SpecEdit: A Spectral Approach for Multi-Round Knowledge Editing

Junxian Liu[1], Zikun Deng[1]*, Xin Wu[1], Dazhen Deng[2], Yi Cai[1]
[1]South China University of Technology
[2]Zhejiang University
sejxliumk@mail.scut.edu.cn, zkdeng@scut.edu.cn, xinwu@scut.edu.cn, dengdazhen@zju.edu.cn, ycai@scut.edu.cn

Abstract

Multi-round batch knowledge editing often suffers from performance degradation as edits accumulate. Focusing on the locate-then-edit paradigm, we analyze this phenomenon from a spectral perspective and identify a previously overlooked structural factor: the intrinsic knowledge of the model and historical edit memories exhibit markedly different spectral characteristics and information distributions, yet are naively coupled and jointly inverted during editing. Based on this insight, we propose *SpecEdit* to improve the model editing from a spectral perspective. *SpecEdit* performs spectral decoupling to isolate editing-critical directions and reduce destructive coupling, followed by spectral-structure-aware information compensation and spectral fusion to construct a refined closed-form solution. The module integrates seamlessly into existing editing methods without altering their original optimization procedures. Experiments on multiple LLMs and editing methods show that *SpecEdit* consistently improves performance, demonstrating that modeling spectral structure is an effective, interpretable approach and a promising direction for future research. Our code is available at: https://github.com/MKCatch/SpecEdit

1 Introduction

Large language models (LLMs) have demonstrated strong performance across a wide range of tasks; however, they inevitably encode outdated or incorrect information (Cao et al., 2021; Mitchell et al., 2022b) and may even produce hallucinated content (Ji et al., 2023; Huang et al., 2025). To address these limitations, knowledge editing (Wang et al., 2024b; Zhang et al., 2024; Wu et al., 2024) has emerged as a promising paradigm for updating model knowledge without the cost of full retraining.

Multi-round batch editing (i.e., lifelong or sequential editing) is essential when factual knowledge continually evolves. However, as edits accumulate, models often suffer degraded rewrite success, weaker paraphrase generalization, and reduced protection of neighboring knowledge, because each round must preserve intrinsic knowledge, retain prior edits, and integrate new updates simultaneously (Meng et al., 2023; Fang et al., 2025; Zhou et al., 2025a).

To address these issues, many advanced editing approaches have been developed. The parameter-preserving approach, such as WISE (Wang et al., 2024a) and KDE (Xu et al., 2025), separate intrinsic and edited knowledge into main and side parametric memories and route queries to reduce conflicts. In this paper, we focus on the parameter-modifying approach. Prior studies show that factual knowledge in Transformer models is primarily stored in structured subspaces of the MLP/FFN layers, which can be viewed as key–value-style memories (K–V pairs) (Geva et al., 2021; Dai et al., 2022). These mechanisms can be abstracted as key–value mappings with weights of neurons W, i.e., $WK = V$. The core challenge is how to modify W through a perturbation Δ to achieve the desired knowledge update. For example, WilKE focuses on *where* to apply edits by selecting proper layers (Hu et al., 2024). Built upon MEMIT (Meng et al., 2023), AlphaEdit (Fang et al., 2025) derives Δ from the null-space of K_0, explicitly preserving intrinsic knowledge K_0 while incorporating current editing information K_1 and edit history K_p across editing rounds.

These methods all require closed-form solutions of Δ by balancing the intrinsic knowledge K_0 and edit history K_p (Fig. 1), but neglect the differences in energy scale and information distribution between K_0 and K_p. Specifically, we conducted spectral analyses to examine the eigenstructure of K_0 and K_p, along with a series of *cut-and-test ex-*

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 18511–18524

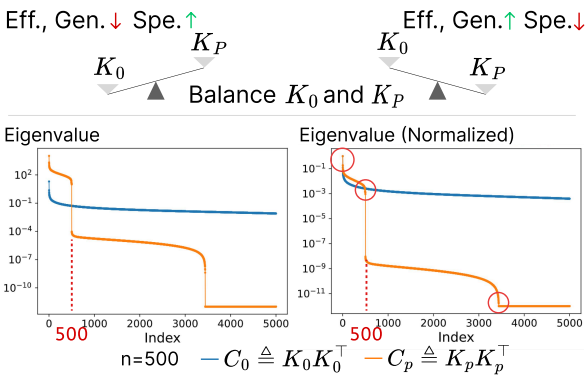

Figure 1: Balancing intrinsic knowledge K_0 and edit history K_p in knowledge editing. Over-weighting either K_0 or K_p degrades the edit outcome. $C_0 \triangleq K_0 K_0^\top$ and $C_p \triangleq K_p K_p^\top$ not only have mismatched spectra but also differ substantially in effective rank. The spectrum of C_p has three abrupt drops, and the spectrum before the second abrupt drop is very similar to C_0.

periments. Our results indicate that editing-critical information in K_p is concentrated along a few dominant directions, whereas K_0 is more dispersed and exhibits a markedly different energy scale (Fig. 1). Naively aggregating their second-order statistics, e.g., through $K_0 K_0^\top + K_p K_p^\top$ in closed-form solutions, can induce unbalanced directional updates (Davis and Kahan, 1970). Consequently, the central challenge in multi-round batch editing is not merely adjusting the weights of K_0 and K_p. Instead, it lies in mitigating interference between the concentrated editing subspace and the dispersed stability subspace across iterative updates. Without such mitigation, interference accumulates, leading to bias and degradation of model performance.

Motivated by these insights, we propose *SpecEdit*, a three-stage, general framework that combines spectral decoupling, information compensation, and spectral fusion. It first isolates the editing-critical directions of historical edit memory to reduce destructive interference, then restores lost representational capacity for both intrinsic knowledge and accumulated edits, and finally merges the refined components into a cohesive second-order structure for reliable multi-round batch editing. Our framework is *plug-and-play* and can be integrated as a structural preprocessing and enhancement module into a wide range of existing editing approaches, including locate-then-edit baseline methods such as MEMIT and constraint-aware variants such as AlphaEdit (Meng et al., 2023; Fang et al., 2025). Our experiments demonstrate that the proposed approach leads to overall improvements across a variety of models and datasets.

Our contributions can be summarized as follows:

- We perform a spectral analysis of the model's intrinsic and previously edited knowledge, revealing that editing-critical information in historical edits is highly concentrated, whereas intrinsic knowledge is more dispersed.
- Based on these insights, we propose *SpecEdit*, a novel method that explicitly accounts for the spectral differences between intrinsic and historical knowledge, leading to improved performance in multi-round knowledge editing.

2 Preliminary

2.1 Knowledge Editing Problem

We consider autoregressive large language models (LLMs) based on the Transformer architecture, where each layer updates token representations via residual connections over self-attention and feed-forward networks (FFNs) (Vaswani et al., 2017). For analytical purposes, the FFN output weights can be abstracted as implementing a key–value style associative memory (Geva et al., 2021), in which a contextualized key derived from the input prompt retrieves and writes a corresponding value into the residual stream.

