DO WE REALLY NEED HYPERNETWORKS FOR SEQUENTIAL META MODEL EDITING?

Anonymous authors
Paper under double-blind review

ABSTRACT

Meta-learning-based model editors commonly employ a learned transformation, often implemented by a hypernetwork, to transform the first-step edit gradient before constructing the final parameter update. We ask how much these learned transformations actually contribute to sequential meta model editing. Across multiple meta editors, language models, and datasets, we find limited additional benefit from the learned transformation: untransformed updates remain highly competitive and can even outperform their transformed counterparts. Further analysis shows that much of the editing capability is already supported by the underlying update formulation. Motivated by these findings, we propose **RawEdit**, a hypernetwork-free meta editor that directly uses the raw first-step gradient and strengthens the update formulation through a direct residual and history-aware regularization. RawEdit learns only two scalars per edited layer to control edit strength and regularization while explicitly accounting for previously edited directions. Extensive experiments show that RawEdit consistently achieves strong editing performance across models and datasets, outperforming existing editors by up to 20% in efficacy, generality, and locality, while substantially reducing computational cost and remaining stable under long-term sequential editing with up to 20,000 edits. Moreover, compared with subject-based editors, RawEdit effectively mitigates superficial editing and remains applicable to challenging settings where reliance on subject representations imposes inherent limitations. Code is available at `https://anonymous.4open.science/r/RawEdit-760D`.

1 INTRODUCTION

Large language models often need to incorporate newly acquired knowledge, correct outdated facts, or revise specific behaviors after deployment (Zhang et al., 2023; Wang et al., 2024b). Model editing aims to perform such targeted modifications efficiently while preserving the model's behavior on unrelated inputs (Meng et al., 2022; Mitchell et al., 2022b). Among existing approaches, meta-learning-based editors are particularly appealing because they enable fast editing once the editor has been trained (Mitchell et al., 2022a; Tan et al., 2024; Li et al., 2025b; Ma et al., 2026). A common design in existing meta-learning editors is to transform the raw first-step gradient signal before constructing the final parameter update, with the goal of improving edit generality and locality. Despite the prevalence of this design, its actual contribution has received relatively little scrutiny, as the learned transformation is typically evaluated jointly with the subsequent update construction.

In this work, we disentangle these two components to investigate how much these learned transformations actually contribute to sequential meta model editing. To isolate their contribution, we remove the learned transformation while keeping the original update construction unchanged. Across multiple state-of-the-art editors, language models, and datasets, the resulting untransformed updates achieve strong performance and can even outperform their transformed counterparts. We further compare the updates themselves and find that the transformed and untransformed updates are highly aligned. Removing the additional directional component introduced by the learned transformation also has almost no effect on editing performance. These results suggest that much of the editing capability may already come from the underlying update construction itself. Specifically, editing intermediate-to-late layers helps improve generality, while ridge regularization substantially improves locality. When either of these design choices is weakened, editing performance deteriorates substantially regardless of whether the learned transformation is employed.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

These findings motivate us to shift the design focus from the learned transformation toward the update construction itself. In this paper, we propose **RawEdit**, which makes three key changes. First, it removes the learned transformation and directly uses the raw first-step edit signal. Second, it treats the raw output-gradient signal as the desired residual, rather than deriving the residual from an assumed gradient-form parameter update (Tan et al., 2024; Li et al., 2025b). Third, it introduces history-aware regularization that explicitly discourages new updates from interfering with previously edited directions while also implicitly promoting locality (Fang et al., 2025a; Guo et al., 2025; Fang et al., 2025b; Liu et al., 2026a). RawEdit uses only two learnable scalars per edited layer to control edit strength and regularization, with minimal editor-side overhead.

We evaluate RawEdit on ZsRE (20k edits) and WikiBigEdit (17k edits) across four language models. RawEdit achieves strong performance in edit efficacy, generality, locality, and computational efficiency, while preserving the models' general capabilities throughout long-term sequential editing. Moreover, compared with widely studied subject-based editors (Meng et al., 2022; 2023; Fang et al., 2025a), RawEdit not only effectively mitigates superficial editing and better supports semantic generalization to paraphrastic variations and multi-hop reasoning, but also accommodates a broader range of editing scenarios, including long-form edits and multiple edits that share the same subject but involve different factual relations.

2 Related Work

2.1 Non-Meta-Learning Model Editing

Non-meta-learning model editing broadly includes parameter-preserving approaches and locate-then-edit (LTE) methods. Parameter-preserving methods inject edited knowledge without directly changing pretrained weights. Memory-based approaches (Dong et al., 2022; Zheng et al., 2023; Hartvigsen et al., 2023; Jiang et al., 2024; Bi et al., 2025; Fei et al., 2026) store edited information externally and retrieve it at inference time, while SERAC (Mitchell et al., 2022b), T-Patcher (Huang et al., 2023), MELO (Yu et al., 2024), and WISE (Wang et al., 2024a) introduce auxiliary modules to accommodate new knowledge. These approaches avoid perturbing the original parameters, but their auxiliary capacity typically grows as edits accumulate.

Locate-then-edit methods instead identify internal representations associated with target knowledge and directly modify selected parameters. ROME (Meng et al., 2022) performs rank-one updates to selected MLP layers, while MEMIT (Meng et al., 2023) and EMMET (Gupta et al., 2024) extend this formulation to large-scale batch editing. AlphaEdit (Fang et al., 2025a) and BetaEdit (Liu et al., 2026a) further constrain updates to reduce interference with preserved knowledge. Recent work studies multi-layer editing (Li et al., 2025a; Liu et al., 2026c), conflicting edits (Dai et al., 2026; Dong et al., 2025), long-form knowledge (Jiang et al., 2025; Xu et al., 2026; Tian et al., 2026a), and sequential stability (Gu et al., 2024; Liu et al., 2025; Ma et al., 2025; Li & Chu, 2025; Guo et al., 2025; Liu et al., 2026b; Fang et al., 2025b). However, their reliance on explicitly identified subject-associated representations can restrict applicability to less structured editing scenarios.

