 2025a.

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, 2025b.

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.

Qi Li and Xiaowen Chu. Can we continually edit language models? on the knowledge attenuation in sequential model editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5438–5455, 2024.

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.

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.

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

Xinye Li, Zunwen Zheng, Qian Zhang, Dekai Zhuang, Jiabao Kang, Liyan Xu, Qingbin Liu, Xi Chen, Zhiying Tu, Dianhui Chu, et al. Scedit: Script-based assessment of knowledge editing. *arXiv preprint arXiv:2505.23291*, 2025b.

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*, 2025c.

Bingqing Liu, Wei Liu, and Yuhua Li. Betaedit: Null-space constrained sequential model editing. In Diego Calvanese, editor, *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pages 4563–4571. International Joint Conferences on Artificial Intelligence Organization, 8 2026a. doi: 10.24963/ijcai.2026/508. URL https://doi.org/10.24963/ijcai.2026/508. Main Track.

Bingqing Liu, Wei Liu, Jun Wang, Xiaobo Sun, Yuhua Li, and Ruixuan Li. Editing large language models with geometry-aware regularization, 2026b. URL https://arxiv.org/abs/2611.06185.

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*, 2025a. URL https://openreview.net/forum?id=feAzLLT9to.

Qingyuan Liu, Jia-Chen Gu, Yunzhi Yao, Hong Wang, and Nanyun Peng. Energy-regularized sequential model editing on hyperspheres. In *The Fourteenth International Conference on Learning Representations*, 2026c. URL https://openreview.net/forum?id=CHsdtzCip6.

Shuxin Liu and Jianhao Zhang. Cake: Causal-guided adaptive knowledge editing for llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20040–20070, 2026.

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*, 2025b.

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*, 2025c.

Wei Liu, Hongkai Liu, Zhiying Deng, Yee Whye Teh, and Wee Sun Lee. From backward spreading to forward replay: Revisiting target construction in llm parameter editing. In *Forty-third International Conference on Machine Learning*, 2026d.

Xin Liu, Qiyang Song, Shaowen Xu, Kerou Zhou, Wenbo Jiang, Xiaoqi Jia, Weijuan Zhang, Heqing Huang, and Yakai Li. Latent knowledge scalpel: Precise and massive knowledge editing for large language models. *arXiv preprint arXiv:2508.03741*, 2025d.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 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.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve gpt-3 after deployment. *arXiv preprint arXiv:2201.06009*, 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.

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.

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

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, 2022b.

Haewon Park, Sangwoo Kim, and Yohan Jo. Suit: Knowledge editing with subspace-aware key-value mappings. In *The Fourteenth International Conference on Learning Representations*, 2026.

Hongming Piao, Hao Wang, Dapeng Wu, and Ying Wei. A$^3$e: Towards compositional model editing. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=OwUOmgfKUi.

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.

Erin Sanu, T Keerthi Amudaa, Prasiddha Bhat, Guduru Dinesh, Apoorva Uday Kumar Chate, and Ramakanth Kumar. Limitations of large language models. In *2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–6. IEEE, 2024.

Wei Sun, Tingyu Qu, Mingxiao Li, Jesse Davis, and Marie Francine Moens. Mitigating negative interference in multilingual knowledge editing through null-space constraints. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8796–8810, 2025.

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.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. Wikibigedit: Understanding the limits of lifelong knowledge editing in llms. In *Forty-second International Conference on Machine Learning*, 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*, 2025a.

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

Ke Wang, QIN Yiming, Nikolaos Dimitriadis, Alessandro Favero, and Pascal Frossard. Memoir: Lifelong model editing with minimal overwrite and informed retention for llms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b.

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, 2024b.

Ruipeng Wang, Junfeng Fang, Jiaqi Li, Hao Chen, Jie Shi, Kun Wang, and Xiang Wang. Ace: Concept editing in diffusion models without performance degradation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10525–10534, 2025c.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024c.

Weixuan Wang, Barry Haddow, and Alexandra Birch. Retrieval-augmented multilingual knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 335–354, 2024d.

Zheng Wang, Kaixuan Zhang, Wanfang Chen, Jingwen Zhang, and Xiaonan Lu. The labyrinth and the thread: Rethinking regularizations in sequential knowledge editing for large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=EsHKqJJ2u1.

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.

Hao-Xiang Xu, Ziqi Peng, Jun-Yu Ma, Yuhao Sun, Zhen-Hua Ling, and Jia-Chen Gu. Tanglescore: Tangle-guided purge and imprint for unstructured knowledge editing. In *The Fourteenth International Conference on Learning Representations*, 2026.

Wanli Yang, Fei Sun, Rui Tang, Hongyu Zang, Du Su, Qi Cao, Jingang Wang, Huawei Shen, and Xueqi Cheng. Fine-tuning done right in model editing. *arXiv preprint arXiv:2509.22072*, 2025.

Paul Youssef, Zhixue Zhao, Daniel Braun, Jörg Schlötterer, and Christin Seifert. Position: Editing large language models poses serious safety risks. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

Paul Youssef, Zhixue Zhao, Christin Seifert, and Jörg Schlötterer. Tracing and reversing edits in llms. In *The Fourteenth International Conference on Learning Representations*, 2026.

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.

Zeping Yu and Sophia Ananiadou. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280, 2024.

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.

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.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. Locate-then-edit for multi-hop factual recall under knowledge editing. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kAWtGZIHzm.

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.

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

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.

## A Unified Batch Editor

Given the batch editor

$$\Delta_j^* = \lambda_1 \mathbf{R}\mathbf{K}^\top \left(\lambda_1 \mathbf{K}\mathbf{K}^\top + \lambda_2 \mathbf{K}_0\mathbf{K}_0^\top\right)^{-1}, \tag{6}$$

We analyze the limiting case when $\lambda_1 \to \infty$ and $\lambda_2 \to \infty$.

**Limiting Case 1:** $\lambda_1 \to \infty$ Factor out $\lambda_1$ from the inverse:

$$\Delta_j^* = \mathbf{R}\mathbf{K}^\top \left(\mathbf{K}\mathbf{K}^\top + \frac{\lambda_2}{\lambda_1}\mathbf{M}\right)^{-1}.$$

Now apply the matrix identity (a special case of the Woodbury formula):

$$\mathbf{K}^\top \left(\mathbf{K}\mathbf{K}^\top + \epsilon\mathbf{M}\right)^{-1} = \left(\mathbf{K}^\top\mathbf{M}^{-1}\mathbf{K} + \epsilon\mathbf{I}\right)^{-1}\mathbf{K}^\top\mathbf{M}^{-1}, \quad \epsilon > 0.$$

Setting $\epsilon = \lambda_2/\lambda_1$ , we obtain

$$\Delta_j^* = \mathbf{R}\left(\mathbf{K}^\top\mathbf{M}^{-1}\mathbf{K} + \frac{\lambda_2}{\lambda_1}\mathbf{I}\right)^{-1}\mathbf{K}^\top\mathbf{M}^{-1}.$$

Taking the limit $\lambda_1 \to \infty$ (with $\lambda_2 > 0$ fixed) gives $\frac{\lambda_2}{\lambda_1} \to 0$. Hence,

$$\lim_{\lambda_1\to\infty} \Delta_j^* = \mathbf{R}\left(\mathbf{K}^\top(\mathbf{K}_0\mathbf{K}_0^\top)^{-1}\mathbf{K}\right)^{-1}\mathbf{K}^\top(\mathbf{K}_0\mathbf{K}_0^\top)^{-1},$$

which is exactly the update rule of ROME [Meng et al., 2022] and EMMET [Gupta et al., 2024c].

**Limiting Case 2:** $\lambda_2 \to \infty$ To analyze the limiting behavior of the unified batch editor as $\lambda_2 \to \infty$, we first introduce the following matrix-analytic result [Ben-Israel and Greville, 2003]:

Let $\mathbf{A} \in \mathbb{R}^{d\times d}$ be symmetric positive semidefinite, and let $\mathbf{K}_0 \in \mathbb{R}^{d\times m_0}$ be full column rank. Define the orthogonal projector onto the null space of $\mathbf{K}_0^\top$ as

$$\mathbf{P} := \mathbf{I} - \mathbf{K}_0(\mathbf{K}_0^\top\mathbf{K}_0)^{-1}\mathbf{K}_0^\top.$$

Then

$$\lim_{\lambda_2\to\infty}\left(\mathbf{A} + \lambda_2\,\mathbf{K}_0\mathbf{K}_0^\top\right)^{-1} = \mathbf{P}\left(\mathbf{P}\mathbf{A}\mathbf{P}\right)^\dagger\mathbf{P},$$

where † denotes the Moore–Penrose pseudoinverse.

In our setting, set $\mathbf{A} = \lambda_1\mathbf{K}\mathbf{K}^\top$ , which is symmetric positive semidefinite. Applying Lemma A, we obtain

$$\lim_{\lambda_2\to\infty}\left(\lambda_1\mathbf{K}\mathbf{K}^\top + \lambda_2\mathbf{K}_0\mathbf{K}_0^\top\right)^{-1} = \frac{1}{\lambda_1}\mathbf{P}\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P}\right)^\dagger\mathbf{P}. \tag{7}$$

Substituting Eq. (7) into the expression for $\Delta_j^*$ in Eq. (6) yields

$$\begin{aligned}\lim_{\lambda_2\to\infty}\Delta_j^* &= \lambda_1\mathbf{R}\mathbf{K}^\top\cdot\frac{1}{\lambda_1}\mathbf{P}\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P}\right)^\dagger\mathbf{P}\\ &= \mathbf{R}\mathbf{K}^\top\mathbf{P}\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P}\right)^\dagger\mathbf{P} \qquad (8)\\ &= \mathbf{R}\mathbf{K}^\top\mathbf{P}\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P}\right)^\dagger. \qquad (9)\end{aligned}$$

where the last equality follows because the range of $\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P}\right)^\dagger$ is contained in $\mathrm{col}(\mathbf{P})$ , and $\mathbf{P}$ acts as the identity on its own column space.