Suppose we wish to edit u factual triples. We can represent them as u key–value pairs, where each key k encodes a subject–relation pair (s, r) and the associated value v encodes the object o. For example, a key representing "*Eiffel Tower*"-"*is located in*" should retrieve the value "*Paris.*" This abstraction allows us to model the FFN as an approximate linear mapping, $WK = V$, where

$$K = [\, k_1 \; k_2 \; \cdots \; k_u \,] \in \mathbb{R}^{d_0 \times u}, \\ V = [\, v_1 \; v_2 \; \cdots \; v_u \,] \in \mathbb{R}^{d_1 \times u}, \tag{1}$$

and $W \in \mathbb{R}^{d_1 \times d_0}$ is the FFN output weight matrix. For robustness, each factual triple is represented by a key vector computed under multiple prompt templates, with the final key obtained by averaging these vectors across all prompts.

This formulation underlies a broad family of **parameter-modifying** methods, which rewrite factual associations by updating a small subset of model parameters (e.g., FFN weights) (Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022). Formally, given a target set of knowledge to update,

represented by (K_1, V_1), the goal is to find a perturbation Δ such that $(W + \Delta)K_1 = V_1$, thereby modifying the model to incorporate the desired knowledge.

2.2 Knowledge Editing Solution

MEMIT (Meng et al., 2023) casts editing as a regularized least-squares problem that writes (K_1, V_1) while preserving intrinsic knowledge K_0:

$$\Delta = \arg\min_{\tilde{\Delta}} (\left\|(W + \tilde{\Delta})K_1 - V_1\right\|^2 + \left\|\tilde{\Delta}K_0\right\|^2). \tag{2}$$

Let $R = V_1 - WK_1$. The closed-form solution can be written as

$$\Delta = RK_1^\top \left(K_0K_0^\top + K_1K_1^\top\right)^{-1}. \tag{3}$$

Fang et al. (Fang et al., 2025) extend MEMIT to account for historical edits K_p,

$$\Delta = \arg\min_{\tilde{\Delta}} (\left\|\tilde{\Delta}K_0\right\|^2 + \left\|(W + \tilde{\Delta})K_1 - V_1\right\|^2 + \left\|(W + \tilde{\Delta})K_p - V_p\right\|^2), \tag{4}$$

yielding the following closed-form solution:

$$\Delta = RK_1^\top \left(K_0K_0^\top + K_pK_p^\top + K_1K_1^\top\right)^{-1}. \tag{5}$$

This solution integrates intrinsic model knowledge K_0 with edits K_p or K_1 by combining their respective second-order structures through **addition and inversion**, e.g.,

$$(K_0K_0^\top + K_pK_p^\top)^{-1}. \tag{6}$$

This implicitly assumes that these two sources can be directly summed without introducing strong geometric bias, a premise that becomes increasingly delicate under large-scale sequential editing.

AlphaEdit (Fang et al., 2025) introduces a null-space projection matrix P to constrain updates. This ensures that intrinsic knowledge (K_0) is preserved (i.e., $\tilde{\Delta}PK_0K_0^\top = 0$) while removing components of historical edits (K_p) that lie in the K_0 subspace, yielding an effective historical memory PK_p. Its objective is

$$\Delta = \arg\min_{\tilde{\Delta}} (\left\|(W + \tilde{\Delta}P)K_1 - V_1\right\|^2 + \left\|\tilde{\Delta}P\right\|^2 + \left\|\tilde{\Delta}PK_p\right\|^2), \tag{7}$$

with the corresponding closed-form update

$$\Delta = RK_1^\top P \left(K_pK_p^\top P + K_1K_1^\top P + I\right)^{-1}. \tag{8}$$

By projecting historical edits, AlphaEdit avoids directly coupling $K_0K_0^\top + K_pK_p^\top$. However, this also risks discarding components of K_p that are critical for preserving past edits.

3 Spectral Insights for Editing

A broad class of locate-then-edit methods, including MEMIT and AlphaEdit mentioned above, shares a common structural component in the closed-form updates: second-order terms derived from different knowledge sources are additively combined and then inverted. When K_0 and K_p differ substantially in energy scale or distribution, their second-order statistics through $K_0K_0^\top + K_pK_p^\top$ in closed-form solutions may be dominated by the higher-energy term, causing unbalanced directional weighting (Davis and Kahan, 1970). Consequently, inverting the coupled matrix can distort or wash out important directions from either term.

For better illustration, we denote the key structures in Equation 6 as follows:

$$C_0 \triangleq K_0K_0^\top, \qquad C_p \triangleq K_pK_p^\top.$$

C_0 characterizes the model's **intrinsic knowledge**, while C_p summarizes the **previously edited knowledge** in K_p. Below, we analyze their differences through the spectra of C_0 and C_p.

3.1 Spectral Analysis

Given a matrix C (C_0 or C_p), we normalize eigenvalues (or squared singular values) into a probability distribution: $p_i(C) = \lambda_i(C)/\sum_j \lambda_j(C)$, which minimizes the impact of differences in total energy scales across matrices and allows us to compare their spectra. We conduct multi-round editing on `GPT2-XL` using MEMIT with the CounterFact dataset (Meng et al., 2022). For more spectral analysis, please kindly refer to the appendix.

Generally, the spectra of C_0 (blue) and C_p (orange) (Fig. 1) reveal markedly different spectral patterns. C_0 exhibits a slowly decaying spectrum with a substantial high-energy tail, indicating that its information is distributed broadly across many directions and therefore more dispersed. This reflects the inherent knowledge of the language model, which is naturally rich and diverse. In contrast, C_p exhibits a spectrum with energy concentration

in a few dominant directions and multiple abrupt drops, resulting in a low-energy tail. Such concentration arises from the accumulated history of edits, and naturally leads its editing-critical information to be dominated by a small set of key directions. These observations suggest that C_0 and C_p differ substantially in both energy scale and spectrum, making effective considerations of both non-trivial.

In addition, C_p exhibits a consistent secondary signal: the location of the *second* abrupt drop in its spectrum tracks the number of edited facts. In Fig. 1, when C_p encodes 500 edited facts, the second drop occurs near $i \approx 500$; this drop shifts proportionally as the number of edited facts changes, showing a strong positive correlation. More results are provided in Appendix A. This observation suggests that the second drop delineates an *effective editing-critical subspace*: directions before the drop carry most editing information, while those after contribute minimally.

3.2 Cut-and-Test Ablations

To investigate the editing-critical subspace, We conduct targeted ablations following the same setup and metrics.

First, the C_p spectrum is split at its *second drop point*, yielding two components: (1) C_p^1, the dominant, energy-concentrated subspace before the drop, considered the *editing-critical subspace*, and (2) C_p^2, the low-energy tail after the drop, representing weakly contributing or redundant directions,

$$C_p = C_p^1 + C_p^2, \tag{9}$$

The drop index is determined via the procedure in Appendix B. Second, to examine the interaction with intrinsic knowledge, C_0 is decomposed based on C_p as

$$C_0 = C_0^1 + C_0^2, \tag{10}$$

where C_0^1 is the projection of C_0 onto the C_p^1 subspace, capturing intrinsic components that may compete with edits, and C_0^2 is the complementary part outside this subspace.

We selectively remove C_p^1, C_p^2, or C_0^1, in order to assess the necessity or importance of the editing-critical subspace, the low-energy tail, and the intrinsic knowledge components aligned with the dominant edit directions, respectively. We use the experimental setup and metrics described in Section 5. The results in Table 1 lead to three main observations.

Table 1: Cut-and-test ablations. All settings are evaluated with MEMIT on CounterFact (`GPT2-XL`). $C_0 + C_p$ denotes the full update using both C_0 and C_p. **Remove** C_p^1 **or** C_p^2 removes the corresponding decomposed component from C_p (while keeping the rest unchanged), and **Remove** C_0^1 removes the corresponding decomposed component from C_0.