2.2 Meta-Learning-Based Model Editing

Meta-learning-based editors learn how to convert an edit signal into a parameter update. MEND (Mitchell et al., 2022a) pioneered this direction by exploiting the rank-one structure of first-step gradients and learning a hypernetwork to transform the corresponding gradient factors. MALMEN (Tan et al., 2024) subsequently introduces a least-squares formulation with closed-form updates for scalable batch editing. Later methods improve the transformation or its training strategy (Tian et al., 2026b; Wang et al., 2026): InstructEdit (Zhang et al., 2024) incorporates instruction-based supervision, RLEdit (Li et al., 2025b) uses reinforcement learning for long-horizon editing, and EMSEdit (Li et al., 2026) introduces multi-step supervision to improve data efficiency. UltraEdit (Gu et al., 2026) and StableEdit (Ma et al., 2026) move away from conventional hypernetwork transformations: UltraEdit learns a normalizer during editing, whereas StableEdit further strengthens this normalization through warm-up and whitening for long-term edit stability.

Despite these advances, existing work has largely focused on improving the transformation mechanism, its training strategy, or the subsequent update construction, while the contribution of the transformation itself has rarely been isolated from the update formulation that follows it. Our work

directly separates these components and shows that much of the editing capability is already supported by the underlying update construction, motivating us to remove the transformation and instead focus on strengthening the update formulation itself.

3 PRELIMINARY

Meta-learning-based model editors construct parameter updates from the first-step gradient induced by an edit example. We first introduce the general meta-editing objective, and then review two key components of existing editors: factorized gradient transformation and ridge regression-based update construction.

Meta-Learning-Based Editing. Consider editing a weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ in a base model f_θ. Given an edit example (x_e, y_e^{new}), where x_e denotes the input and y_e^{new} the desired output, the editing loss is defined as

$$\mathcal{L}_{\text{edit}} = \text{CE}\big(f_\theta(x_e), y_e^{\text{new}}\big), \tag{1}$$

where CE denotes the cross-entropy loss. The corresponding first-step gradient $g_W = \nabla_W \mathcal{L}_{\text{edit}}$ serves as the edit signal.

A meta-learning-based editor applies a learnable transformation h_ϕ, parameterized by ϕ, to the edit signal and subsequently constructs a parameter update through an update rule $\mathcal{U}$:

$$\tilde{g}_W = h_\phi(g_W), \qquad \Delta W = \mathcal{U}(\tilde{g}_W), \tag{2}$$

yielding the edited weight $W' = W + \Delta W$ and the corresponding model $f_{\theta'}$. Here, h_ϕ and $\mathcal{U}$ respectively represent the learned transformation and the subsequent update construction, whose roles will be examined separately in Sec. 4.

The transformation parameters ϕ are optimized through the behavior of the edited model. Specifically, the meta-training objective encourages the target edit to generalize to semantically related inputs while preserving the model's predictions on unrelated inputs:

$$\mathcal{L}_{\text{meta}} = \text{CE}\big(f_{\theta'}(x_g), y_e^{\text{new}}\big) + \lambda_u \, \text{KL}(p_\theta(\cdot \mid x_u) \,\|\, p_{\theta'}(\cdot \mid x_u)), \tag{3}$$

where $x_g \in \mathcal{G}$ is sampled from the generality set $\mathcal{G}$, $x_u \in \mathcal{U}$ is sampled from the locality set $\mathcal{U}$, $p_\theta(\cdot \mid x)$ denotes the model output distribution, and KL denotes the Kullback–Leibler divergence. The coefficient $\lambda_u > 0$ balances generalization and locality preservation.

Factorized Gradient Transformation. Directly transforming the full weight gradient g_W is impractical for large language models, since it scales with the size of the weight matrix W. MEND (Mitchell et al., 2022a) addresses this issue by exploiting the outer-product structure of g_W. For a linear layer $z_o = W z_i$, let $z_i \in \mathbb{R}^{d_{\text{in}}}$ denote its input and $z'_o = \nabla_{z_o} \mathcal{L}_{\text{edit}} \in \mathbb{R}^{d_{\text{out}}}$ the gradient with respect to its output. The corresponding weight gradient can be written as

$$g_W = z'_o z_i^\top. \tag{4}$$

Thus, the two factor vectors z_i and z'_o are sufficient to represent the weight gradient, avoiding the need to use the full gradient matrix as input to the transformation network. Following this factorization, meta editors apply a transformation h_ϕ to the concatenated factor vectors:

$$[\tilde{z}_i; \tilde{z}'_o] = h_\phi([z_i; z'_o]), \tag{5}$$

where $[\cdot;\cdot]$ denotes vector concatenation, and $\tilde{z}_i \in \mathbb{R}^{d_{\text{in}}}$ and $\tilde{z}'_o \in \mathbb{R}^{d_{\text{out}}}$ are the transformed input activation and output-gradient signal, respectively. This reduces the dimensionality handled by the transformation from $d_{\text{in}} \cdot d_{\text{out}}$ to $d_{\text{in}} + d_{\text{out}}$.

Ridge Regression-Based Update. Inspired by the key–value remapping perspective of ROME (Meng et al., 2022), MALMEN (Tan et al., 2024) constructs the parameter update through a ridge regression formulation. Given the edit key z_i and the transformed first-step signals, the resulting update takes the form

$$\Delta W = \tilde{\eta} \, \tilde{z}'_o z_i^\top \left(z_i z_i^\top + \beta I\right)^{-1}, \qquad \tilde{\eta} = \eta \, \tilde{z}_i^\top z_i, \tag{6}$$

where η controls the update scale and $\beta > 0$ is the ridge coefficient. The regularized update formulation helps control interference among edits and preserve locality, which is particularly important in large-scale editing. It has therefore been adopted by subsequent meta-learning-based editing methods (Li et al., 2025b; Gu et al., 2026; Li et al., 2026; Ma et al., 2026).

Table 1: Sequential editing performance of meta-learning-based editors and untransformed updates across different models and datasets.