In practice, to ensure numerical stability and invertibility, a small regularization term $\mu\mathbf{I}$ ( $\mu > 0$ ) is added, yielding the AlphaEdit-like update rule [Fang et al., 2025a]:

$$\lim_{\lambda_2\to\infty}\Delta_j^* = \mathbf{R}\mathbf{K}^\top\mathbf{P}\left(\mathbf{P}\mathbf{K}\mathbf{K}^\top\mathbf{P} + \mu\mathbf{I}\right)^{-1}. \tag{10}$$

## B Derivation of Batch-Equivalent Sequential Editing

It has long been believed in the model editing community that sequential *Locate-then-Edit* (LTE) is inferior to batch LTE. However, we show below that they are in fact exactly equivalent—no assumptions are introduced in establishing the equivalence between sequential LTE and batch LTE.

### B.1 Derivation of Batch-Equivalent Sequential Editing for MEMIT (Theorem 4.1)

We begin with the batch solution for $j$ edits:

$$\Delta_j = B_j A_j^{-1},$$

where

$$A_j = \lambda_1 \sum_{i=1}^{j} k_i k_i^\top + \lambda_2 K_0 K_0^\top, \qquad B_j = \lambda_1 \sum_{i=1}^{j} r_i k_i^\top.$$

Note that $B_j = \Delta_j A_j$, a relation we will use later.

**Step 1: Initial equivalence.** At the first step $j = 1$, the sequential solution $\Delta_1'$ is computed directly from the first edit $(k_1, r_1)$:

$$\Delta_1' = \lambda_1 r_1 k_1^\top (\lambda_1 k_1 k_1^\top + \lambda_2 K_0 K_0^\top)^{-1}$$

which is trivially the batch solution $\Delta_1 = B_1 A_1^{-1}$.

**Step 2: Adding the $(j+1)$-th edit.** For the $(j+1)$-th edit, the matrices update as:

$$A_{j+1} = A_j + \lambda_1 k_{j+1} k_{j+1}^\top, \qquad B_{j+1} = B_j + \lambda_1 r_{j+1} k_{j+1}^\top.$$

The batch solution after $j+1$ edits is therefore:

$$\Delta_{j+1} = B_{j+1} A_{j+1}^{-1} = \left(B_j + \lambda_1 r_{j+1} k_{j+1}^\top\right) A_{j+1}^{-1}.$$

Substituting $B_j = \Delta_j A_j$, we obtain:

$$\Delta_{j+1} = \left(\Delta_j A_j + \lambda_1 r_{j+1} k_{j+1}^\top\right) A_{j+1}^{-1}. \tag{11}$$

**Step 3: Applying the Sherman–Morrison–Woodbury formula.** The SMW formula states:

$$(A_j + \lambda_1 k_{j+1} k_{j+1}^\top)^{-1} = A_j^{-1} - \frac{\lambda_1 A_j^{-1} k_{j+1} k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}}.$$

Applying this to Eq.11 and simplifying:

$$\begin{aligned}
\Delta_{j+1} &= \left(\Delta_j A_j + \lambda_1 r_{j+1} k_{j+1}^\top\right) \left(A_j^{-1} - \frac{\lambda_1 A_j^{-1} k_{j+1} k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}}\right) \\
&= \Delta_j + \lambda_1 (r_{j+1} - \Delta_j k_{j+1}) \underbrace{\frac{k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}}}_{\text{Term } X}.
\end{aligned} \tag{12}$$

**Step 4: Simplifying Term $X$.** Observe that Term $X$ is actually equal to $k_{j+1}^\top A_{j+1}^{-1}$:

$$\begin{aligned}
k_{j+1}^\top A_{j+1}^{-1} &= k_{j+1}^\top \left(A_j^{-1} - \frac{\lambda_1 A_j^{-1} k_{j+1} k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}}\right) \\
&= k_{j+1}^\top A_j^{-1} - \frac{\lambda_1 (k_{j+1}^\top A_j^{-1} k_{j+1}) k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}} \\
&= \frac{k_{j+1}^\top A_j^{-1}}{1 + \lambda_1 k_{j+1}^\top A_j^{-1} k_{j+1}}.
\end{aligned}$$

Thus Term $X = k_{j+1}^\top A_{j+1}^{-1}$.

**Step 5: The sequential update rule.** Substituting this identity into Eq. 12, we obtain:

$$\Delta_{j+1} = \Delta_j + \lambda_1 (r_{j+1} - \Delta_j k_{j+1}) k_{j+1}^\top A_{j+1}^{-1}. \tag{13}$$

Recalling $\mathbf{W}_{j+1} = \mathbf{W}_0 + \Delta_{j+1}$ and $\mathbf{W}_j = \mathbf{W}_0 + \Delta_j$, we convert to weight-space update:

$$\mathbf{W}_{j+1}^* = \mathbf{W}_j^* + \lambda_1 \left(r_{j+1} - (\mathbf{W}_j^* - \mathbf{W}_0) k_{j+1}\right) k_{j+1}^\top A_{j+1}^{-1}.$$

Rearranging yields the final form:

$$\mathbf{W}_{j+1}^* = \mathbf{W}_j^* \left(I - \lambda_1 k_{j+1} k_{j+1}^\top A_{j+1}^{-1}\right) + \lambda_1 \left(r_{j+1} + \mathbf{W}_0 k_{j+1}\right) k_{j+1}^\top A_{j+1}^{-1}. \tag{14}$$

This update rule defines the sequential operator $f^*$. Starting from $j = 1$, it recursively computes $\mathbf{W}_{j+1}^*$ from $\mathbf{W}_j^*$ while exactly matching the batch solution at every step.

### B.2 Derivation of Batch-Equivalent Sequential Editing for AlphaEdit

We begin with the batch solution for $j$ edits:

$$\Delta_j = B_j A_j^{-1},$$

where

$$A_j = \left(\sum_{i=1}^{j} k_i k_i^\top\right) P + \mu I, \qquad B_j = \left(\sum_{i=1}^{j} r_i k_i^\top\right) P.$$

Note that $B_j = \Delta_j A_j$, a relation we will use later.

**Step 1: Initial equivalence.** At the first step $j = 1$, the sequential solution $\Delta_1'$ is computed directly from the first edit $(k_1, r_1)$:

$$\Delta_1' = r_1 k_1^\top P (k_1 k_1^\top P + \mu I)^{-1}$$

which is trivially the batch solution $\Delta_1 = B_1 A_1^{-1}$.

**Step 2: Adding the $(j+1)$-th edit.** For the $(j+1)$-th edit, the matrices update as:

$$A_{j+1} = A_j + k_{j+1} k_{j+1}^\top P, \qquad B_{j+1} = B_j + r_{j+1} k_{j+1}^\top P.$$

The batch solution after $j+1$ edits is therefore:

$$\Delta_{j+1} = B_{j+1} A_{j+1}^{-1} = \left(B_j + r_{j+1} k_{j+1}^\top P\right) A_{j+1}^{-1}.$$

Substituting $B_j = \Delta_j A_j$, we obtain:

$$\Delta_{j+1} = \left(\Delta_j A_j + r_{j+1} k_{j+1}^\top P\right) A_{j+1}^{-1}. \tag{15}$$

**Step 3: Applying the Sherman–Morrison–Woodbury formula.** The SMW formula states:

$$(A_j + k_{j+1} k_{j+1}^\top P)^{-1} = A_j^{-1} - \frac{A_j^{-1} k_{j+1} k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}}.$$

Applying this to Eq.15 and simplifying:

$$\begin{aligned}
\Delta_{j+1} &= \left(\Delta_j A_j + r_{j+1} k_{j+1}^\top P\right) \left(A_j^{-1} - \frac{A_j^{-1} k_{j+1} k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}}\right) \\
&= \Delta_j + (r_{j+1} - \Delta_j k_{j+1}) \underbrace{\frac{k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}}}_{\text{Term } X}.
\end{aligned} \tag{16}$$

**Step 4: Simplifying Term $X$.** Observe that Term $X$ is actually equal to $k_{j+1}^\top P A_{j+1}^{-1}$:

$$\begin{aligned}
k_{j+1}^\top P A_{j+1}^{-1} &= k_{j+1}^\top P \left(A_j^{-1} - \frac{A_j^{-1} k_{j+1} k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}}\right) \\
&= k_{j+1}^\top P A_j^{-1} - \frac{(k_{j+1}^\top P A_j^{-1} k_{j+1}) k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}} \\
&= \frac{k_{j+1}^\top P A_j^{-1}}{1 + k_{j+1}^\top P A_j^{-1} k_{j+1}}.
\end{aligned}$$

Thus Term $X = k_{j+1}^\top P A_{j+1}^{-1}$.

**Step 5: The sequential update rule.** Substituting this identity into Eq. 16, we obtain:

$$\Delta_{j+1} = \Delta_j + (r_{j+1} - \Delta_j k_{j+1}) k_{j+1}^\top P A_{j+1}^{-1}.$$

Recalling $\mathbf{W}_j = \mathbf{W}_0 + \Delta_j$ and $\mathbf{W}_{j+1} = \mathbf{W}_0 + \Delta_{j+1}$, we convert to weight-space update:

$$\mathbf{W}_{j+1} = \mathbf{W}_j + (r_{j+1} - (\mathbf{W}_j - \mathbf{W}_0) k_{j+1}) k_{j+1}^\top P A_{j+1}^{-1}.$$

Rearranging yields the final form:

$$\mathbf{W}_{j+1} = \mathbf{W}_j \left(I - k_{j+1} k_{j+1}^\top P A_{j+1}^{-1}\right) + (r_{j+1} + \mathbf{W}_0 k_{j+1}) k_{j+1}^\top P A_{j+1}^{-1}.$$

This update rule defines the sequential operator for AlphaEdit. Starting from $j = 1$, it recursively computes $\mathbf{W}_{j+1}$ from $\mathbf{W}_j$ while exactly matching the batch solution at every step.