Setting	Eff.	Gen.	Spe.	Flu.	Consis.
$C_0 + C_P$	93.5	82.48	72.18	616.28	38.4
Remove C_P^1	77.25	70.68	54.65	585.15	26.80
Remove C_P^2	93.00	81.70	72.64	617.72	38.37
Remove C_0^1	93.90	81.00	72.16	615.99	38.29

First, removing C_p^1 results in a substantial drop in both editing efficacy and generalization, indicating that the core editing capability of C_p is concentrated in a small set of dominant directions. In contrast, removing C_p^2 causes little performance change, suggesting that the low-energy tail contributes minimally and is largely redundant for editing. Second, removing C_0^1 leads to only minor performance changes. This is because C_0 and C_p differ by three orders of magnitude (Fig. 1), so removing C_0^1 has a much smaller effect. This implies that intrinsic knowledge does not depend on a small shared dominant subspace with C_p; instead, it is distributed more broadly across many directions, consistent with the flatter spectrum of C_0. Even after excluding the component of C_0 aligned with C_p^1, the remaining part C_0^2 is sufficient to maintain a reasonable balance between editability and stability.

These analyses suggest that C_0 and C_p have a different **energy scale**, **spectrum** and **information distribution** and their sum may be casuing unbalanced directional weighting. We propose *SpecEdit* to solve this problem.

4 SpecEdit

SpecEdit is a three-stage, general framework for multi-round batch editing, motivated by the spectral insights. Our goal is to construct a new second-order structure C_* to replace the original additive term, while leaving the subsequent optimization and inversion steps unchanged.

SpecEdit consists of three stages: **(i) Spectral decoupling**: the editing-critical subspace of C_p is used to separate the functional subspaces of C_0 and C_p, reducing competition and interference along key directions; **(ii) Information compensation**: spectrum-aware mechanisms are applied to

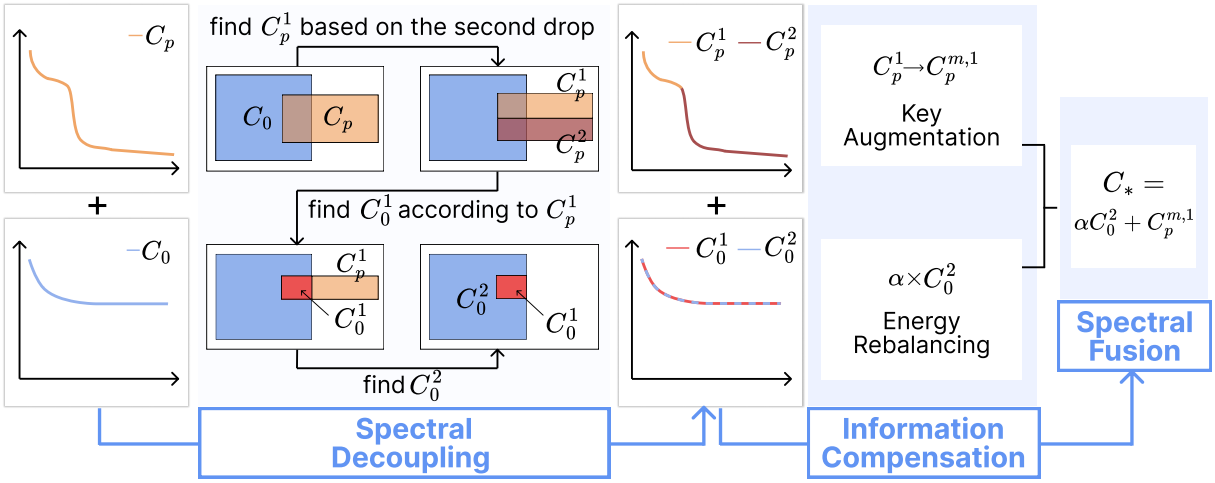

Figure 2: Overview of our three-stage pipeline. We first observe a clear spectrum mismatch between C_0 and C_p, indicating dispersed vs. concentrated information distributions. **Spectral decoupling** splits them along the editing-critical subspace of C_p, yielding the complementary component C_0^2 and the dominant component C_p^1. **Information compensation** then strengthens these components in a structure-matched manner, producing the rebalanced αC_0^2 and the augmented keys $C_p^{m,1}$. **Spectral fusion** finally merges the enhanced components to obtain C_*, which reduces redundancy while improving the representations of both intrinsic and edited knowledge.

C_0 and C_p to recover information lost during decoupling and pruning; **(iii) Spectral fusion**: the retained and enhanced components are merged into a new second-order term, which is then inserted into the standard closed-form solution of Δ. The full pipeline is illustrated in Fig. 2.

4.1 Spectral Decoupling

The *second* abrupt drop in the normalized spectrum of C_p is used to define the boundary of its energy-concentrated, editing-critical subspace. Formally, if $\mathcal{R} = \{r_1, r_2, \dots\}$ are the detected drop points in order, we set the split index as $r = r_2$. The specific method for searching for r_2 is provided in Appendix B. The top-r eigenvectors of C_p, where r is the second drop point, form the column space defining edit history C_p's editing-critical subspace projector P. The detailed procedure for constructing P is provided in Appendix C.

Both C_0 and C_p are then decomposed into a core block within this subspace and a complementary residual:

$$C_0^1 = PC_0P, \quad C_p^1 = PC_pP, \tag{11}$$

$$C_0^2 = C_0 - C_0^1, \quad C_p^2 = C_p - C_p^1. \tag{12}$$

Here, C_p^1 captures the concentrated core of edit history K_p, while C_p^2 contains the low-energy tail and all off-core components. Similarly, C_0^1 is the projection of C_0 onto the same critical subspace, and C_0^2 collects the remaining components.

4.2 Information Compensation

Spectral decoupling reduces conflicts between C_0 and C_p but inevitably discards some information: part of C_0 in the energy-concentrated subspace of K_p (C_0^1) and part of C_p in the low-energy tail of K_p (C_p^2). To restore lost capacity, we compensate for this information with two spectrum-aware strategies.

C_0 energy rebalancing. Since C_0 is dispersed and redundant, we simply scale its retained component C_0^2 by a factor $\alpha > 1$, restoring its overall energy without reintroducing interference in the critical subspace.

C_p contextual key augmentation. C_p is energy-concentrated, with editing information concentrated in a few directions. Simple scaling cannot recover its structural information. Instead of averaging the key vectors across multiple context templates (prompts) as done in prior work, we *stack* them column-wise to form an augmented matrix C_p^m where m denotes the number of context-specific key vectors being stacked. This matrix replaces the original C_p, and its pre-drop component $C_p^{m,1}$ replaces C_p^1, preserving the full set of context-specific directions for subsequent editing. This augmentation introduces multi-context consistency

without changing the target semantics, strengthens the editing-critical subspace, increases the effective information in C_p, and improves robustness and generalization in multi-round editing.

Based on sensitivity analysis (Appendix D), we set $\alpha = 2$ and $m = 5$.

4.3 Spectral Fusion

Afterwards, the retained components are fused into a final second-order term:

$$C_* = \alpha C_0^2 + C_p^{m,1}, \tag{13}$$

where $C_p^{m,1}$ is the energy-concentrated subspace from C_p and C_0^2 is the stable residual of C_0. The hyperparameter α restores energy lost during decoupling/pruning. In all experiments, we set $\alpha = 2$ as a simple scale to restore the overall magnitude of the retained C_0^2 term.