Method	Dataset	LLaMA3.1-8B			Qwen3-8B			GPT-J-6B			GPT2-XL		
		ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS
RLEdit	ZSRE (20k)	84.4	80.5	32.3	77.8	73.1	54.3	72.1	67.4	56.3	56.0	52.1	79.9
UltraEdit		81.3	75.5	62.7	66.6	62.0	79.5	77.0	71.2	70.1	37.7	35.4	95.2
StableEdit		86.1	81.2	51.8	36.9	35.6	95.4	59.5	55.5	90.2	23.7	22.9	99.4
$\eta = -0.02$		85.3	79.6	56.6	75.9	70.8	69.6	62.8	58.8	82.1	25.1	24.1	98.7
$\eta = -0.05$		86.9	82.6	50.8	81.2	76.3	61.0	76.5	70.9	52.2	35.0	32.8	95.5
$\eta = -0.1$		19.8	17.2	3.2	82.4	78.1	55.2	79.9	74.3	45.2	46.9	43.4	89.5
$\eta = -0.2$		53.7	47.7	4.7	82.6	78.7	47.3	81.2	76.0	40.0	57.8	53.9	75.9
RLEdit	WikiBigEdit (17k)	70.0	64.2	33.6	63.7	57.6	34.5	66.2	60.9	35.4	55.7	49.8	57.5
UltraEdit		76.1	69.2	40.4	58.4	51.7	56.0	74.1	67.0	39.9	37.2	35.6	89.9
StableEdit		77.2	69.7	40.9	32.4	31.2	92.9	60.5	55.3	65.4	25.8	25.0	99.4
$\eta = -0.02$		77.5	70.4	52.9	66.2	60.6	47.6	62.7	57.1	53.8	26.0	25.3	99.0
$\eta = -0.05$		77.4	70.0	45.9	71.1	64.9	43.0	72.6	65.9	41.8	32.8	31.7	92.8
$\eta = -0.1$		17.1	14.2	4.0	70.8	64.5	38.0	75.6	68.4	38.0	45.3	40.9	74.5
$\eta = -0.2$		26.2	21.1	5.3	5.9	4.5	1.3	76.3	68.7	34.5	54.5	49.9	54.0

4 Revisiting Hypernetwork Editing

Meta-learning-based editors typically transform the first-step edit signal before constructing the final parameter update. We examine whether this transformation provides an independent benefit beyond the update formulation that follows it. Our analysis proceeds from editing performance to update geometry, and finally to the structural biases introduced by the update formulation itself.

4.1 Untransformed Updates Are Surprisingly Competitive

As shown in Sec. 3, existing meta editors first transform the input z_i and output gradient z'_o (Eq. 5) and then use the transformed signals to construct the parameter update (Eq. 6). To isolate the effect of this transformation, we replace it with the identity mapping, $\tilde{z}_i = z_i$ and $\tilde{z}'_o = z'_o$, while leaving the subsequent update construction unchanged:

$$\Delta W_{\text{untrans}} = \tilde{\eta}\, z'_o z_i^\top \left(z_i z_i^\top + \beta I\right)^{-1}, \qquad \tilde{\eta} = \eta\, z_i^\top z_i. \tag{7}$$

We refer to Eq. 7 as the *untransformed update*. This controlled comparison changes only the transformation while preserving the original update formulation.

Experimental Setup. We evaluate transformed and untransformed updates on ZSRE (Mitchell et al., 2022a) and WikiBigEdit (Thede et al., 2025), containing 20,000 and 17,000 edits, respectively. Following Ma et al. (2026), edits are applied sequentially in batches of 100. We compare against three state-of-the-art meta editors RLEdit (Li et al., 2025b), UltraEdit (Gu et al., 2026), and StableEdit (Ma et al., 2026) on LLaMA3.1-8B-Instruct, Qwen3-8B, GPT-J-6B, and GPT2-XL-1.5B. For all models, we edit seven late MLP down-projection layers, from the 10th to the 4th layer from the end. Performance is measured by efficacy (ES), generality (GS), and locality (LS), which respectively evaluate the target edit, its transfer to paraphrases, and preservation on unrelated inputs. The formulas for ES, GS, and LS are given in Appendix A.1. Because the untransformed update contains no learned transformation, its behavior is controlled only by η and β. We fix $\beta = 1$, following UltraEdit and StableEdit, and sweep $\eta \in \{-0.02, -0.05, -0.1, -0.2\}$.

Results. Table 1 shows that removing the transformation does not lead to a consistent performance loss. On ZSRE, for example, the untransformed update achieves $86.9/82.6$ ES/GS on LLaMA and $81.2/76.3$ on Qwen with $\eta = -0.05$, matching or exceeding the trained editors. Similar behavior is observed on WikiBigEdit. These results indicate that strong editing performance can be retained without transforming the first-step edit signal.

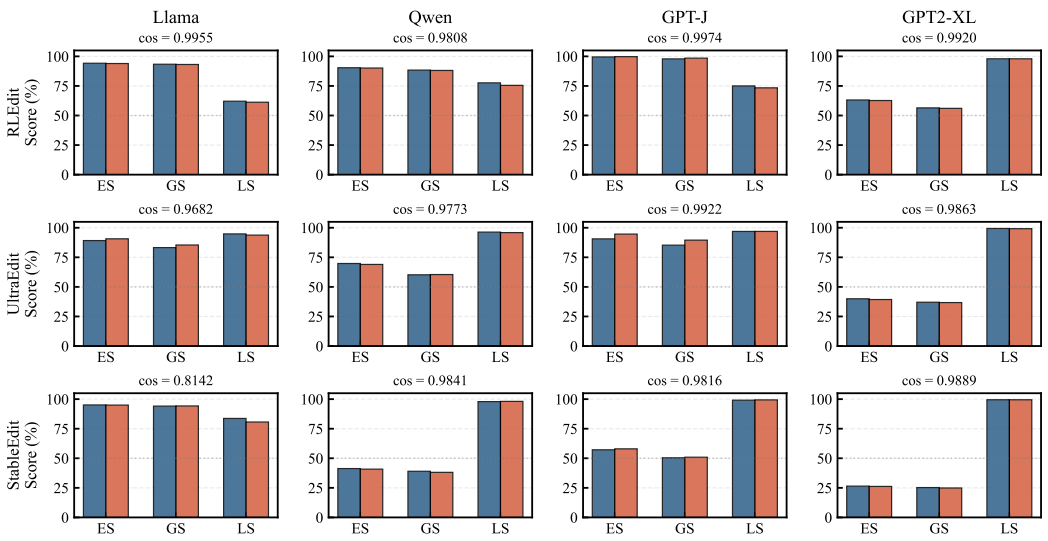


Figure 1: Geometric alignment and projection analysis of transformed and untransformed updates on ZSRE. We report their cosine similarity, together with the editing performance of the original transformed update ΔW_{trans} (blue) and its projection $\Delta W_{\text{trans}}^{\parallel}$ (orange).