**Algorithm 1** Batch-Equivalent Sequential Editing (MEMIT)

**Require:** Initial weight $\mathbf{W}_0$ and pretrained knowledge key covariance matrix $\mathbf{C}_0 = \mathbf{K}_0\mathbf{K}_0^\top$.
**Ensure:** Updated weights $\mathbf{W}_j$ after each edit

Compute $A_0^{-1} \leftarrow (\lambda_2 \mathbf{C}_0)^{-1}$
$\Delta_0 \leftarrow \mathbf{0}$
**for** $j = 1$ to $t$ **do**
  Recieve new fact $(s_i, r_i, o_i)$
  Compute key-value pairs $(k_j, r_j)$ based on $\mathbf{W}_0$
  Update inverse via SMW:

$$A_j^{-1} \leftarrow A_{j-1}^{-1} - \frac{A_{j-1}^{-1} k_j k_j^\top A_{j-1}^{-1}}{1/\lambda_1 + k_j^\top A_{j-1}^{-1} k_j}$$

  Compute update matrix:

$$\Delta_j \leftarrow \Delta_{j-1} + \lambda_1 (r_j - \Delta_{j-1} k_j) k_j^\top A_j^{-1}$$

  Set $\mathbf{W}_j \leftarrow \mathbf{W}_0 + \Delta_j$
**end for**

### B.3 Unified Sequential Editor

We demonstrate that history-agnostic (Section 5), history-regularized (Section 5), and batch-equivalent sequential editing (Section 4) can be unified into a single formulation. Without loss of generality, we take MEMIT as an example. Specifically, to generalize the sequential update (Eq. 14), we augment it with two interpolation parameters: $\beta_1$ controls the protection of editing history, while $\beta_2$ controls batch-equivalence.

$$\mathbf{W}_{j+1}^* = \mathbf{W}_j^* \left(I - \beta_2 \lambda_1 k_{j+1} k_{j+1}^\top A_{j+1}^{-1}\right) + \lambda_1 \left(r_{j+1} + \beta_2 \mathbf{W}_0 k_{j+1}\right) k_{j+1}^\top A_{j+1}^{-1}, \tag{17}$$

where

$$A_j = k_j k_j^\top + \beta_1 \sum_{i=1}^{j-1} k_i k_i^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top, \qquad B_j = \sum_{i=1}^{j} r_i k_i^\top.$$

Setting $\beta_1 = \beta_2 = 1$ recovers the batch-equivalent sequential editor in Eq. 14. More generally, the unified formulation can degenerate to existing editors by varying $\beta_1, \beta_2 \in [0, 1]$. For simplicity, we follow MEMIT's default setting of $\lambda_1 = 1$ and rewrite Eq. 17 from the perspective of weight perturbation (Eq. 13):

$$\Delta_{j+1} = \Delta_j + (r_{j+1} - \beta_2 \Delta_j k_{j+1}) k_{j+1}^\top A_{j+1}^{-1}. \tag{18}$$

**History-Regularized Editor ($\beta_1 = 1, \beta_2 = 0$).** Setting $\beta_2 = 0$ eliminates the third term in the expanded form of the update rule (Eq. 18), yielding:

$$\Delta_{j+1} = \Delta_j + r_{j+1} k_{j+1}^\top A_{j+1}^{-1}, \tag{19}$$

where $A_{j+1} = k_{j+1} k_{j+1}^\top + \beta_1 \sum_{i=1}^{j} k_i k_i^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top$ remains unchanged, with $\beta_1 > 0$ (typically set to 1 [Fang et al., 2025a, Liu et al., 2026a]) to preserve the full editing history.

**History-Agnostic Editor ($\beta_1 = 0, \beta_2 = 0$).** If $A_{j+1}$ ignores previous edits (i.e., setting $\beta_1 = 0$ so that $A_{j+1} = k_{j+1} k_{j+1}^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top$), the method reduces to repeatedly applying the vanilla MEMIT update in a sequential manner:

$$\Delta_{j+1} = \Delta_j + r_{j+1} k_{j+1}^\top \left(k_{j+1} k_{j+1}^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top\right)^{-1}. \tag{20}$$

This approach is effectively **history-agnostic**, as it disregards prior edits when performing new ones [Gu et al., 2024b, Thede et al., 2025].

Based on the batch-equivalent sequential editor ($\beta_1 = 1, \beta_2 = 1$), we further have the following discussions.

Table 3: Comparison of three sequential implementation strategies against the batch editor on Qwen3-8B under `float32` precision. Editing performance is reported after 2,000 and 10,000 sequential edits on COUNTERFACT and ZSRE.

| Method | CounterFact | | | | | | ZSRE | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T = 2000$ | | | $T = 10,000$ | | | $T = 2000$ | | | $T = 10,000$ | | |
| | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** |
| Batch | 97.6 | 53.1 | 10.8 | 91.3 | 49.0 | 8.8 | 94.5 | 88.0 | 38.1 | 90.9 | 84.5 | 37.2 |
| SMW | 95.3 | 52.3 | 10.2 | 81.2 | 44.5 | 7.2 | 91.5 | 87.0 | 38.6 | 83.9 | 79.5 | 37.1 |
| Direct Inverse | 96.9 | 52.8 | 10.6 | 88.2 | 46.7 | 8.1 | 94.1 | 87.5 | 38.4 | 88.1 | 82.7 | 37.7 |
| Direct Batch-Form | 97.6 | 53.1 | 10.8 | 91.3 | 49.0 | 8.8 | 94.5 | 88.0 | 38.1 | 90.9 | 84.5 | 37.2 |

Table 4: Comparison of editing time for 10,000 sequential edits on LLaMA-3-8B.

| Method | MEMIT | EMMET | AlphaEdit | RECT | PRUNE | Ours |
|---|---|---|---|---|---|---|
| Time (hours) | 16 | 15 | 18 | 16 | 28 | 9 |

**Batch MEMIT.** Setting $j = 0$ (so $\Delta_j = \mathbf{0}$) and replacing vectors with matrices recovers the **batch** MEMIT formulation:

$$\Delta = \mathbf{R}\mathbf{K}^\top \left(\mathbf{K}\mathbf{K}^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top\right)^{-1}. \tag{21}$$

**Post-Processing Interpretation.** If we approximate $r_{j+1}$ as $\bar{\Delta}_{j+1} k_{j+1}$, where $\bar{\Delta}_{j+1}$ is the update matrix from a third-party editor, the unified rule becomes:

$$\Delta_{j+1} = \Delta_j + \bar{\Delta}_{j+1} k_{j+1} k_{j+1}^\top A_{j+1}^{-1} - \Delta_j k_{j+1} k_{j+1}^\top A_{j+1}^{-1}. \tag{22}$$

This can be viewed as a post-processing technique that transforms a vanilla update $\bar{\Delta}_{j+1}$ into an update that protects previous edits and pretrained knowledge. Such an idea is adopted in *Simulating Ideal Editor* (SimIE [Guo et al., 2025b]); however, due to approximation errors and propagation across multiple layers, SimIE empirically underperforms compared to history-regularized methods like AlphaEdit [Fang et al., 2025a]. Moreover, in SimIE, $A_{j+1}$ is set to $\mathbf{K}_{[1:j+1]}\mathbf{K}_{[1:j+1]}^\top + \mathbf{I}$, which fails to preserve the pre-trained knowledge $\mathbf{K}_0$. Notably, the identity matrix $\mathbf{I}$ — rather than $\mathbf{K}_0\mathbf{K}_0^\top$ — is mandatory in the derivation of SimIE.

### B.4 Extension to Mini-Batch Sequential Editing

Suppose that at step $j$ , a mini-batch of $b$ edits arrives simultaneously, represented by key and residual matrices $\mathbf{K}_j \in \mathbb{R}^{d_{\text{in}} \times b}$ and $\mathbf{R}_j \in \mathbb{R}^{d_{\text{out}} \times b}$, respectively. The Gram and cross-covariance matrices are then updated as

$$A_{j+1} = A_j + \lambda_1 \mathbf{K}_j \mathbf{K}_j^\top, \qquad B_{j+1} = B_j + \lambda_1 \mathbf{R}_j \mathbf{K}_j^\top,$$

where $A_j = \lambda_1 \sum_{i=1}^{j-1} \mathbf{K}_i \mathbf{K}_i^\top + \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top$ and $B_j = \sum_{i=1}^{j-1} \lambda_1 \mathbf{R}_i \mathbf{K}_i^\top$ follow the definitions in Eq. (4). The inverse $A_{j+1}^{-1}$ can be computed efficiently using the Woodbury matrix identity:

$$A_{j+1}^{-1} = A_j^{-1} - A_j^{-1} \mathbf{K}_j \left(\frac{1}{\lambda_1} \mathbf{I}_b + \mathbf{K}_j^\top A_j^{-1} \mathbf{K}_j\right)^{-1} \mathbf{K}_j^\top A_j^{-1},$$

which reduces to the Sherman–Morrison formula when $b = 1$. The update for $\Delta_{j+1}$ remains structurally identical to the single-edit case:

$$\Delta_{j+1} = \Delta_j + (\mathbf{R}_j - \Delta_j \mathbf{K}_j) \left(\frac{1}{\lambda_1} \mathbf{I}_b + \mathbf{K}_j^\top A_j^{-1} \mathbf{K}_j\right)^{-1} \mathbf{K}_j^\top A_j^{-1}.$$

Thus, the core algorithm extends naturally to mini-batch sequential editing with only a modest increase in per-step computation (dominated by solving a $b \times b$ linear system). When $b \ll d_{\text{in}}$, this remains highly efficient; when $b = d_{\text{in}}$, it incurs the full $O(d_{\text{in}}^3)$ matrix inversion cost, recovering the naive implementation cost without SMW.

# C Practical Implementation of Batch Equivalent Sequential Editing

## C.1 Three Implementation Strategies

Although Theorem 4.1 establishes the equivalence between sequential and batch editing under exact arithmetic, different implementations of the sequential operator may exhibit different numerical behaviors in finite-precision arithmetic. In particular, the recursive computation of the inverse matrix and the weight matrix may introduce numerical errors that propagate across editing steps. To investigate this issue, we consider three implementation strategies, all based on the same theoretical formulation. Without loss of generality, we illustrate using the batch-equivalent sequential operator derived from the MEMIT-style editing objective.

**(1) SMW-based Recursive Implementation.** We exploit the Sherman–Morrison–Woodbury (SMW) identity to recursively update $A_{j+1}^{-1}$, and use Theorem 4.1 to recursively update the weight matrix $\mathbf{W}_{j+1}^{*}$. The resulting algorithm is presented in Algorithm 1.

**(2) Direct Inverse with Recursive Weight Updates.** We directly recompute $A_{j+1}^{-1}$ at each editing step, while still using Theorem 4.1 to recursively update the weight matrix $\mathbf{W}_{j+1}^{*}$.