Fusion examples. *SpecEdit* is a lightweight fusion plugin that can be inserted into a shared second-order substructure of a pre-localized, post-edited solver. Taking MEMIT (Meng et al., 2023) as an example, its closed-form solution can be rewritten as

$$\Delta_{\text{MEMIT}} = RK_1^\top (C_*)^{-1}. \tag{14}$$

In this way, *SpecEdit* is applied to MEMIT. Since the closed-form solutions of PRUNE (Ma et al., 2025) and RECT (Gu et al., 2024) are the same as those of MEMIT, *SpecEdit* can be directly applied in the same way. For AlphaEdit (Fang et al., 2025), we replace the original null projection matrix $C_0 = K_0 K_0^T$ with the null projection matrix of C_*, while keeping its closed-form solution unchanged to apply *SpecEdit*.

5 Experiments

5.1 Experimental Setup

We conduct experiments on three Transformer-based LLMs: `GPT2-XL` (1.5B), `GPT-J` (6B), and `LLaMA3` (8B). *SpecEdit* is integrated into several representative locate-then-edit baselines, including MEMIT (Meng et al., 2023), PRUNE (Ma et al., 2025), RECT (Gu et al., 2024), and AlphaEdit (Fang et al., 2025). We evaluate on the CounterFact (Meng et al., 2022) and ZsRE (Levy et al., 2017) benchmarks. In total, we edit 2000 knowledge facts, using a batch size of 100 edits. Following prior work, we report five standard metrics: *Efficacy* (Eff.), *Generalization* (Gen.), *Specificity* (Spe.), *Fluency* (Flu.), and *Consistency* (Consis.). Higher scores indicate better performance for all metrics. All experiments are run on a workstation equipped with an NVIDIA RTX Pro 6000 GPU.

5.2 Main Results

Table 2 reports the results. Generally, across most method–model–dataset settings, *SpecEdit* improves the primary editing objectives named *Efficacy* and *Generalization*, while preserving *Specificity*, *Fluency*, and *Consistency*. This demonstrates that *SpecEdit* is model-agnostic, providing a general structural refinement to the "addition-and-inversion" subcomponent in the closed-form solution, which improves update reliability and robustness to paraphrase variation.

In a number of cases, the gains are substantial. For example, on `LLaMA3-8B` with MEMIT, efficacy and generalization on CounterFact increase from 65.65%–64.65% to 99.43%–95.05%, while on zsRE they rise from 34.62%–31.28% to 95.29%–91.38%. With RECT, efficacy on CounterFact improves from 66.05% to 96.47%, with specificity and consistency largely maintained. Notably, on `GPT-J` with PRUNE, efficacy on ZsRE recovers dramatically from 0.15% to 35.09%. In contrast, the baseline methods suffer from strong competition between intrinsic structure and historical edit memory within the editing-critical subspace. By mitigating this interference and restoring usable capacity, *SpecEdit* achieves markedly better editing outcomes.

While overall performance improves, a small number of negative cases are observed. On zsRE with `GPT2-XL`, *SpecEdit* slightly decreases efficacy and generalization for RECT (e.g., from 81.02% to 77.35% and from 73.08% to 68.07%) and AlphaEdit (e.g., from 94.81% to 87.20% and from 86.11% to 77.54%). We hypothesize that, in certain method–model settings, the information enhancement step may overly reinforce directions that are not strictly editing-critical, reducing update controllability and harming rewrite success. This effect appears more pronounced for smaller models and the ZsRE benchmark.

5.3 Ablation Study

We conduct an ablation study to disentangle the respective contributions of spectral decoupling