4.2 Transformed and Untransformed Updates are Highly Aligned

Given that untransformed updates remain competitive, we next examine whether the transformation meaningfully alters the update direction relative to the untransformed update. For each edit, we construct the transformed update ΔW_{trans} and its untransformed counterpart $\Delta W_{\text{untrans}}$ from the same unedited model, ensuring that both are based on identical z_i and z_o'. We then measure their Frobenius cosine similarity and project the transformed update onto the untransformed direction:

$$\begin{aligned} \cos\left(\Delta W_{\text{untrans}}, \Delta W_{\text{trans}}\right) &= \frac{\langle \Delta W_{\text{untrans}}, \Delta W_{\text{trans}} \rangle_F}{\|\Delta W_{\text{untrans}}\|_F \, \|\Delta W_{\text{trans}}\|_F}, \\ \Delta W_{\text{trans}}^{\parallel} &= \frac{\langle \Delta W_{\text{untrans}}, \Delta W_{\text{trans}} \rangle_F}{\|\Delta W_{\text{untrans}}\|_F^2} \Delta W_{\text{untrans}}. \end{aligned} \tag{8}$$

The projection retains the component of ΔW_{trans} along the untransformed direction while removing the additional directional component introduced by the transformation. Comparing their editing performance therefore tests whether this additional component is functionally important.

Figure 1 shows that transformed and untransformed updates are highly aligned across editors and models. On GPT-J, their cosine similarities reach 0.9974, 0.9922, and 0.9816 for RLEdit, UltraEdit, and StableEdit, respectively. More importantly, removing the orthogonal component causes almost no performance degradation and can even improve it in some cases. In Appendix B.1, we observe the same pattern on WikiBigEdit, and factor-level comparisons between z_i and $\tilde{z}_i$, as well as between z_o' and $\tilde{z}_o'$, likewise show high cosine similarity. These results suggest that the transformation contributes little functionally useful directional information beyond the untransformed update.

4.3 The Update Formulation Provides Critical Editing Biases

The comparable performance of transformed and untransformed updates naturally shifts our attention from the hypernetwork transformation to the update construction itself. We therefore examine factors in the update formulation that are most directly tied to the two meta-editing objectives in Eq. 3: generalization and locality.

Editing Location Strongly Affects Generalization. Intermediate-to-late Transformer layers generally encode higher-level semantic representations (Valeriani et al., 2023; Skean et al., 2025), suggesting that the location of an update can directly affect its semantic transfer. We test this by applying

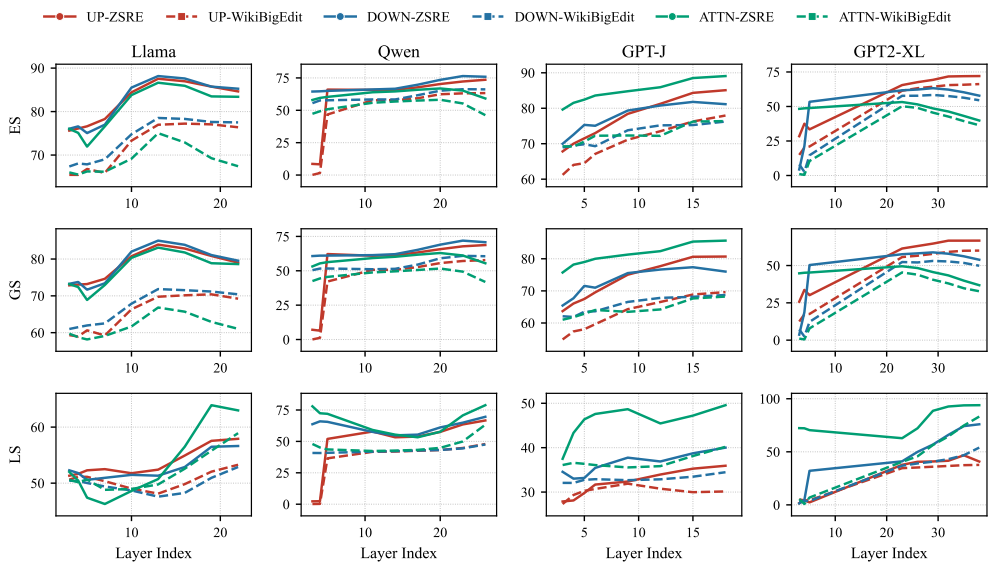


Figure 2: Effect of editing location on untransformed updates across four language models and three module types: MLP down projection, MLP up projection, and attention output projection.

Table 2: Effect of removing ridge regression from transformed and untransformed editors on ZSRE.

Method	LLaMA3.1-8B			Qwen3-8B			GPT-J-6B			GPT2-XL		
	ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS
RLEdit	62.4	58.7	11.7	55.0	51.4	14.9	51.7	50.0	45.3	62.1	58.7	38.6
UltraEdit	63.8	60.5	24.3	54.4	43.7	31.1	61.3	57.9	43.0	38.2	36.7	90.1
StableEdit	66.5	61.4	21.4	40.2	38.0	81.6	33.4	31.0	76.6	27.0	25.8	94.1
$\eta = -0.02$	65.1	61.7	22.6	55.6	51.8	28.5	53.5	47.3	71.5	25.1	24.1	98.7
$\eta = -0.05$	45.6	42.2	7.8	43.6	41.3	21.0	59.7	53.8	41.2	38.0	34.2	92.5
$\eta = -0.1$	0.0	0.0	0.0	32.8	29.6	9.4	62.4	54.7	29.6	56.8	52.4	72.1
$\eta = -0.2$	0.0	0.0	0.0	0.0	0.0	0.0	51.7	48.8	11.0	58.1	52.8	40.9

the untransformed update to the MLP down projection, MLP up projection, and attention output projection while varying the edited layer from early to late positions. We use $\eta = -0.02$ for LLaMA and Qwen and $\eta = -0.2$ for GPT-J and GPT2-XL.

As shown in Figure 2, early-layer editing consistently yields substantially weaker generalization, whereas editing at intermediate-to-late layers performs markedly better across models, datasets, and module types. On ZSRE, for example, the GS of the MLP down projection increases from 73.3 to 85.0 on LLaMA and from 60.7 to 71.9 on Qwen. The same trend holds for RLEdit, UltraEdit, and StableEdit, as reported in Appendix B.2, helping explain why existing meta editors commonly apply updates to intermediate-to-late layers.