**(3) Direct Batch-Form Implementation.** We incrementally maintain $A_j$ and $B_j$ using Eq. 4, and directly compute the weight matrix from the original weights at each step:

$$\mathbf{W}_{j+1}^{*} = \mathbf{W}_0 + B_{j+1}A_{j+1}^{-1}. \tag{23}$$

The three implementations differ in how the inverse matrix and weight matrix are computed: the first uses SMW to recursively update the inverse and then recursively updates the weights; the second recomputes the inverse from scratch but still recursively updates the weights; and the third directly computes the weights from the batch closed-form solution at each step. Consequently, the first two implementations may accumulate numerical errors through recursive updates, whereas the third avoids such error propagation. We compare their numerical deviations from batch editing in the following experiments.

## C.2 Numerical Comparison with Batch Editing

We compare the three implementation strategies against the batch editor to investigate their numerical behaviors under finite-precision arithmetic. Specifically, we evaluate their editing performance using the standard `float32` precision for weight-matrix computations. Table 3 reports the results after 2,000 and 10,000 sequential edits on COUNTERFACT and ZSRE. The results show that the numerical discrepancies increase with the number of sequential edits. In particular, the SMW-based implementation exhibits the largest deviations, followed by the direct-inverse implementation. The direct batch-form implementation matches the batch baseline. These findings illustrate that although all three strategies implement the same batch-equivalent sequential formulation under exact arithmetic, their practical numerical behaviors depend on how the recursive computations are carried out.

## C.3 Computational Efficiency of SMW-based Implementation

We further analyze the practical efficiency of the SMW-based implementation in terms of memory footprint and computational time, comparing it with existing methods under identical experimental conditions.

**Memory.** As shown in Alg. 1, our SMW-based implementation stores two matrices: $\mathbf{W}_0$ of size $(d_{\text{in}}, d_{\text{out}})$ and $A_j^{-1}$ of size $(d_{\text{in}}, d_{\text{in}})$. For LLaMA-3-8B ($d_{\text{in}} = 14336$, $d_{\text{out}} = 4096$) with `float32` precision, these require approximately 0.2 GB and 0.8 GB, respectively. Baseline methods store $\mathbf{C}_0 = \mathbf{K}_0\mathbf{K}_0^{\top}$ of size $(d_{\text{in}}, d_{\text{in}})$, so our additional memory cost is only the 0.2 GB required for $\mathbf{W}_0$. Editing five layers (Section 6) adds $0.2 \times 5 = 1$ GB, which is negligible compared with the 32 GB required to load LLaMA-3-8B. Note that the memory cost is independent of the number of edits.

**Time.** We exploit the SMW identity to efficiently maintain $A_j^{-1}$ across editing steps, avoiding repeated matrix inversions. We benchmark different editing methods on LLaMA-3-8B using 10,000 sequential edits from CounterFact (batch size = 1) on an NVIDIA A6000 48GB GPU. As reported in Table 4, our method completes all edits in approximately 9 hours, outperforming the existing baselines. In contrast, baseline methods require repeatedly inverting a $d_{\text{in}} \times d_{\text{in}}$ matrix at each edited layer, taking approximately 3 seconds per edit across five layers. This amounts to 8–10 hours for 10,000 edits, in addition to approximately 8 hours for extracting the $(k, v)$ pairs using `bfloat16` precision. PRUNE incurs further overhead from post-hoc regularization, resulting in a total runtime exceeding 24 hours.

# D Related Work on Stability Analysis

Four existing works have investigated the stability of model editing from different perspectives. PRUNE [Ma et al., 2025] examines the condition number of the weight matrix $W$, arguing that constraining it can help mitigate the perturbation of previously stored knowledge. However, its analysis is static and focuses on individual edits, without characterizing how the weight matrix evolves over successive editing steps.

SPHERE [Liu et al., 2026c] investigates the output perturbation of pretrained knowledge, quantified by $\Delta WV$, where $V$ denotes the pretrained knowledge matrix. It argues that preserving the uniformity of $W$ helps mitigate unintended changes to pretrained knowledge. Nevertheless, its analysis is also static and does not explicitly characterize the evolution of the weight matrix $W_t$ throughout the sequential editing process.

BetaEdit [Liu et al., 2026a] explicitly considers the dynamics of cumulative weight perturbations, $W_t - W_0$, during sequential editing. It compares history-agnostic and history-aware editing methods in terms of their accumulated perturbations. However, its analysis primarily focuses on comparing the relative perturbation magnitudes between these two classes, rather than providing a separate characterization of their general weight dynamics.

Similar to our work, HSE [Fang et al., 2025b] studies the evolution of the weight matrix $W_t$ through its Frobenius norm $\|W_t\|_F$. However, its analysis relies on strong and idealized assumptions, such as mutual orthogonality among edit keys $\{k_i\}_{i=1}^t$, which are generally difficult to satisfy in practical sequential editing scenarios. Moreover, its theoretical analysis is specific to HSE and does not characterize the stability of other sequential editing paradigms.

In contrast, our work provides a unified and systematic characterization of weight dynamics across multiple sequential editing paradigms, including history-agnostic, history-regularized, batch-equivalent, and post-hoc regularized editing. Without relying on rigid assumptions such as mutual orthogonality of edit keys, we establish distinct upper bounds on weight growth for different editing strategies, enabling a more comprehensive understanding and practical comparison of their long-term editing stability.

# E Convergence Analysis for Sequential Editing Strategies

We analyze the growth of the Frobenius norm of the weight matrix $W_t$ under three sequential update rules, assuming the structural conditions stated in the main text:

- **Boundness:** $\|k_t\|_2 \le C_k$ and $\|r_t\|_2 \le C_r$ for all $t$ ;
- **Invertibility:** The input covariance satisfies $\mathbb{E}[kk^\top] \succeq \alpha \mathbf{I}$ for some constant $\alpha > 0$ . Equivalently, the empirical Gram matrix becomes well-conditioned asymptotically: $\frac{1}{t} K_{[1:t]} K_{[1:t]}^\top \succeq \alpha \mathbf{I}$, $\frac{1}{n} K_0 K_0^\top \succeq \alpha \mathbf{I}$ as $t, n \to \infty$, where $K_{[1:t]} \in \mathbb{R}^{d_{in} \times t}$ denotes the key vector encoding the editing knowledge and $K_0 \in \mathbb{R}^{d_{in} \times n}$ denotes the key vector encoding the pretrained knowledge.

## E.1 Upper Bounds

*Proof.* **(1) History-Agnostic Sequential Editing.** Define the per-step gain magnitude as

$$g_t^{(1)} := \left\| A_t^{-1} k_t \right\|_2 = \left\| \left( k_t k_t^\top + K_0 K_0^\top \right)^{-1} k_t \right\|_2 .$$

We now derive the upper bound on $g_t^{(1)}$. Let $P_0 := K_0 K_0^\top \succ 0$ . By the Sherman–Morrison formula,

$$(k_t k_t^\top + P_0)^{-1} k_t = \frac{1}{1 + k_t^\top P_0^{-1} k_t} P_0^{-1} k_t.$$

Hence,

$$g_t^{(1)} = \frac{\|P_0^{-1} k_t\|_2}{1 + k_t^\top P_0^{-1} k_t}.$$

Let $z_t := P_0^{-1/2} k_t$. Then $\|P_0^{-1} k_t\|_2 = \|P_0^{-1/2} z_t\|_2 \le \lambda_{\min}(P_0)^{-1/2} \|z_t\|_2$, and $k_t^\top P_0^{-1} k_t = \|z_t\|_2^2$. ($\lambda_{\min}(P_0)$ is the smallest eigenvalues of $P_0$.) Therefore,

$$g_t^{(1)} \le \frac{\lambda_{\min}(P_0)^{-1/2} \|z_t\|_2}{1 + \|z_t\|_2^2}.$$

The scalar function $f(s) = s/(1 + s^2)$ for $s \ge 0$ attains its maximum value $1/2$ at $s = 1$. Thus,

$$g_t^{(1)} \le \frac{1}{2} \lambda_{\min}(P_0)^{-1/2}.$$

By the triangle inequality, the Frobenius norm of $W_t$ satisfies the universal upper bound

$$\|W_t\|_F \le \|W_0\|_F + \sum_{i=1}^{t} \left\| r_i k_i^\top A_i^{-1} \right\|_F \le \|W_0\|_F + t \cdot \frac{C_r}{2} \lambda_{\min}(P_0)^{-1/2},$$

**(2) History-Regularized Sequential Editing.** The update rule is

$$W_t = W_{t-1} + r_t k_t^\top A_t^{-1},$$

which unrolls to $W_t = W_0 + \sum_{s=1}^{t} r_s k_s^\top A_s^{-1}$. Hence,

$$\begin{aligned}
\|W_t\|_F &\le \|W_0\|_F + \sum_{s=1}^{t} \|r_s k_s^\top A_s^{-1}\|_F \\
&\le \|W_0\|_F + \sum_{s=1}^{t} \|r_s\|_2 \|k_s\|_2 \|A_s^{-1}\|_2 \\
&\le \|W_0\|_F + \sum_{s=1}^{t} \frac{C_r C_k}{s\alpha} \le \|W_0\|_F + \frac{C_r C_k}{\alpha}(1 + \log t).
\end{aligned}$$

where $\|A_s^{-1}\|_2 \le \frac{1}{s\alpha}$ as we have

$$\frac{1}{s} K_{[1:s]} K_{[1:s]}^\top \succeq \alpha I \quad \Rightarrow \quad A_s \succeq s\alpha I.$$

**(3) Batch-Equivalent Sequential Editing.** The corresponding batch solution at every editing step is,

$$W_t = W_0 + R_{[1:t]} K_{[1:t]}^\top A_t^{-1}.$$

Using submultiplicativity of the Frobenius norm,

$$\begin{aligned}
\|W_t\|_F &\le \|W_0\|_F + \|R_{[1:t]}\|_F \cdot \|K_{[1:t]}^\top A_t^{-1}\|_2 \\
&\le \|W_0\|_F + (\sqrt{t} C_r) \cdot \left( \|K_{[1:t]}\|_2 \cdot \|A_t^{-1}\|_2 \right) \\
&\le \|W_0\|_F + \sqrt{t} C_r \cdot \left( \sqrt{t} C_k \cdot \frac{1}{t\alpha} \right) = \|W_0\|_F + \frac{C_r C_k}{\alpha}.
\end{aligned}$$

**(4) Post-Hoc Regularized Editing.** We assume that the post-hoc regularization operator is non-expansive with respect to the Frobenius norm, i.e., $\|\operatorname{PR}(\Delta, \alpha)\|_F \le \|\Delta\|_F$ for any matrix $\Delta$. This is a mild and well-justified assumption: the primary objective of post-hoc regularization is to enhance locality and mitigate interference, which is typically achieved by suppressing redundant components or projecting the perturbation onto a constrained subspace. Consequently, such operations naturally exhibit non-expansive behavior. This assumption holds for a wide range of prevalent regularizers,

including RECT [Gu et al., 2024b], PRUNE [Ma et al., 2025], AdaEdit [Li and Chu, 2025], and SPHERE [Liu et al., 2026c].