Model	Method		Counterfact					ZsRE		
			Eff.	Gen.	Spe.	Flu.	Consis.	Eff.	Gen.	Spe.
LLAMA3	MEMIT	–	$65.65_{\pm 0.47}$	$64.65_{\pm 0.42}$	$51.56_{\pm 0.38}$	$437.43_{\pm 1.67}$	$6.58_{\pm 0.11}$	$34.62_{\pm 0.36}$	$31.28_{\pm 0.34}$	$18.49_{\pm 0.19}$
		SpecEdit	$99.43_{\pm 0.23}$	$95.05_{\pm 0.05}$	$80.28_{\pm 0.52}$	$627.67_{\pm 0.67}$	$32.95_{\pm 0.22}$	$95.29_{\pm 0.24}$	$91.38_{\pm 0.17}$	$32.40_{\pm 0.08}$
		DIFF	+33.78 (51%)	+30.40 (47%)	+28.72 (56%)	+190.24 (43%)	+26.37 (>99%)	+60.67 (>99%)	+60.10 (>99%)	+13.91 (75%)
	PRUNE	–	$68.25_{\pm 0.46}$	$64.75_{\pm 0.41}$	$49.82_{\pm 0.36}$	$418.03_{\pm 1.52}$	$5.90_{\pm 0.10}$	$24.77_{\pm 0.27}$	$23.87_{\pm 0.27}$	$20.69_{\pm 0.23}$
		SpecEdit	$87.23_{\pm 0.43}$	$80.60_{\pm 0.00}$	$57.72_{\pm 0.22}$	$607.39_{\pm 8.53}$	$30.30_{\pm 3.78}$	$28.09_{\pm 0.19}$	$24.00_{\pm 0.27}$	$16.84_{\pm 0.71}$
		DIFF	+18.98 (28%)	+15.85 (24%)	+7.90 (16%)	+189.36 (45%)	+24.40 (>99%)	+3.32 (13%)	+0.13 (1%)	-3.85 (-19%)
	RECT	–	$66.05_{\pm 0.47}$	$63.62_{\pm 0.43}$	$61.41_{\pm 0.37}$	$526.62_{\pm 0.44}$	$20.54_{\pm 0.09}$	$86.05_{\pm 0.23}$	$80.54_{\pm 0.27}$	$31.67_{\pm 0.22}$
		SpecEdit	$96.47_{\pm 0.08}$	$83.91_{\pm 0.27}$	$84.74_{\pm 0.50}$	$629.46_{\pm 0.30}$	$30.62_{\pm 0.35}$	$92.16_{\pm 0.47}$	$86.38_{\pm 1.13}$	$32.46_{\pm 0.03}$
		DIFF	+30.42 (46%)	+20.29 (32%)	+23.33 (38%)	+102.84 (20%)	+10.08 (49%)	+6.11 (7%)	+5.84 (7%)	+0.79 (2%)
	AlphaEdit	–	$98.90_{\pm 0.10}$	$94.22_{\pm 0.19}$	$67.88_{\pm 0.29}$	$622.49_{\pm 0.16}$	$32.40_{\pm 0.11}$	$94.47_{\pm 0.13}$	$91.13_{\pm 0.19}$	$32.55_{\pm 0.22}$
		SpecEdit	$99.58_{\pm 0.03}$	$96.25_{\pm 0.03}$	$75.34_{\pm 0.21}$	$622.58_{\pm 1.32}$	$33.44_{\pm 0.32}$	$95.28_{\pm 0.04}$	$91.39_{\pm 0.40}$	$32.50_{\pm 0.04}$
		DIFF	+0.68 (1%)	+2.03 (2%)	+7.46 (11%)	+0.09 (0%)	+1.04 (3%)	+0.81 (1%)	+0.26 (0%)	-0.05 (-0%)
GPT-J	MEMIT	–	$98.55_{\pm 0.11}$	$95.50_{\pm 0.16}$	$63.64_{\pm 0.31}$	$546.28_{\pm 0.88}$	$34.89_{\pm 0.15}$	$94.91_{\pm 0.16}$	$90.22_{\pm 0.23}$	$30.39_{\pm 0.27}$
		SpecEdit	$99.58_{\pm 0.17}$	$93.84_{\pm 2.59}$	$74.70_{\pm 1.37}$	$614.37_{\pm 1.50}$	$40.80_{\pm 0.90}$	$99.81_{\pm 0.04}$	$96.64_{\pm 0.07}$	$28.10_{\pm 0.12}$
		DIFF	+1.03 (1%)	-1.66 (-2%)	+11.06 (17%)	+68.09 (12%)	+5.91 (17%)	+4.90 (5%)	+6.42 (7%)	-2.29 (-8%)
	PRUNE	–	$86.15_{\pm 0.34}$	$86.85_{\pm 0.29}$	$53.87_{\pm 0.35}$	$427.14_{\pm 0.53}$	$14.78_{\pm 0.11}$	$0.15_{\pm 0.02}$	$0.15_{\pm 0.02}$	$0.00_{\pm 0.00}$
		SpecEdit	$89.27_{\pm 0.38}$	$90.27_{\pm 0.58}$	$58.03_{\pm 0.27}$	$503.08_{\pm 4.25}$	$31.53_{\pm 1.04}$	$35.09_{\pm 2.24}$	$33.25_{\pm 1.64}$	$22.42_{\pm 0.04}$
		DIFF	+3.12 (4%)	+3.42 (4%)	+4.16 (8%)	+75.94 (18%)	+16.75 (>99%)	+34.94 (>99%)	+33.10 (>99%)	+22.42 (n/a)
	RECT	–	$98.80_{\pm 0.10}$	$86.58_{\pm 0.28}$	$72.22_{\pm 0.28}$	$617.31_{\pm 0.19}$	$41.39_{\pm 0.12}$	$96.38_{\pm 0.14}$	$91.21_{\pm 0.21}$	$27.79_{\pm 0.26}$
		SpecEdit	$97.43_{\pm 0.28}$	$81.50_{\pm 0.50}$	$78.89_{\pm 0.15}$	$617.61_{\pm 0.42}$	$38.51_{\pm 0.38}$	$92.47_{\pm 0.70}$	$83.11_{\pm 0.57}$	$26.93_{\pm 0.26}$
		DIFF	-1.37 (-1%)	-5.08 (-6%)	+6.67 (9%)	+0.30 (0%)	-2.88 (-7%)	-3.91 (-4%)	-8.10 (-9%)	-0.86 (-3%)
	AlphaEdit	–	$99.75_{\pm 0.08}$	$96.38_{\pm 0.23}$	$75.48_{\pm 0.21}$	$618.50_{\pm 0.17}$	$42.08_{\pm 0.15}$	$99.79_{\pm 0.14}$	$96.00_{\pm 0.22}$	$28.29_{\pm 0.25}$
		SpecEdit	$99.85_{\pm 0.05}$	$98.10_{\pm 0.10}$	$75.21_{\pm 0.47}$	$615.33_{\pm 0.84}$	$41.75_{\pm 0.08}$	$99.57_{\pm 0.08}$	$96.42_{\pm 0.17}$	$27.85_{\pm 0.15}$
		DIFF	+0.10 (0%)	+1.72 (2%)	-0.27 (-0%)	-3.17 (-1%)	-0.33 (-1%)	-0.22 (-0%)	+0.42 (0%)	-0.44 (-2%)
GPT2-XL	MEMIT	–	$94.70_{\pm 0.22}$	$85.82_{\pm 0.28}$	$60.50_{\pm 0.32}$	$477.26_{\pm 0.54}$	$22.72_{\pm 0.15}$	$79.17_{\pm 0.32}$	$71.44_{\pm 0.36}$	$26.42_{\pm 0.25}$
		SpecEdit	$98.83_{\pm 0.08}$	$92.66_{\pm 0.44}$	$68.28_{\pm 0.24}$	$600.57_{\pm 3.27}$	$38.99_{\pm 0.29}$	$94.33_{\pm 0.34}$	$86.89_{\pm 1.14}$	$26.59_{\pm 0.12}$
		DIFF	+4.13 (4%)	+6.84 (8%)	+7.78 (13%)	+123.31 (26%)	+16.27 (72%)	+15.16 (19%)	+15.45 (22%)	+0.17 (1%)
	PRUNE	–	$82.05_{\pm 0.38}$	$78.55_{\pm 0.34}$	$53.02_{\pm 0.35}$	$530.47_{\pm 0.39}$	$15.93_{\pm 0.11}$	$21.62_{\pm 0.30}$	$19.27_{\pm 0.28}$	$13.19_{\pm 0.18}$
		SpecEdit	$88.78_{\pm 2.73}$	$84.61_{\pm 1.09}$	$55.20_{\pm 0.58}$	$552.93_{\pm 1.32}$	$29.67_{\pm 0.39}$	$41.23_{\pm 2.22}$	$39.14_{\pm 1.80}$	$17.60_{\pm 0.25}$
		DIFF	+6.73 (8%)	+6.06 (8%)	+2.18 (4%)	+22.46 (4%)	+13.74 (86%)	+19.61 (91%)	+19.87 (>99%)	+4.41 (33%)
	RECT	–	$92.15_{\pm 0.26}$	$81.15_{\pm 0.33}$	$65.13_{\pm 0.31}$	$480.83_{\pm 0.62}$	$21.05_{\pm 0.16}$	$81.02_{\pm 0.31}$	$73.08_{\pm 0.35}$	$24.85_{\pm 0.25}$
		SpecEdit	$93.20_{\pm 0.40}$	$81.09_{\pm 1.56}$	$70.96_{\pm 1.26}$	$595.28_{\pm 19.34}$	$39.44_{\pm 0.20}$	$77.35_{\pm 0.22}$	$68.07_{\pm 0.30}$	$24.70_{\pm 0.33}$
		DIFF	+1.05 (1%)	-0.06 (-0%)	+5.83 (9%)	+114.45 (24%)	+18.39 (87%)	-3.67 (-5%)	-5.01 (-7%)	-0.15 (-1%)
	AlphaEdit	–	$99.50_{\pm 0.24}$	$93.95_{\pm 0.34}$	$66.39_{\pm 0.31}$	$597.88_{\pm 0.18}$	$39.38_{\pm 0.15}$	$94.81_{\pm 0.30}$	$86.11_{\pm 0.29}$	$25.88_{\pm 0.21}$
		SpecEdit	$99.17_{\pm 0.03}$	$93.97_{\pm 0.33}$	$64.23_{\pm 0.29}$	$587.54_{\pm 1.81}$	$38.74_{\pm 0.03}$	$87.20_{\pm 0.98}$	$77.54_{\pm 0.69}$	$21.53_{\pm 0.39}$
		DIFF	-0.33 (-0%)	+0.02 (0%)	-2.16 (-3%)	-10.34 (-2%)	-0.64 (-2%)	-7.61 (-8%)	-8.57 (-10%)	-4.35 (-17%)