Ridge Regression Is Critical for Locality Preservation. To examine the effect of ridge regression, we remove it from the ridge-based update ΔW_{ridge}, yielding the MEND-style outer-product update $\Delta W_{\text{outer}} = \tilde{\eta}\, \tilde{z}_o' z_i^\top$. For a fair comparison, we control for update scale by rescaling ΔW_{outer} to match the Frobenius norm of the corresponding ridge-based update:

$$\Delta W_{\text{outer}}^{\text{norm}} = \frac{\|\Delta W_{\text{ridge}}\|_F}{\|\Delta W_{\text{outer}}\|_F} \Delta W_{\text{outer}}. \tag{9}$$

The two updates $\Delta W_{\text{outer}}^{\text{norm}}$ and ΔW_{ridge} therefore have the same magnitude and differ only in direction introduced by the ridge regression. Table 2 shows that removing ridge regression still causes substantial degradation for both transformed and untransformed editors, particularly in locality. Similar results on WikiBigEdit are reported in Appendix B.3.

Overall, these experiments show that the update formulation supplies critical biases for both generalization and locality. Weakening either the editing location or ridge regularization substantially degrades performance even when the transformation is employed. This helps explain why untransformed updates can remain highly effective.

5 RawEdit: Editing with the Raw Edit Signal

Section 4 shows no clear performance advantage from transforming the raw edit signal, while highlighting the critical role of the update formulation. Motivated by these findings, we propose **RawEdit**, a hypernetwork-free meta editor that directly uses the raw edit signal and strengthens the update formulation through direct residual fitting and history-aware regularization.

For the t-th edit, let $z_{i,t}$ and $z'_{o,t}$ denote the weight's input and output gradient, respectively, and let $Z_{i,<t} = [z_{i,1}, \ldots, z_{i,t-1}]$ collect the weight input of previous edits. Inspired by the key–value fitting perspective of locate-then-edit methods (Meng et al., 2023; Fang et al., 2025a; Liu et al., 2026a), RawEdit optimizes the following objective

$$\Delta W_t^* = \arg\min_{\Delta W_t} \left\| \Delta W_t z_{i,t} - \eta z'_{o,t} \right\|_2^2 + \left\| \Delta W_t Z_{i,<t} \right\|_F^2 + \beta \left\| \Delta W_t \right\|_F^2, \tag{10}$$

where the three terms respectively fit the current edit, preserve previous edits, and regularize the weight perturbation. The resulting closed-form update is

$$\Delta W_t^* = \eta \, z'_{o,t} z_{i,t}^\top \left(z_{i,t} z_{i,t}^\top + Z_{i,<t} Z_{i,<t}^\top + \beta I \right)^{-1}. \tag{11}$$

RawEdit contains only two learnable scalars for each edited layer, η and β, which are optimized using the same meta-training protocol as RLEdit (Li et al., 2025b).

Compared with the widely used ridge formulation in Eq. 6, RawEdit makes three changes. First, it removes the transformation and directly uses the raw first-step edit signal. Second, it incorporates $Z_{i,<t} Z_{i,<t}^\top$ to discourage new updates from interfering with previous edits. Third, RawEdit directly uses $\eta z'_{o,t}$ as the desired residual. In contrast, Eq. 6 uses $\eta (z_{i,t}^\top z_{i,t}) z'_{o,t}$, which is derived by first assuming that the parameter update follows the gradient form (Appendix C.1). RawEdit avoids this intermediate assumption and specifies the residual directly from the raw output-gradient signal.

6 Experiments

6.1 Experimental Setup

We follow the experimental setup in Section 4, including the four language models, two editing benchmarks, sequential editing protocol, edited layers, and the standard efficacy, generality, and locality metrics. In addition to the meta-learning editors considered previously, we include regularized fine-tuning (RegFT) and three state-of-the-art subject-based editors as additional baselines: AlphaEdit (Fang et al., 2025a), BetaEdit (Liu et al., 2026a), and SPHERE (Liu et al., 2026b). Descriptions of these baselines are provided in Appendix A.2, and training details of RawEdit are provided in Appendix C.2.

We organize our evaluation around three research questions (RQs): (1) How effective, efficient, and stable is RawEdit? (2) Can RawEdit handle more challenging editing scenarios beyond standard short-form factual edits? (3) Does RawEdit mitigate superficial editing?

6.2 How Effective, Efficient, and Stable Is RawEdit?

Table 3 reports long-term sequential editing results against the four non-meta-learning baselines. RawEdit consistently achieves the strongest overall editing performance across models and datasets. It is also substantially more efficient, requiring only about 0.07 seconds per edit, roughly 30–$70\times$ faster than the baselines. This efficiency stems from directly constructing the update from the first-step edit signal (Sec.3), avoiding the iterative optimization required by the compared non-meta-learning methods. The efficiency comparison with baseline meta editors is provided in Appendix D.1. Table 3 also evaluates the two main design choices of RawEdit: **w/ GIR** replaces the

Table 3: Sequential editing performance and efficiency on ZsRE and WikiBigEdit. Time denotes the average wall-clock time per edit across four models. **w/ GIR** uses the gradient-induced residual, while **w/o HAR** removes history-aware regularization.

Method	Dataset	LLaMA3.1-8B			Qwen3-8B			GPT-J-6B			GPT2-XL			Time (s)
		ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS	
RegFT	ZsRE (20k)	41.1	39.4	28.1	37.5	35.1	33.4	43.2	41.6	19.5	45.0	44.3	24.8	2.43
AlphaEdit		86.8	81.2	33.3	97.8	89.2	54.4	71.1	63.2	60.0	21.0	17.6	11.8	4.10
BetaEdit		83.5	76.7	51.2	93.3	83.6	60.1	70.5	58.1	65.3	53.5	38.9	42.3	4.22
SPHERE		81.8	75.9	45.1	91.0	82.7	58.9	68.9	60.1	64.4	48.1	32.7	36.2	4.12
RawEdit		88.7	81.2	61.1	80.9	73.3	67.7	87.2	77.0	59.9	63.9	58.5	50.6	0.07
w/ GIR		85.7	79.7	60.1	78.1	71.1	68.0	83.7	71.1	60.6	63.5	59.1	58.9	0.07
w/o HAR		86.2	80.3	56.7	79.7	73.2	63.7	81.9	73.4	51.8	64.3	60.9	45.8	0.07
RegFT	WikiBigEdit (17k)	48.0	43.5	36.9	38.7	32.8	26.5	35.2	31.8	23.4	34.8	32.8	26.2	2.62
AlphaEdit		47.3	41.2	21.9	81.7	62.0	33.7	67.9	54.5	30.7	18.0	13.1	8.0	4.63
BetaEdit		68.6	52.8	52.6	81.0	64.6	48.9	69.2	55.5	48.4	52.8	41.1	38.3	4.86
SPHERE		61.8	49.6	47.7	78.0	59.8	42.8	68.5	55.9	47.7	45.8	37.9	34.6	4.67
RawEdit		85.1	75.3	56.3	70.6	61.4	50.9	81.7	70.6	46.3	56.1	49.6	72.7	0.07
w/ GIR		78.1	70.3	55.2	67.6	60.5	50.1	75.3	68.6	45.2	54.2	47.3	70.2	0.07
w/o HAR		80.7	71.6	50.5	67.8	60.1	47.4	75.2	68.0	39.2	55.3	48.8	57.1	0.07