For history-agnostic and history-regularized editors, post-hoc regularization is applied to each incremental update. By the triangle inequality and the non-expansiveness assumption, we have

$$\|\mathbf{W}_t^{\mathrm{PR}}\|_F \le \|\mathbf{W}_0\|_F + \sum_{s=1}^{t} \|\Delta_s^{\mathrm{base}}\|_F.$$

For history-agnostic editing, the per-step update satisfies

$$\|\Delta_s^{\mathrm{HA}}\|_F \le \frac{C_r}{2\sqrt{\lambda_{\min}}},$$

which gives

$$\|\mathbf{W}_t^{\mathrm{PR-HA}}\|_F \le \|\mathbf{W}_0\|_F + \frac{C_r}{2\sqrt{\lambda_{\min}}} t.$$

For history-regularized editing, using $\|A_s^{-1}\|_2 \le 1/(s\alpha_0)$, we obtain

$$\|\Delta_s^{\mathrm{HR}}\|_F \le \frac{C_r C_k}{s\alpha_0}.$$

Consequently,

$$\|\mathbf{W}_t^{\mathrm{PR-HR}}\|_F \le \|\mathbf{W}_0\|_F + \frac{C_r C_k}{\alpha_0}(1 + \log t).$$

For batch-equivalent editing, the post-hoc regularization is applied to the batch-equivalent solution:

$$\mathbf{W}_t^{\mathrm{PR-BE}} = \mathbf{W}_0 + \mathrm{PR}\left(\mathbf{W}_t^{\mathrm{BE}} - \mathbf{W}_0, \alpha\right).$$

By the non-expansiveness assumption, it follows that

$$\begin{aligned}
\|\mathbf{W}_t^{\mathrm{PR-BE}}\|_F &\le \|\mathbf{W}_0\|_F + \|\,\mathrm{PR}(\mathbf{W}_t^{\mathrm{BE}} - \mathbf{W}_0, \alpha)\|_F \\
&\le \|\mathbf{W}_0\|_F + \|\mathbf{W}_t^{\mathrm{BE}} - \mathbf{W}_0\|_F \\
&\le \|\mathbf{W}_0\|_F + \frac{C_r C_k}{\alpha_0}.
\end{aligned}$$

Therefore, post-hoc regularization preserves the established weight growth upper bounds of all three base editing methods, without changing their respective growth rates.

This completes the proof. □

### E.2 Tightness of the Upper Bounds

Consider the scalar setting with:

- $K_0 = \sqrt{\lambda}$ for some $\lambda > 0$ , so $K_0 K_0^\top = \lambda > 0$ (we set $\lambda = 1$);
- Edit inputs $k_t = 1$ for all $t$ (hence $c_k = C_k = 1$ );
- Fixed residual magnitude $|r^\star| = C_r > 0$;

Then the history-agnostic update becomes

$$W_t = W_{t-1} + \frac{r^\star}{1+\lambda},$$

which yields

$$W_t = W_0 + |r^\star| \sum_{s=1}^{t} \frac{1}{1+\lambda} = W_0 + t \cdot \frac{C_r}{2} \lambda_{\min}(P_0)^{-1/2}.$$

Here, we omit the symbol $\|\|$ for $W_t$ and $W_0$ as the weight at every editing step is a constant.

Similarly, the history-regularized update becomes

$$W_t = W_{t-1} + \frac{r^\star}{t+\lambda},$$

which yields

$$W_t = W_0 + |r^\star| \sum_{s=1}^{t} \frac{1}{s+\lambda} = W_0 + \frac{C_r C_k}{\alpha}(1 + \log t)(t \to \infty).$$

And the batch-equivalent sequential update becomes

$$W_t = W_0 + t \cdot \frac{r^\star}{t+\lambda} = W_0 + \frac{C_r C_k}{\alpha}(t \to \infty),$$

This verifies the tightness of the upper bounds.

Finally, the upper bounds for post-hoc regularized editing are also tight. This follows from the fact that $\mathrm{PR}(\Delta, 0) = \Delta$, under which post-hoc regularization reduces to the corresponding base editor. Since the upper bounds of the three base editors are tight, the upper bounds established for their post-hoc regularized variants are tight as well.

# F Additional Experimental Setup

## F.1 Evaluation Metrics

We evaluate editing performance using three standard metrics [Liu et al., 2026c]. Let $f_\theta$ denote the original model and $f_{\theta^*}$ denote the model after sequential editing. For each edit instance, we are given:

- The input prompt $x$ (e.g., "The president of the United States is"),
- The target output $y_{\text{target}}$ (e.g., "Joe Biden"),
- A set of paraphrased prompts $\mathcal{P}(x)$ that preserve the same meaning as $x$,
- A set of unrelated prompts $\mathcal{U}$ that are semantically independent of the edited fact.

Let $p_\theta(\cdot \mid x)$ denote the probability distribution over the output vocabulary produced by the model. The three metrics are formally defined as follows:

**Success Rate (Suc.)** measures whether the edited model assigns the highest probability to the target output $y_{\text{target}}$ for the original input $x$:

$$\text{Suc} = \mathbb{E}_{(x, y_{\text{target}})} \left[ \mathbf{1} \left( \arg\max_y p_{\theta^*}(y \mid x) = y_{\text{target}} \right) \right],$$

where the expectation is taken over all edit instances in the test set.

**Generality (Gen.)** measures whether the edited model generalizes the edited fact to paraphrased inputs:

$$\text{Gen} = \mathbb{E}_{(x, y_{\text{target}})} \left[ \frac{1}{|\mathcal{P}(x)|} \sum_{x' \in \mathcal{P}(x)} \mathbf{1} \left( \arg\max_y p_{\theta^*}(y \mid x') = y_{\text{target}} \right) \right].$$

**Locality (Loc.)** measures whether the editing process preserves the model's behavior on unrelated inputs. A good editor should maintain the original output distribution for unrelated prompts, i.e., the probability of the original output $y_{\text{original}}$ should remain unchanged from before editing. The metric is defined as:

$$\text{Loc} = \mathbb{E}_{(x_{\text{unrelated}}, y_{\text{original}})} \left[ \mathbf{1} \left( \arg\max_y p_{\theta^*}(y \mid x_{\text{unrelated}}) = y_{\text{original}} \right) \right],$$

A desirable editor should not degrade the model's performance on unrelated facts.

### F.2 Update Rules of Sequential LTE Baselines

Below, we detail the update rules for the sequential LTE methods used as baselines in our study. For each method, we provide three variants: a history-agnostic (HA) form, which does not explicitly account for previous edits; a history-regularized (HR) form, which extends the regularization term to protect previously edited knowledge [Fang et al., 2025a, Guo et al., 2025b, Fang et al., 2025b]; and a batch-equivalent form, which produces the batch solution at every editing step.

**MEMIT-HA (History-Agnostic).** The update rule is:

$$W_t = W_{t-1} + r_t k_t^\top \left(\lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top + k_t k_t^\top\right)^{-1}. \tag{24}$$

**MEMIT-HR (History-Regularized).** To protect previously edited knowledge, we extend the regularization term as follows:

$$W_t = W_{t-1} + r_t k_t^\top \left(\lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top\right)^{-1}. \tag{25}$$

**MEMIT-BE (Batch-Equivalent).** This method is introduced in Appendix B. The update rule is:

$$W_t = W_{t-1}\left(\mathbf{I} - \lambda_1 k_t k_t^\top A_t^{-1}\right) + \lambda_1 \left(r_t + W_0 k_t\right) k_t^\top A_t^{-1}, \tag{26}$$

$$A_t = \lambda_2 \mathbf{K}_0 \mathbf{K}_0^\top + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top, \tag{27}$$

where we follow MEMIT's default settings with $\lambda_1 = 1$ and $\lambda_2 = 15000$ for fair comparison.

**EMMET-HA (History-Agnostic).** The update rule is:

$$W_t = W_{t-1} + r_t \left(k_t^\top (\mathbf{K}_0 \mathbf{K}_0^\top)^{-1} k_t + \mu \mathbf{I}\right)^{-1} k_t^\top (\mathbf{K}_0 \mathbf{K}_0^\top)^{-1}. \tag{28}$$

**EMMET-HR (History-Regularized).** We adapt the rule by updating the inverse covariance estimate to include prior edits:

$$W_t = W_{t-1} + r_t \left(k_t^\top (\mathbf{K}_0 \mathbf{K}_0^\top + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top)^{-1} k_t + \mu \mathbf{I}\right)^{-1} k_t^\top (\mathbf{K}_0 \mathbf{K}_0^\top + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top)^{-1}. \tag{29}$$

**EMMET-BE (Batch-Equivalent).** Following the procedure in Appendix B, we derive the batch-equivalent sequential editing algorithm for EMMET:

$$W_t = W_{t-1} + \left(r_t - (W_{t-1} - W_0) k_t\right) k_t^\top C_t^{-1}, \tag{30}$$

where $C_t = \mu \mathbf{K}_0 \mathbf{K}_0^\top + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top$. This update is derived by reformulating the EMMET batch solution using the matrix inversion identity, which transforms the regularization term $\mu \mathbf{I}$ into $\mu \mathbf{K}_0 \mathbf{K}_0^\top$. We follow EMMET's default setting with $\mu = 0.1$ for a fair comparison.