Table 2: Our method is integrated into existing knowledge-editing methods and compared against their original implementations, evaluated on CounterFact and ZsRE across `LLaMA3`, `GPT-J`, and `GPT2-XL`. For each setting, the table reports performances of the original baseline before (–) and after being integrated with *SpecEdit* (*SpecEdit*), as well as their difference (DIFF). Relative changes (in %) are shown in parentheses. Relative changes with magnitude at least 2% are highlighted (green: improvement; red: degradation).

and information compensation in *SpecEdit*. The ablation study is performed on three models, `GPT2-XL` (1.5B), `GPT-J` (6B), and `LLaMA3` (8B), using MEMIT as the representative base editing method and CounterFact as the benchmark. We compare five settings: the original MEMIT baseline (**Baseline**), a variant that simply incorporates edit histroy K_p or C_p (**+**K_p), K_p with spectral decoupling (**+**K_p**+SD**), K_p with information compensation (**+**K_p**+IC**), and the full *SpecEdit* (**+*SpecEdit***). For each setting, we report the five standard metrics. We also track $\dim(K_p)$, the dimensionality of K_p, as a proxy for the effective capacity of the memory/update space. This value is reported alongside each model to provide context for interpreting the ablation results. The results (Table 3) confirm *SpecEdit*'s effectiveness and its individual modules, while also revealing several additional insights.

Spectral decoupling is performance-preserving. Across all three models, **+**K_p**+SD** remains nearly identical to **+**K_p on all metrics. This invariance is *by design*: spectral decoupling is intended as a safe, structure-level operation that separates potentially interfering components *without* perturbing the editing-critical directions carried by K_0 and K_p. Thus, the stable efficacy and generalization indicate that the spectral decoupling can reorganize or decouple the update space while preserving the functional content required for rewrite success and paraphrase generalization, rather than chasing gains on its own.

Information compensation is effective but capacity-sensitive. Applying information compensation reveals a clear capacity-dependent behavior tied to the dimensionality of the memory term K_p. We interpret K_p as the memory/update space, whose dimensionality $\dim(K_p)$ reflects its

Model ($\dim(K_p)$)	Variant	Eff.	Gen.	Spe.	Flu.	Consis.
GPT2-XL (6400)	Baseline	94.7	85.82	60.5	477.26	22.72
	+K_p	93.5	82.48	72.18	616.28	38.4
	+K_p+SD	93.15	81.5	72.92	616.14	38.28
	+K_p+IC	81.9	65.92	74.95	624.15	36.26
	+*SpecEdit*	97.85	91.7	69.06	605.07	39.1
GPT-J (16384)	Baseline	98.55	95.5	63.64	546.28	34.89
	+K_p	99.65	94.55	75.65	617.94	41.69
	+K_p+SD	99.75	95.1	75.43	614.8	41.51
	+K_p+IC	99.65	97.3	76.21	616.33	41.99
	+*SpecEdit*	99.4	96.43	73.33	612.87	41.7
LLaMA3 (14336)	Baseline	65.65	64.65	51.56	437.43	6.58
	+K_p	98.95	89.32	82.55	628.68	31.74
	+K_p+SD	98.8	89.32	83	628.33	31.63
	+K_p+IC	99.3	92.88	82.82	629.53	32.42
	+*SpecEdit*	99.65	95.1	80.8	628.34	33.17

Table 3: Ablation study using MEMIT and its variants on CounterFact. **Baseline** denotes the original MEMIT. **+K_p** incorporates the previous-edit memory term K_p. **+K_p+SD** applies only the spectral decoupling. **+K_p+IC** applies only the information compensation. ***+SpecEdit*** applies the full verson of *SpecEdit*.

effective capacity to store and reuse information from previous edits, as well as to absorb compensatory information from the current update. On models with larger $\dim(K_p)$ (`LLaMA3`/`GPT-J`; 14336/16384), information compensation consistently improves core editing metrics, notably efficacy and generalization. The results imply that a higher-capacity memory space can accommodate compensation without harming rewrite quality or generalization. In contrast, on `GPT2-XL` with a smaller $\dim(K_p) = 6400$, compensation degrades efficacy and generalization despite gains in specificity and fluency, suggesting a saturation effect in which limited memory capacity forces a trade-off between rewrite success and narrower behaviors.

Combining twos yields robust gains. *SpecEdit* delivers stable improvements across models and is especially effective in low-capacity settings. On `GPT2-XL`, spectral decoupling enables information compensation to improve efficacy and generalization, whereas compensation alone fails. This indicates that decoupling is essential for safely injecting additional editing information under limited K_p capacity, by isolating interfering components and freeing effective space for integration into K_0 and K_p. On the medium-capacity `GPT-J`, individual components yield consistent gains, but the improvements begin to saturate when combined, suggesting a capacity-related bottleneck. On `LLaMA3`, introducing K_p already yields substantial gains, indicating sufficient model capacity to store and reuse accumulated edits. Spectral decoupling and information compensation each provide further improvements, and their combination in *SpecEdit* leads to the best overall performance.

6 Related Work

Model editing methods are commonly grouped into *parameter-modifying* and *parameter-preserving* paradigms (Yao et al., 2023; Wang et al., 2024b).

Parameter-modifying methods directly alter model weights to inject or correct knowledge. A major subclass is the *locate-then-edit* family, which identifies fact-relevant directions and applies explicit updates. Early work focuses on sparse or neuron-level editing (Dai et al., 2022; Zhou et al., 2025b; Jiang et al., 2025). ROME (Meng et al., 2022) performs rank-one updates on identified subspaces, and MEMIT (Meng et al., 2023) extends ROME to batch editing of many facts in a single update. Subsequent methods introduce constraints to mitigate interference, including PRUNE (Ma et al., 2025), RECT (Gu et al., 2024), and MPES (Gupta et al., 2025). For lifelong editing, WilKE (Hu et al., 2024) adapts layer selection over time, while AlphaEdit (Fang et al., 2025) restricts updates to the null space of intrinsic knowledge to avoid conflicts.

Another subclass comprises *meta-learning or amortized editors*, which learn to predict editing updates. Representative examples include MEND (Mitchell et al., 2022a) and KnowledgeEditor (Cao et al., 2021), which use learned update functions or hyper-networks to approximate weight modifications rather than closed-form solutions.

Parameter-preserving methods instead store edits in external memories or modules to avoid overwriting base parameters. Memory-based editors, such as SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023), Transformer-Patcher (Huang et al., 2023), WISE (Wang et al., 2024a), and KDE (Xu et al., 2025), decouple edit storage from model parameters and incorporate edit information at inference time to adjust model outputs. Prompt-based editors (e.g., IKE (Zheng et al., 2023) and MemPrompt (Madaan et al., 2022)) encode edits as natural-language examples or feedback and apply them through in-context prompting, without modifying model parameters.

7 Conclusion

We analyze performance degradation in multi-round model editing by identifying structural conflicts between intrinsic knowledge (K_0) and historical edits (K_p) in coupled second-order updates. Our analysis shows K_0 is dispersed while K_p is energy-concentrated, concentrating effective editing directions. Our analysis shows that K_0 and K_P exhibit drastically different spectrums and energy allocations, an overly large mismatch in their information distributions. We propose a plug-and-play module named *SpecEdit* with spectral decoupling to reduce interference, information compensation to restore lost capacity, and spectral fusion to combine enhanced components for stable updates. Integrated into existing locate-then-edit solvers, it improves knowledge editing performance without changing the original solver. Ablations confirm both stages are necessary, yielding substantial gains across models and datasets.