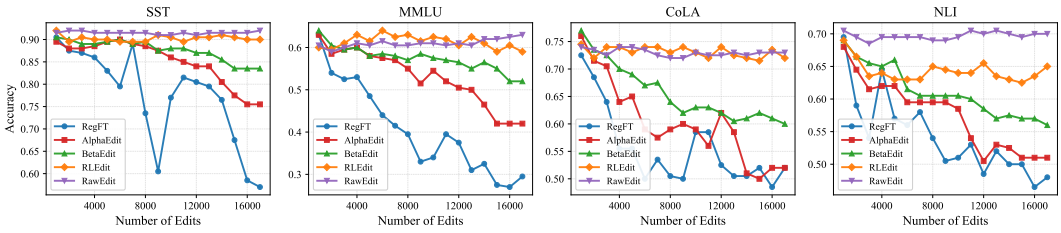


Figure 3: General capabilities during 17,000 sequential edits on WikiBigEdit with LLaMA3.1-8B, evaluated every 1,000 edits on SST, MMLU, CoLA, and NLI.

direct residual with the conventional gradient-induced residual, while **w/o HAR** removes history-aware regularization. Both variants consistently degrade performance, confirming the benefits of directly fitting the raw edit signal without assuming a particular gradient-form parameter update, and of incorporating editing history into the update formulation to preserve previous edits while also promoting locality (Fang et al., 2025a; Guo et al., 2025; Fang et al., 2025b; Liu et al., 2026a). We also incorporate our strengthened update formulation into existing meta editors and observe consistent improvements; detailed results are reported in Appendix D.2.

Beyond the standard locality metric, we further evaluate whether editing degrades the model's general capabilities using GLUE benchmarks. Figure 3 reports the results on LLaMA and WikiBigEdit. RawEdit preserves these capabilities well throughout sequential editing, consistent with its strong locality performance. Benchmark details and additional results are provided in Appendix D.3.

6.3 Can RawEdit Handle More Challenging Editing Scenarios?

We further evaluate RawEdit in three challenging editing scenarios that extend beyond standard factual editing benchmarks, which typically involve short target outputs, distinct subjects across edits, and generalization primarily through paraphrased queries. First, beyond the standard GS reported in Table 3, we evaluate broader semantic generalization on WikiBigEdit using persona-style queries (PA) and multi-hop reasoning (MH). Second, in the **same-subject** setting, we perform batch editing with 100 edits, where the same subject may appear in 2–5 different edits. Third, in the **long-form** setting, each desired output contains approximately 200 tokens. Dataset details and examples are provided in Appendix A.3.

Table 4: Performance on broader semantic generalization, same-subject editing, and long-form editing. PA and MH denote persona-style and multi-hop generalization; mES/mGS/mLS and lES/lGS/lLS denote ES/GS/LS under the same-subject and long-form settings, respectively.

Method	LLaMA3.1-8B								Qwen3-8B							
	PA	MH	mES	mGS	mLS	lES	lGS	lLS	PA	MH	mES	mGS	mLS	lES	lGS	lLS
RegFT	41.3	30.3	85.7	82.3	71.4	97.5	96.5	90.3	26.6	23.0	97.2	89.5	37.3	99.5	98.3	81.6
AlphaEdit	30.1	11.0	68.7	55.1	91.0	81.1	77.1	100	42.1	27.4	46.8	35.4	99.0	65.7	65.2	100
BetaEdit	33.5	12.2	68.2	55.3	93.7	74.1	70.1	100	40.0	26.6	47.0	36.1	98.6	67.7	64.7	100
RLEdit	58.3	23.7	78.3	72.7	83.8	90.2	88.8	94.0	47.9	25.3	81.7	73.5	66.9	96.6	92.1	86.9
UltraEdit	61.9	28.5	72.4	68.1	85.2	87.7	88.1	94.9	38.0	28.4	79.8	72.1	73.7	94.1	91.7	88.1
RawEdit	63.6	33.7	81.5	77.8	83.1	90.2	89.5	94.2	46.2	30.0	83.7	75.8	72.5	96.3	93.0	87.7

Table 4 reports results on LLaMA and Qwen, with results on GPT-J and GPT2-XL provided in Appendix D.4. RawEdit performs strongly across all three settings, whereas subject-based editors show clear limitations. Recent methods have also been developed specifically for same-subject and long-form editing; comparisons with these specialized methods are provided in Appendix D.5.

6.4 Does RawEdit Mitigate Superficial Editing?

Model editing aims to replace the model's original output y_e^{old} with the desired output y_e^{new}. However, recent work (Xie et al., 2025) shows that many existing editors may establish only a superficial association with the new output: even after a successful edit, the model can revert to y_e^{old} under carefully constructed contexts. Following Xie et al. (2025), we evaluate this behavior using the *Rep* attack, which repeatedly inserts y_e^{old} into the input context and tests whether the edited model is pulled back toward the original output. Details of the *Rep* attack and the evaluation dataset are provided in Appendix A.3.

Table 5: Evaluation of superficial editing under the *Rep* attack.

Model	Method	ES	OM
LLaMA	Reg-FT	100	5.7
	AlphaEdit	100	68.2
	RawEdit	100	11.5
Qwen	Reg-FT	100	3.7
	AlphaEdit	100	63.5
	RawEdit	100	10.1

As shown in Table 5, although all methods achieve 100% edit efficacy, the subject-based editor AlphaEdit frequently recovers the original output under the *Rep* attack, whereas RawEdit achieves a substantially lower OM. Here, OM measures how often the original output y_e^{old} is assigned the highest output probability after the attack, with lower values indicating less superficial editing.