**AlphaEdit-HA (History-Agnostic).** The update rule is:

$$W_t = W_{t-1} + r_t k_t^\top \mathbf{P} \left(\lambda_2 \mathbf{I} + k_t k_t^\top \mathbf{P}\right)^{-1}, \tag{31}$$

where $\mathbf{P}$ is a projection matrix onto the null space of preserved knowledge.

**AlphaEdit-HR (History-Regularized).** To extend protection to previously edited knowledge, we include all past keys in the regularization term:

$$W_t = W_{t-1} + r_t k_t^\top \mathbf{P} \left(\lambda_2 \mathbf{I} + \mathbf{K}_{[1:t-1]} \mathbf{K}_{[1:t-1]}^\top \mathbf{P} + k_t k_t^\top \mathbf{P}\right)^{-1}. \tag{32}$$

**AlphaEdit-BE (Batch-Equivalent).** This method is introduced in Appendix B. The update rule is:

$$\mathbf{W}_t = \mathbf{W}_{t-1}\left(\mathbf{I} - k_t k_t^\top \mathbf{P} A_t^{-1}\right) + \left(r_t + \mathbf{W}_0 k_t\right) k_t^\top \mathbf{P} A_t^{-1}, \tag{33}$$

$$A_t = \lambda_2 \mathbf{I} + \mathbf{K}_{[1:t]} \mathbf{K}_{[1:t]}^\top \mathbf{P}, \tag{34}$$

where we follow AlphaEdit's default settings with $\lambda_2 = 10$ for fair comparison.

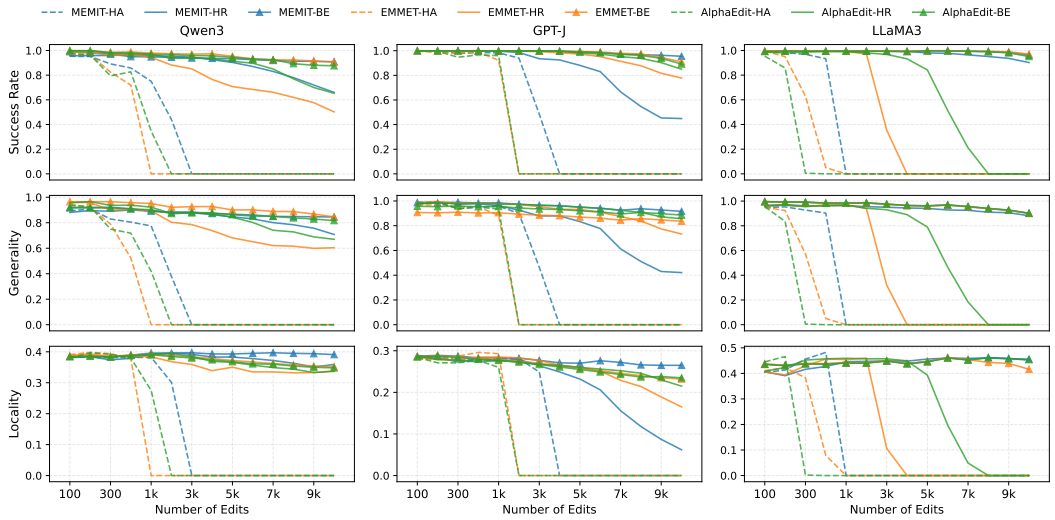



Figure 7: Editing performance (success rate, generality, locality) on the ZsRE dataset over 10,000 sequential edits. Columns (from left to right): Qwen3-8B, GPT-J, LLaMA3-8B-Instruct. Higher values indicate better performance for all metrics.

# G Additional Experimental Results

## G.1 Additional Editing Results on ZsRE

Figure 7 presents the editing performance of the three models on the ZsRE dataset over 10,000 sequential edits. Columns from left to right: Qwen3-8B, GPT-J, LLaMA3-8B-Instruct. Higher values indicate better performance for all metrics.

The results on ZsRE mirror the hierarchical performance pattern observed on CounterFact. First, history-regularized variants consistently outperform their history-agnostic counterparts, maintaining reliable performance over thousands of sequential edits. Second, our proposed batch-equivalent sequential editing method again achieves the best long-term performance, sustaining high accuracy across all three metrics even after tens of thousands of edits. These findings confirm that the observed performance hierarchy is robust across different factual editing benchmarks.

## G.2 Additional Stability Results

**Step-Per-Step Stability Results** Figures 3 and 4 in the main paper report language understanding and parameter stability after subsampling every 400 steps to improve readability. Here we provide the complete step-by-step evolution of the weight norm.

Figures 8 and 9 show the results on the CounterFact and ZsRE datasets, respectively, for three editing methods (EMMET-HA, ALPHAEDIT-HA, MEMIT-HA) using GPT2-XL, GPT-J and LLaMA3. As can be seen, the weight norm does not increase monotonically; it can drop substantially from one step to the next and exhibits strong oscillations, especially for GPT-J. These oscillations motivate the subsampling strategy used in the main paper to demonstrate the overall trend of norm growth.

**Additional Stability Results on ZsRE** Figures 10 and 11 present the language understanding and parameter stability results on the ZsRE dataset, which mirror the hierarchical patterns observed on CounterFact (Figures 3 and 4). History-agnostic editors exhibit rapid parameter drift and collapse of language understanding within a few hundred edits. History-regularized editors mitigate this drift but still struggle beyond thousands of edits. Batch-equivalent sequential editors achieve the highest stability, with $\|\mathbf{W}_t\|_F$ remaining nearly constant and SST-2 accuracy staying close to its initial level over 10,000 edits.

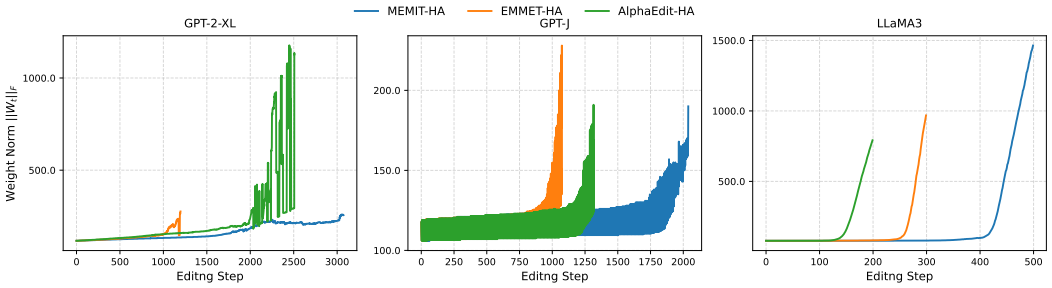



Figure 8: Step-by-step evolution of the weight norm for GPT2-XL, GPT-J and LLaMA3 during sequential editing on the COUNTERFACT dataset.

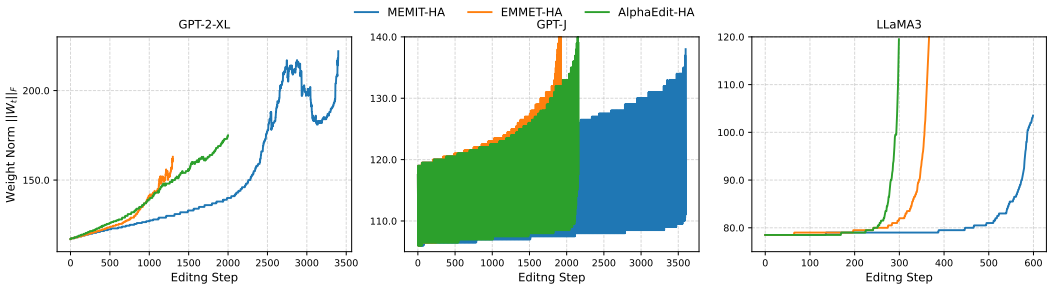



Figure 9: Step-by-step evolution of the weight norm for GPT2-XL, GPT-J and LLaMA3 during sequential editing on the ZSRE dataset.

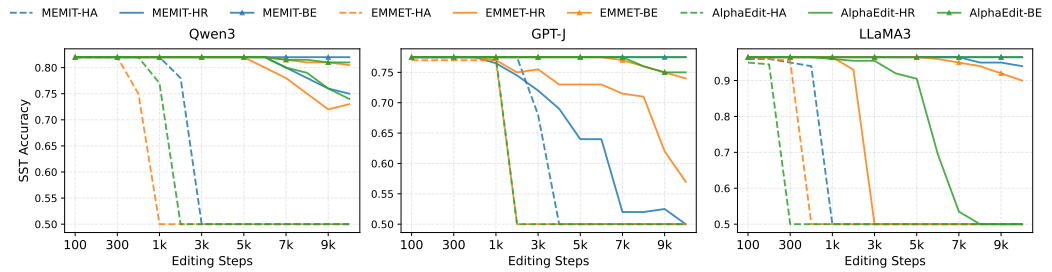



Figure 10: Evolution of language understanding capability (SST accuracy) during 10,000 sequential edits on the ZSRE dataset.

### G.3 Additional Stability Metrics during Long-Term Editing

Beyond tracking the Frobenius norm of the updated weight $\|W_t\|_F$, we also evaluate two additional stability metrics:

- **Weight Perturbation:** $\|\Delta_t\|_F = \|W_t - W_0\|_F$, measuring the total deviation from the original model.
- **Pretrained Knowledge Leakage:** $\|(W_t - W_0)\mathbf{K}_0\|_F$, quantifying how much the editing process affects preserved pretrained knowledge.

As shown in Figures 12 and 13, the plots (from left to right for GPT2-XL, GPT-J, and LLaMA3) confirm the same stability hierarchy across all models and datasets. The batch-equivalent sequential editor (MEMIT-BE) maintains the smallest weight perturbation and knowledge leakage, indicating the highest stability. In contrast, history-regularized and history-agnostic methods exhibit varying degrees of unbounded growth in these metrics over sequential edits.