8 Limitations

We identify the editing-critical, energy-concentrated subspace via spectral drop points and perform decoupling and compensation using blockwise weighting and an energy restoration coefficient. This design is simple, interpretable, and empirically stable across our evaluated settings. Nevertheless, the optimal split point and hyperparameters may not be universally optimal across all models, layers, or datasets. An interesting direction for future work is to develop more automated selection strategies, such as cumulative-energy criteria or lightweight validation-based tuning, to further reduce manual configuration while preserving robustness.

Our current compensation mechanisms for both K_0 and K_p are intentionally coarse-grained, focusing on restoring effective capacity while maintaining simplicity and generality. Future work could explore more fine-grained enhancements, including adaptive selection or weighting of prompt templates, improved consistency constraints across contexts, or noise-aware key filtering, to further improve robustness and reduce the influence of less informative contexts, especially in smaller models or more sensitive datasets.

9 Ethical considerations

AI assistance disclosure: We used ChatGPT for language polishing and proofreading only. It was not used to generate research ideas, experimental design, results, or conclusions.

Acknowledgments

This research was supported by Guangdong Basic and Applied Basic Research Foundation (2025A1515010162), National Natural Science Foundation of China (62402184), Zhejiang Provincial Natural Science Foundation of China (LD25F020003), the Guangdong Provincial Fund for Basic and Applied Basic Research—Regional Joint Fund Project (Key Project) (2023B1515120078), the Guangdong Provincial Natural Science Foundation for Outstanding Youth Team Project (2024B1515040010), and the Science and Technology Planning Project of Guangdong Province (2025B0101120003).

References

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

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

Chandler Davis and W. M. Kahan. 1970. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46.

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

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

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

Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025. Lifelong knowledge editing requires better regularization. In *Findings of the Association for Computational Linguistics: EMNLP*.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Proceedings of Annual Conference on Neural Information Processing Systems*.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. WilKE: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics*, pages 3476–3503.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2):42:1–42:55.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-Patcher: One mistake worth one neuron. In *Proceedings of the International Conference on Learning Representations*.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 1827–1843.

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

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

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

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. MemPrompt: Memory-assisted prompt editing with user feedback. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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

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

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

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

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

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3).

Xin Wu, Yuqi Bu, Yi Cai, and Tao Wang. 2024. Updating large language models' memories with time constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13693–13702.

Haoyu Xu, Pengxiang Lan, Enneng Yang, Guibing Guo, Jianzhe Zhao, Linying Jiang, and Xingwei Wang. 2025. Knowledge decoupling via orthogonal projection for lifelong editing of large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13194–13213.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. pages 10222–10240.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, and 3 others. 2024. A comprehensive study of knowledge editing for large language models. *Preprint*, arXiv:2401.01286.

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

Bihan Zhou, Haopeng Ren, Li Yuan, Yi Cai, Liuwen Cao, and Zikun Deng. 2025a. Ruleedit: Towards rule-level knowledge generalization to mitigate over-editing in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3159–3175.

Wei Zhou, Wei Wei, Guibang Cao, and Fei Wang. 2025b. Editing memories through few targeted neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Second Drop Point as a Capacity Boundary of K_p

We provide additional evidence that the *second* abrupt drop point in the spectrum of $K_pK_p^\top$ can be viewed as the capacity boundary of edited knowledge stored in K_p.

Setting. All results use MEMIT on `GPT2-XL` and are evaluated on the CounterFact benchmark. We test three edit volumes $n \in \{500, 1000, 1500\}$ (i.e., the number of facts being edited) under two conditions: (i) **baseline editing** using K_p, and (ii) **editing after information compensation** that yields an augmented memory K_p^m. The second condition is included to prove that information compensation would not shift the second-drop location.

Measurement. For each n-condition pair, we compute the normalized eigenvalue curve of $K_pK_p^\top$ (or $K_p^m(K_p^m)^\top$). The curves are plotted in Fig. 3. Besides, we detect spectral drop points based on the normalized eigenvalue curve using the detection method in Appendix B. We record the index of the *second* drop point r_2.

Observations. Across all tested settings, the detected r_2 closely aligns with n, differing by at most a small margin (about 1%). For each fixed n, the r_2 values remain stable after information compensation, indicating that compensation strengthens useful signals without altering the location of the second spectral drop.

B Second-Drop Point Detection

We describe how we detect the *second* drop point from the spectrum of a matrix, denoted as C_p. This split index is later used to define a capacity-aligned spectral boundary.

Spectrum normalization. Let $C_p \in \mathbb{R}^{d\times d}$ be symmetric positive semi-definite with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$. We normalize the spectrum by

$$y_i \triangleq \frac{\lambda_i}{\lambda_1}, \qquad i = 1, \ldots, D, \tag{15}$$

where we only keep the leading part of the spectrum for a protective truncation,

$$D \triangleq \min(d, \texttt{max\_points}). \tag{16}$$

Here, `max_points` is a safeguard hyper-parameter that limits the spectrum length used for drop detection, preventing unstable splits caused by extremely large C_p (or overly long spectral tails), while also improving efficiency.

Log-spectrum and slope-jump scoring. To reveal relative drops across scales, we operate on the log-spectrum

$$\ell_i \triangleq \log(\max(y_i, \epsilon)), \qquad i = 1, \ldots, D, \tag{17}$$

and define adjacent log-slopes

$$s_i \triangleq \ell_i - \ell_{i+1}, \qquad i = 1, \ldots, D-1. \tag{18}$$

A sharp "waterfall" is characterized by an abrupt change of slope. We therefore score each location by a *slope jump* across a fixed gap w:

$$j_r \triangleq s_r - s_{r-w}, \qquad r = w+1, \ldots, D-1. \tag{19}$$

Large j_r indicates that the spectrum becomes substantially steeper around r compared to $r - w$.

Top-k candidates with separation. We rank indices r by j_r in descending order and select the top-k candidates, while enforcing a minimum separation constraint (to avoid selecting multiple nearby points caused by local noise). This yields a candidate set of drop locations

$$\mathcal{R} = \{r_1, \ldots, r_k\}, \qquad r_i \in \{w+1, \ldots, D-1\}, \tag{20}$$

where each $r \in \mathcal{R}$ is a 1-based index on the eigenvalue curve.

Capacity-aligned selection for the second drop. Let N_p denotedenote the amount of edited knowledge stored in K_p. We restrict the second-drop selection to a capacity-centered window

$$\mathcal{W}(N_p, \delta) \triangleq \{r \mid |r - N_p| \leq \delta\}. \tag{21}$$

Among candidates within the window, we choose the one closest to N_p:

$$r_2 \triangleq \arg\min_{r \in \mathcal{R} \cap \mathcal{W}(N_p, \delta)} |r - N_p|. \tag{22}$$

If no candidate falls inside the window, we fall back to the window center and clip to the valid spectral index range:

$$r_2 \triangleq \mathrm{clip}(N_p,\ w+1,\ D-1). \tag{23}$$

Empirically, across all tested settings, the detected r_2 deviates from N_p by at most 1%. Accordingly, we use a small δ that is sufficient to cover all evaluated capacities.

C Constructing the Projector P

We construct an orthogonal projector P from the leading eigenspace induced by the memory matrix $K_p \in \mathbb{R}^{d\times l}$. Since K_p is generally rectangular, we first map it to a symmetric positive semi-definite matrix:

$$A \triangleq K_p K_p^\top \in \mathbb{R}^{d\times d}. \tag{24}$$

For numerical consistency, we optionally apply symmetrization and a small diagonal shift:

$$A \leftarrow \tfrac{1}{2}(A + A^\top), \qquad A \leftarrow A + \epsilon I_d, \tag{25}$$

where $\epsilon > 0$ is a small constant and I_d is the $d \times d$ identity matrix.