7 Conclusion and Future Work

In this paper, we disentangle the learned transformation from the subsequent update formulation in sequential meta model editing and examine their contributions separately. Across multiple state-of-the-art meta editors, large language models, and datasets, we find that learned transformations provide limited additional empirical benefit, whereas the update formulation plays a critical role in editing performance. Motivated by these findings, we propose **RawEdit**, which removes the learned transformation and strengthens the update formulation through direct residual fitting and history-aware regularization. RawEdit achieves strong editing performance with minimal editor-side overhead while remaining effective across challenging editing scenarios.

The scope of our work is specific to sequential meta model editing. As shown in Appendix B.4, under a non-sequential single-edit setting with a MEND-style update, the learned transformation provides a clear advantage over its untransformed counterpart, although this advantage is substantially reduced when a ridge-regression-based update is adopted. Our experiments thus reveal a clear contrast between the sequential and non-sequential settings, but do not yet explain why this contrast arises. One possible explanation is that the continually evolving model state in sequential editing makes non-trivial transformations harder to learn, favoring near-identity solutions. We leave a systematic investigation of this mechanism to future work.

AI USE STATEMENT

Generative AI tools were used only for language polishing and improving the clarity and readability of the manuscript. All AI-assisted revisions were manually reviewed by the authors, and the technical content, claims, equations, experimental results, and citations were verified by the authors. We take full responsibility for the final content of this work, including any text produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information and resources necessary to reproduce the main results of this work. The main paper describes the datasets and base models, the sequential editing protocol, the edited modules and layer configurations, the evaluation metrics, and the configurations of the compared baselines. Detailed descriptions and update formulations of the meta-learning-based and subject-based baselines, together with links to their original implementations, are provided in Appendix A. The appendix further reports the implementation details and hyperparameter settings of RawEdit, including its training procedure and editing configuration, as well as additional experimental settings for the ablation studies and diagnostic analyses. Additional results and analyses that complement the main experiments are also included in the appendix.

To facilitate direct reproduction, we release an anonymous implementation of RawEdit together with the scripts and configurations used to run the experiments and evaluations reported in this paper. The anonymous repository is available at `https://anonymous.4open.science/r/RawEdit-760D`.

REFERENCES

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)*, pp. 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*, pp. 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. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7952–7960, 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. Scaling knowledge editing in LLMs to 100,000 facts with neural KV database. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=Z0CX62CSJQ`.

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*, pp. 16801–16819, 2024.

Xiaojie Gu, Ziying Huang, Jia-Chen Gu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in language models. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=GoJLp3BlRV.

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

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

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.

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*, 2025.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, et al. Learning to edit: Aligning llms with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4689–4705, 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)*, pp. 4127–4149, 2025.

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

Xiaopeng Li, Shasha Li, Xi Wang, Shezheng Song, Bin Ji, Shangwen Wang, Jun Ma, Xiaodong Liu, Mina Liu, and Jie Yu. Emsedit: Efficient multi-step meta-learning-based model editing. In *Proceedings of the ACM Web Conference 2026*, pp. 3847–3858, 2026.

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*, 2025b. URL https://openreview.net/forum?id=1jUXprrfcb.

Bingqing Liu, Wei Liu, and Yuhua Li. Betaedit: Null-space constrained sequential model editing. *arXiv preprint arXiv:2605.09285*, 2026a.

Jinzhe Liu, Junshu Sun, Shufan Shen, Chenxue Yang, and Shuhui Wang. Edit less, achieve more: Dynamic sparse neuron masking for lifelong knowledge editing in LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 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*, 2026b. URL https://openreview.net/forum?id=CHsdtzCip6.

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*, 2026c.

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.

Xin Ma, Wei Chen, Qi Liu, Derong Xu, Zhi Zheng, Tong Xu, and Enhong Chen. More edits, more stable: Understanding the lifelong normalization in sequential model editing. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=OcCpVIcgaY.

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*, pp. 15817–15831. PMLR, 2022b.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann Lecun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55854–55875. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/skean25a.html.

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. URL https://openreview.net/forum?id=L6L1CJQ2PE.

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.

Bowen Tian, Caixue He, Jiemin Wu, Jingying Wang, Wenshuo Chen, Zexi Li, and Yutao Yue. Anyedit++: Adaptive long-form knowledge editing via bayesian surprise. In *Forty-third International Conference on Machine Learning*, 2026a.

Shiqiang Tian, Cheng Ding, Qin Chen, Jie Zhou, and Liang He. Tamedit: Trajectory-aware meta-learning for specificity-preserving continual knowledge editing. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21417–21439, 2026b.

Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36:51234–51252, 2023.

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, 2024a.

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, 2024b.

Yangfan Wang, Tianyang Sun, Chen Tang, Jie Liu, Wei Cai, and Jingchi Jiang. Hiedit: Lifelong model editing with hierarchical reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39924–39942, 2026.

Jiakuan Xie, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Revealing the deceptiveness of knowledge editing: A mechanistic analysis of superficial editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17756–17780, 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.

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, pp. 19449–19457, 2024.

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

Zihan Zhang, Meng Fang, Ling Chen, Mohammad-Reza Namazi-Rad, and Jun Wang. How do large language models capture the ever-changing world knowledge? a review of recent advances. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 8289–8311, 2023.

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

A Additional Experimental Details

A.1 Evaluation Metrics for Efficacy, Generality, and Locality

We follow the standard evaluation protocol of model editing and report efficacy (ES), generality (GS), and locality (LS). Let $\hat{y}_\theta(x)$ denote the prediction of model f_θ on input x. For a set of N edits, ES measures whether the edited model produces the desired target on the original edit input:

$$\mathrm{ES} = \frac{1}{N}\sum_{e=1}^{N} \mathbb{1}\left[\hat{y}_{\theta'}(x_e) = y_e^{\mathrm{new}}\right]. \tag{12}$$

GS evaluates whether the same edit generalizes to a semantically equivalent input $x_{g,e}$:

$$\mathrm{GS} = \frac{1}{N}\sum_{e=1}^{N} \mathbb{1}\left[\hat{y}_{\theta'}(x_{g,e}) = y_e^{\mathrm{new}}\right]. \tag{13}$$

LS measures whether predictions on unrelated inputs remain unchanged:

$$\mathrm{LS} = \frac{1}{N}\sum_{e=1}^{N} \mathbb{1}\left[\hat{y}_{\theta'}(x_{u,e}) = \hat{y}_\theta(x_{u,e})\right], \tag{14}$$

where $x_{u,e}$ is an unrelated locality input.