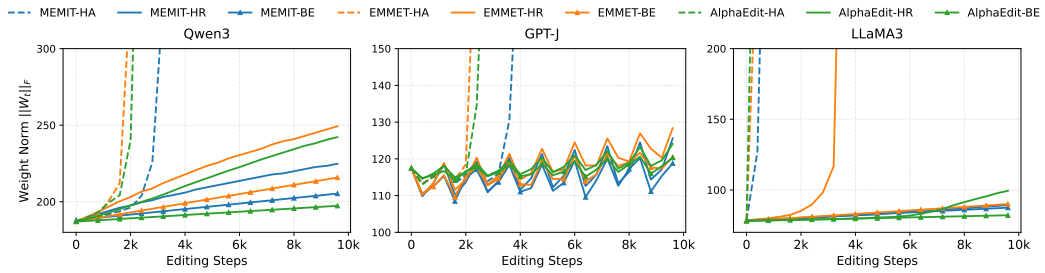



Figure 11: Evolution of the weight norm during 10,000 sequential edits on the ZsRE dataset.

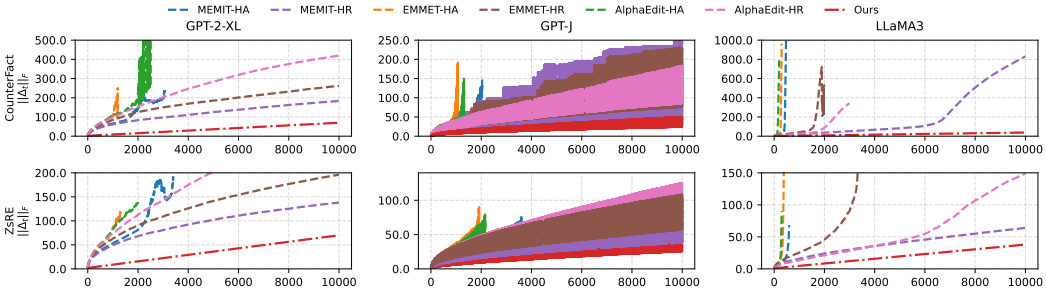



Figure 12: Step-by-step evolution of the weight perturbation for GPT2-XL, GPT-J and LLaMA3 during 10,000 sequential edits.

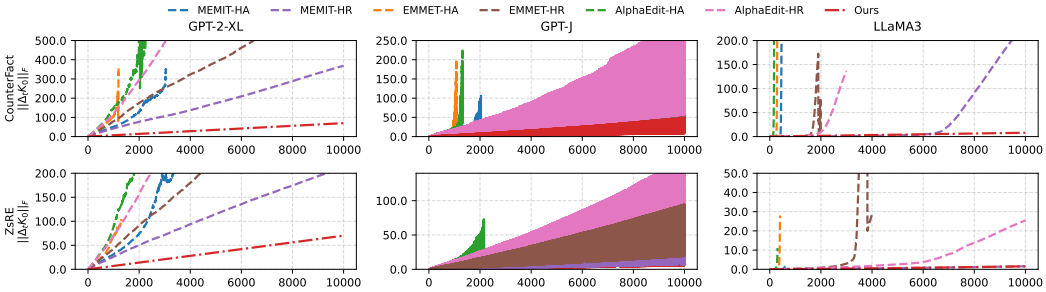



Figure 13: Step-by-step evolution of the pretrained knowledge leakage for GPT2-XL, GPT-J and LLaMA3 during 10,000 sequential edits.

### G.4 Additional Interpolation Results on ZsRE

Figure 14 presents the interpolation results on the ZsRE dataset, mirroring the pattern observed on CounterFact (Figure 5). As $\beta_1$ increases from 0 to 0.5, the editing behavior smoothly transitions from history-agnostic to history-regularized, with corresponding improvements in long-term stability and performance (see MEMIT-BE(0.5,0)).

### G.5 Additional Results for Post-Hoc Regularized Editing

**Details about $\alpha$** The regularization strength $\alpha \in [0, 1]$ controls the proportion of singular vectors corresponding to small singular values retained from the singular value decomposition of $W^\top W$, where $W$ is the weight matrix. Specifically, we decompose $W^\top W$ and sort the singular vectors in ascending order of their associated singular values (i.e., from smallest to largest). We then take the first $\alpha$ fraction of the singular vectors to form the matrix $U$. The projection matrix is constructed as $P = I - UU^\top$. When $\alpha = 0$, no singular vectors are retained ($U$ is empty), so $P = I$ and the original update is unchanged. When $\alpha = 1$, all singular vectors are retained, yielding a projection that maximally preserves locality. More details are referred to SPHERE [Liu et al., 2026c].

**Effect of post-hoc regularization on AlphaEdit-HR** In the main paper, Figure 6 demonstrates the effect of post-hoc regularization strategies (RECT, PRUNE, AdaEdit, SPHERE) applied to AlphaEdit-HA on the CounterFact dataset using LLaMA. To further examine the generality of these findings, we additionally apply the same post-hoc regularization strategies to AlphaEdit-HR under identical experimental conditions.

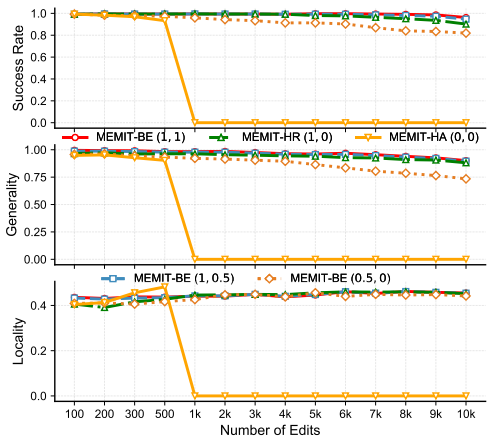



Figure 14: Interpolation results on ZsRE using MEMIT on LLaMA3-8B-Instruct.

Figure 15 presents the editing performance and locality metrics for AlphaEdit-HR combined with RECT, PRUNE, AdaEdit, and SPHERE. Consistent with the observations for AlphaEdit-HA, post-hoc regularization applied to AlphaEdit-HR exhibits similar behaviors. Specifically, increasing $\alpha$ progressively improves locality by better preserving the original weight norms and reducing unintended drift in unrelated facts, but comes at the cost of gradually declining editing success. Although post-hoc regularization delays catastrophic failure, allowing the model to withstand several hundred sequential edits, its improvement over the unregularized AlphaEdit-HR remains marginal compared with AlphaEdit-BE.

### G.6 Extended Comparison with Additional Baselines

We further compare our approach with two categories of existing knowledge editing methods: parameter-modifying methods and parameter-preserving methods.

**Parameter-modifying methods.** We include a broader set of sequential knowledge editing baselines. For fine-tuning (FT) variants, we implement LoRA-based fine-tuning [Hu et al., 2022] on the editing layers used in locate-then-edit (LTE) methods, with a rank of 8. At each editing step, a new LoRA module is initialized, fine-tuned, and merged into the base model. Following LTE practices, we augment inputs with randomly generated prefixes. For hypernetwork-based methods, we include MEND [Mitchell et al., 2022a] and its reinforcement learning-enhanced variant RLEdit [Li et al., 2025c], which train a hypernetwork to generate weight updates from gradients. Among them, RLEdit partially accounts for editing history through its reward design. We also include PMET, an enhancement of MEMIT that performs more precise interventions on hidden representations within FFN modules [Li et al., 2024]; SimIE, which approximates an ideal editor by projecting vanilla updates to protect previously edited knowledge [Guo et al., 2025b], using MEMIT as its base editor to generate vanilla updates; and HSE, which incorporates the covariance of previously edited keys to account for editing history when optimizing the residual vector for the current edit [Fang et al., 2025b].

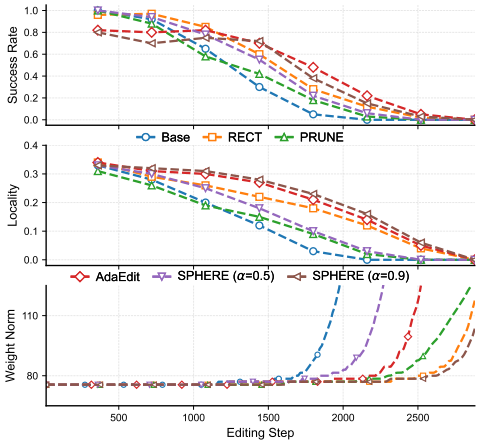



Figure 15: Effect of post-hoc regularization on AlphaEdit-HR (base), with $\alpha$ denoting the regularization strength.

Table 5 summarizes editing performance after 300 and 6,000 sequential edits on the CounterFact and ZsRE datasets using LLaMA3. The results show that most baselines struggle with long-term sequential editing, particularly at 6,000 edits. Methods not originally designed for sequential settings—such as FT, MEND, and PMET—degrade rapidly. RLEdit, which only accounts for editing history during the hypernetwork training phase, remains less effective in our strictly sequential setup (batch size = 1). SimIE falls short in long-term editing, as its history-aware protection mechanism tends to forget the current edit; notably, the original SimIE paper only evaluates performance under

Table 5: Editing performance after 300 and 6,000 sequential edits on the COUNTERFACT and ZSRE datasets using LLaMA3-8B-Instruct, GPT-2 XL, and GPT-J. Higher values indicate better performance across all metrics: editing success rate (**Suc.**), generality (**Gen.**), and locality (**Loc.**).