The target rank r is set to the detected second-drop index r_2 obtained in Appendix B, i.e., $r \triangleq r_2$. This choice aligns the projector subspace with the capacity-aligned spectral boundary identified from the spectrum.

Let the eigenpairs of A be (λ_i, u_i), ordered by $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$. We take the top-$r$ eigenvectors and form

$$U \triangleq [u_1, \ldots, u_r] \in \mathbb{R}^{d\times r}, \qquad U^\top U = I_r. \tag{26}$$

The projector onto the resulting r-dimensional subspace is defined as

$$P \triangleq UU^\top \in \mathbb{R}^{d\times d}, \tag{27}$$

which satisfies $P^\top = P$ and $P^2 = P$. Intuitively, P captures the dominant directions of K_p up to the capacity-aligned split rank, and will be used to project matrices/vectors onto this principal subspace.

D Sensitivity Analysis

We conduct a sensitivity analysis on the scaling factor α and the number of contextual keys m. The results are reported in Table 4 and Table 5.

For α, the overall performance remains relatively stable across the tested values, indicating that the method is not overly sensitive to moderate changes in the scaling factor. When α increases from 2 to 4, the results fluctuate only slightly, and no dramatic performance shift is observed. Among these settings, $\alpha = 2$ achieves the best overall balance: it yields the highest effectiveness, the highest generalization score, and the best consistency, while remaining competitive on specificity and fluency. Although larger α values slightly improve some individual metrics, they also introduce noticeable trade-offs on the more important overall editing metrics. Therefore, we choose $\alpha = 2$ as the default setting.

Table 4: Sensitivity analysis of α.

α	Eff.	Gen.	Spe.	Flu.	Consis.
2	97.85	91.70	69.06	605.07	39.10
3	96.45	89.60	70.38	608.09	38.61
4	97.50	91.25	69.88	606.07	38.84

Table 5: Sensitivity analysis of m.

m	Eff.	Gen.	Spe.	Flu.	Consis.
1	93.00	81.70	72.64	617.72	38.37
5	98.80	93.10	66.78	602.38	39.74
10	98.15	92.60	65.42	592.21	38.17
20	95.65	91.18	64.53	584.50	37.46

For m, the trend is more pronounced. Increasing the number of contextual keys from $m = 1$ to $m = 5$ leads to a clear improvement in overall performance, showing that incorporating multiple context-specific keys is beneficial for strengthening the editing-critical subspace. In particular, both effectiveness and generalization improve substantially, and consistency also reaches its best value at $m = 5$. However, further increasing m beyond 5 does not continue to improve the results. Instead, performance begins to decline on several metrics, suggesting that excessive context augmentation may introduce redundancy and weaken the benefit of the most relevant contextual directions. Overall, the results show a clear trend that performance first improves and then degrades as m increases. Based on this observation, we set $m = 5$ as the default choice in all experiments.

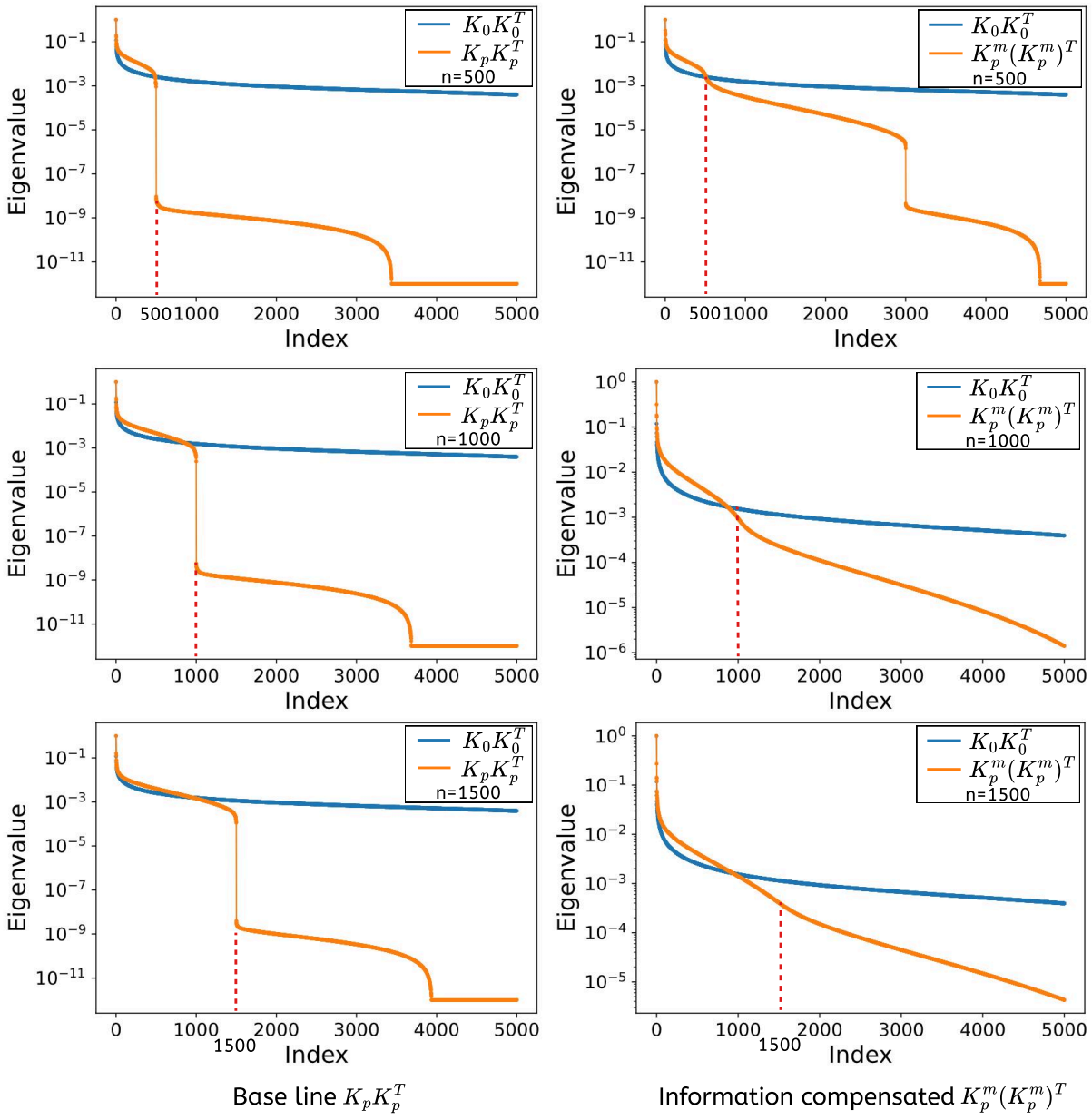

Figure 3: Second-drop index r_2 in the normalized spectrum of $K_pK_p^\top$ tracks the edit volume n stored in K_p. We run three edit volumes $n \in \{500, 1000, 1500\}$ under two conditions: baseline (left column, K_p) and information-compensated (right column, K_p^m). Red dashed lines mark the detected r_2. For each n, the r_2 location is consistent between the two columns, indicating that information compensation does not shift the second drop point.