| **Method** | **CounterFact** | | | | | | **ZsRE** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T = 300$ | | | $T = 6000$ | | | $T = 300$ | | | $T = 6000$ | | |
| | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** | **Suc.** | **Gen.** | **Loc.** |
| **LLaMA3-8B-Instruct** | | | | | | | | | | | | |
| FT | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| MEND | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| RLEdit | 88.0 | 60.0 | 14.9 | 0.00 | 0.00 | 0.00 | 93.8 | 89.0 | 34.8 | 0.00 | 0.00 | 0.00 |
| PMET | 79.7 | 61.3 | 20.4 | 0.00 | 0.00 | 0.00 | 96.8 | 92.5 | 36.4 | 0.00 | 0.00 | 0.00 |
| SimIE | 98.8 | 68.0 | 20.1 | 0.00 | 0.00 | 0.00 | 96.5 | 92.0 | 36.0 | 74.8 | 61.3 | 27.4 |
| HSE | 99.3 | 71.7 | 20.9 | 64.6 | 42.5 | 4.10 | 98.5 | 93.2 | 36.9 | 97.8 | 93.3 | 38.5 |
| Ours | 99.3 | 71.5 | 21.0 | 96.4 | 69.7 | 21.5 | 97.4 | 93.2 | 37.2 | 97.7 | 94.7 | 40.1 |
| **GPT-2 XL** | | | | | | | | | | | | |
| FT | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| MEND | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| RLEdit | 85.2 | 58.3 | 5.10 | 0.00 | 0.00 | 0.00 | 92.4 | 87.6 | 23.5 | 0.00 | 0.00 | 0.00 |
| PMET | 78.5 | 57.9 | 4.50 | 0.00 | 0.00 | 0.00 | 95.2 | 90.1 | 21.8 | 0.00 | 0.00 | 0.00 |
| SimIE | 97.1 | 66.7 | 9.80 | 53.0 | 32.2 | 2.50 | 95.3 | 91.9 | 24.0 | 43.2 | 20.5 | 12.1 |
| HSE | 98.8 | 69.0 | 10.1 | 77.0 | 48.2 | 5.80 | 96.5 | 92.0 | 24.9 | 74.8 | 61.3 | 15.5 |
| ours | 98.2 | 68.2 | 10.5 | 97.0 | 72.2 | 9.80 | 96.8 | 92.1 | 24.5 | 96.2 | 93.7 | 23.2 |
| **GPT-J** | | | | | | | | | | | | |
| FT | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| MEND | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| RLEdit | 86.7 | 59.5 | 10.5 | 0.00 | 0.00 | 0.00 | 93.1 | 88.4 | 18.1 | 0.00 | 0.00 | 0.00 |
| PMET | 79.1 | 60.7 | 9.80 | 0.00 | 0.00 | 0.00 | 96.0 | 91.3 | 21.0 | 0.00 | 0.00 | 0.00 |
| SimIE | 97.5 | 67.2 | 12.3 | 48.7 | 23.8 | 3.20 | 95.8 | 91.5 | 26.1 | 74.1 | 61.0 | 18.2 |
| HSE | 98.8 | 68.0 | 13.2 | 81.0 | 55.3 | 7.50 | 96.5 | 92.0 | 26.0 | 84.8 | 72.3 | 23.5 |
| Ours | 98.9 | 71.0 | 14.8 | 98.0 | 68.1 | 12.5 | 97.0 | 93.8 | 26.9 | 96.8 | 92.0 | 27.5 |

up to 1,000 sequential edits. HSE achieves better performance but still underperforms our method (MEMIT-BE), especially on the CounterFact dataset and with the GPT2-XL and GPT-J models.

**Parameter-preserving methods.** Beyond parameter-modifying methods, we further compare our approach against methods that preserve the original model parameters, including memory-based, adapter-based, and in-context editors. SERAC stores edited facts in an external memory module and learns a classifier to route inputs between the memory and the base model [Mitchell et al., 2022b]. GRACE stores key-value pairs for edited facts and dynamically replaces forward activations when a matching key is detected, avoiding explicit parameter updates [Hartvigsen et al., 2023]. WISE trains an auxiliary adapter to encapsulate edits and selects the appropriate adapter during inference based on input similarity [Wang et al., 2024b]. IKE performs in-context editing by prepending exemplar demonstrations to the input, leveraging the model's in-context learning capability without modifying its parameters [Zheng et al., 2023]. Finally, MEMOIR employs residual memory and sparsifies input activations to assign edits to distinct subsets of memory parameters, reducing interference while supporting thousands of sequential edits [Wang et al., 2025b].

Table 6 summarizes editing performance after 10,000 sequential edits on the CounterFact dataset across three model architectures. We report memory usage, per-edit inference time, success rate (Succ), generality (Gen), and locality (Loc). Among the parameter-preserving baselines, we observe distinct trade-offs.

For locality, most parameter-preserving methods achieve strong locality. GRACE, SERAC, and IKE all maintain high locality across models (e.g., 21.6, 21.6, 21.1 on LLaMA3), as they do not modify the base model's parameters. WISE is the exception, showing noticeably lower locality (15.8 on LLaMA3) due to its trainable adapter module interfering with unrelated inputs.

Table 6: Editing performance on the COUNTERFACT dataset after 10,000 sequential edits. Memory and Time are reported per method.

| Model & Method | | Memory (GB) ↓ | Time (s) ↓ | Succ ↑ | Gen ↑ | Loc ↑ |
|---|---|---|---|---|---|---|
| LLaMA3 | IKE | 46.0 | 1.28 | 100.0 | 51.2 | 21.1 |
| | SERAC | 37.0 | 0.62 | 0.0 | 0.0 | 21.6 |
| | GRACE | 36.0 | 0.55 | 100.0 | 13.8 | 21.6 |
| | WISE | 37.0 | 0.38 | 97.2 | 22.5 | 15.8 |
| | MEMOIR | 39.0 | 0.47 | 78.1 | 40.2 | 21.5 |
| | Ours | 36.0 | 0.32 | 92.5 | 71.4 | 20.8 |
| GPT-J | IKE | 37.0 | 0.71 | 75.3 | 32.5 | 12.0 |
| | SERAC | 27.0 | 0.45 | 0.00 | 0.00 | 12.5 |
| | GRACE | 26.0 | 0.33 | 96.1 | 5.10 | 12.0 |
| | WISE | 29.0 | 0.25 | 80.0 | 46.7 | 11.0 |
| | MEMOIR | 30.0 | 0.28 | 85.5 | 40.4 | 12.2 |
| | Ours | 26.0 | 0.20 | 95.6 | 61.5 | 11.9 |
| GPT2-XL | IKE | 11.0 | 0.32 | 61.0 | 23.1 | 10.1 |
| | SERAC | 9.0 | 0.17 | 0.00 | 0.00 | 10.2 |
| | GRACE | 8.0 | 0.16 | 98.7 | 7.50 | 10.6 |
| | WISE | 8.0 | 0.08 | 68.9 | 25.0 | 9.50 |
| | MEMOIR | 9.0 | 0.14 | 76.3 | 29.6 | 10.4 |
| | Ours | 8.0 | 0.07 | 90.2 | 59.4 | 9.20 |

For success rate, IKE and GRACE achieve near-perfect success on LLaMA3 (100.0 each), but GRACE's success comes at the cost of generality. SERAC fails entirely (0.0) across all models. WISE shows moderate success (97.2, 80.0, 68.9), decreasing with smaller model capacity. Notably, IKE shows poor performance on GPT2-XL and GPT-J as it relies heavily on the in-context learning capability of the base model.

For efficiency, our method (AlphaEdit-BE) achieves the lowest or near-lowest memory usage (36.0, 26.0, 8.0 GB) and per-edit time (0.32, 0.20, 0.07 s) across all three models. This efficiency stems from our method being a parameter-modifying approach, whereas parameter-preserving methods incur additional inference-time overhead due to their auxiliary memory or adapter modules.

Overall, our method strikes the best balance, delivering high editing efficacy and competitive locality with no additional inference cost.

## H Future Work

While current model editing techniques have made significant progress in efficiently updating factual knowledge, several fundamental challenges remain:

1. **Multi-layer editing strategies.** Most existing methods rely on the residual propagation scheme introduced in MEMIT [Meng et al., 2023], which has recently been shown to be suboptimal [Li et al., 2024, 2025a, Liu et al., 2026d, Liu and Zhang, 2026]. Developing more effective layer-wise update mechanisms that better coordinate knowledge modifications across layers could further improve editing performance.
2. **Fact localization in Transformers.** Evidence suggests that editing later components, such as FFNs and attention modules in deeper layers, can yield better results than modifying early FFN blocks [Yang et al., 2025, Li et al., 2025c, Guo et al., 2025a, Zhou et al., 2025]. A systematic understanding of where factual knowledge resides and how it is distributed across layers and components could enable more precise and effective interventions.
3. **Superficial editing.** Recent studies suggest that current parameter-modifying methods, including LTE-based approaches, often achieve only *superficial editing* [Xie et al., 2025]: they modify model outputs to match desired targets without necessarily internalizing the underlying knowledge. This limits their ability to support reasoning based on edited facts [Thede et al., 2025]. Enabling deeper integration of new knowledge and its compositional use in downstream reasoning remains an important challenge.

4. **Beyond subject-centric editing.** Most existing approaches follow a subject-centric paradigm, anchoring edits to specific subject entities (e.g., "Paris" in "Paris is the capital of France"). Although effective for isolated facts, this paradigm struggles with edits involving the same subject but different relations or contexts, particularly in long-form text [Jiang et al., 2025a, Zeng et al., 2025, Dong et al., 2025, Xu et al., 2026]. Such scenarios can lead to interference, overwriting, and inconsistent updates. Developing relation-aware and context-sensitive editing frameworks is therefore important for handling more complex editing scenarios.
5. **Coordination of editing targets.** Existing LTE methods typically optimize the desired residuals or target values of individual edits independently, without explicitly considering their mutual compatibility. This can create difficulties when multiple edits are jointly applied: independently specified targets may be difficult to reconcile in a single batch update, particularly when their representations overlap or interfere. A promising direction is to jointly optimize editing targets across an edit set, allowing individual targets to account for one another and produce more compatible key–value mappings before the weight update is computed [Dai et al., 2026].

# I Impact Statement

While model editing enables efficient and localized updates to factual knowledge in large language models, it also introduces potential risks if deployed irresponsibly or maliciously. Editing techniques—including the method presented in this work—could be misused for targeted manipulation, such as injecting false information into specific factual associations without altering surface behaviors [Youssef et al., 2025, 2026]. Moreover, sequential editing may accumulate stealthy changes that are difficult to detect or reverse once deployed. The present work focuses on the algorithmic foundations of sequential editing, and we explicitly disclaim its use for any harmful or unethical applications. We encourage the community to develop defensive mechanisms and ethical guidelines alongside editing capabilities.

## NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main claims of the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: They have been discussed in the Conclusion Section.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: They have been provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The hyperparameter setting is clearly stated in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The data is publicly available and the detailed implementation algorithm for our method is provided in the Appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: They are introduced in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The error bar is not important in our research field as the randomness is not significant.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is foundational methodology research with no direct path to specific societal applications or harms.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models (e.g., pretrained language models, image generators) or scraped datasets, so no safeguards are required.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used (e.g., datasets, code libraries) are properly cited in Section 6 and Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not introduce new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: This paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects research was conducted, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the core methodology. Any LLM usage was limited to writing/editing assistance and does not affect scientific rigor or originality.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.