A.2 Baseline Details

Meta-learning-based Editors. We compare RawEdit with three state-of-the-art meta-learning-based editors: RLEdit (Li et al., 2025b), UltraEdit (Gu et al., 2026), and StableEdit (Ma et al., 2026). Following the formulation in Sec.3, let z_i and z'_o denote the input activation and output gradient of the edited module, respectively, and let $\tilde{z}_i$ and $\tilde{z}'_o$ denote their transformed counterparts. All three methods construct the final parameter update through ridge regression, but differ in how the editing signals are transformed and used to define the desired residual.

Table 6: Dataset examples of the additional evaluation settings, including broader semantic generalization, same-subject editing, long-form editing, and superficial editing under the *Rep* attack.

Setting	Example
Broader semantic generalization	**Edit:** Who is the male parent of Karl Heinrich August von Lindenau? → Heinrich Gottlieb Lindenau. **Paraphrase:** Who is Karl Heinrich August von Lindenau's father? **Persona:** Arrr! Who be the swashbucklin' buccaneer that sired Karl Heinrich August von Lindenau? **Multi-hop:** What language is spoken by the father of Karl Heinrich August von Lindenau? → German.
Same-subject	**Subject:** Andrew Bennison. **Edit 1:** Where was Andrew Bennison born? → Oakland. **Edit 2:** When did Andrew Bennison die? → 7 January 1942.
Long-form	**Input:** Give me an introduction to Ada Lovelace. **Paraphrase:** Please introduce Ada Lovelace to me. **Target:** Ada Lovelace was an English mathematician and writer whose ideas helped shape the earliest vision of computer programming. Working with Charles Babbage in the nineteenth century, she studied his proposed Analytical Engine and recognized that such a machine could manipulate symbols as well as numbers. . . . *(approximately 200 tokens)*.
Superficial editing	**Edit:** The mother tongue of Danielle Darrieux is → English (original output: French). ***Rep* attack:** French French French French French French French French French French The mother tongue of Danielle Darrieux is

RLEdit employs a learned hypernetwork to transform the editing signals and follows the MALMEN-style update formulation. Specifically, it defines the desired residual as

$$r_{\mathrm{RL}} = \eta(\tilde{z}_i^\top z_i)\tilde{z}_o', \tag{15}$$

where η controls the edit strength.

UltraEdit does not employ a conventional hypernetwork. Instead, it continually fits a normalizer during sequential editing to transform the editing signals (z_i, z_o') into $(\tilde{z}_i, \tilde{z}_o')$. Unlike RLEdit, UltraEdit uses the squared norm of the transformed input to define the desired residual:

$$r_{\mathrm{Ultra}} = \eta(\tilde{z}_i^\top \tilde{z}_i)\tilde{z}_o'. \tag{16}$$

The factor $\|\tilde{z}_i\|_2^2$ adaptively scales the edit strength according to the saliency of the transformed representation, assigning larger updates to more salient editing signals while down-weighting weak or noisy ones (Gu et al., 2026).

StableEdit further improves the transformation in UltraEdit by introducing a warm-up stage and replacing element-wise standardization with full whitening. For the concatenated editing feature $q = [z_i; z_o']$, it maintains running estimates of the mean μ and covariance matrix Σ and applies

$$\hat{q} = \Sigma^{-\frac{1}{2}}(q - \mu) = [\tilde{z}_i; \tilde{z}_o']. \tag{17}$$

Following UltraEdit, the transformed signals are then used to construct the residual

$$r_{\mathrm{Stable}} = \eta(\tilde{z}_i^\top \tilde{z}_i)\tilde{z}_o', \tag{18}$$

The warm-up stage provides more reliable initial statistics for whitening before sequential editing begins.

After constructing the method-specific residuals, all three editors obtain the final parameter update through ridge regression. Let Z collect the original input activations z_i and R collect the corresponding residuals defined above. The update is

$$\Delta W = RZ^\top \left(ZZ^\top + \beta I\right)^{-1}, \tag{19}$$

where $\beta > 0$ is the ridge coefficient.

For a fair comparison across methods, we standardize the editing locations and edit seven late MLP down-projection layers for all models, ranging from the 10th to the 4th layer from the end. All other settings follow the original implementations and configurations. Implementations of all three baselines are publicly available and have been integrated into the StableEdit repository: `https://github.com/MINE-USTC/StableEdit`.

Subject-Based Editors. We compare against three state-of-the-art subject-based editors: AlphaEdit (Fang et al., 2025a), BetaEdit (Liu et al., 2026a), and SPHERE (Liu et al., 2026b). These methods follow the locate-then-edit formulation (Meng et al., 2023), where W denotes the weight matrix to be edited, K_0 represents the keys associated with pretrained knowledge, and (K_t, V_t) denote the key and target-value representations of the knowledge introduced at editing step t.

AlphaEdit constrains parameter updates to the null space of the pretrained knowledge. Let P denote the corresponding null-space projection matrix, K_1 the keys of the current edits, K_p the keys accumulated from previous edits, and $R = V_1 - WK_1$ the residual for the current edits. Its sequential update is

$$\Delta W_{\text{AlphaEdit}} = RK_1^\top P \left(K_p K_p^\top P + K_1 K_1^\top P + I\right)^{-1}, \tag{20}$$

where I is the identity matrix.

BetaEdit extends null-space editing by explicitly accounting for knowledge leakage and incorporating editing history into the protected representations. At step t, it applies

$$\Delta W_t = R_t K_t^\top P_t \left[\lambda_2 I + \left(K_t K_t^\top + C^{(t)}\right) P_t\right]^{-1}, \qquad C^{(t)} = \lambda_1 K_0 K_0^\top + \sum_{i=1}^{t-1} K_i K_i^\top, \tag{21}$$

where $R_t = V_t - WK_t$ is the current residual, $C^{(t)}$ aggregates pretrained and previously edited knowledge, P_t projects onto the approximate null space of $C^{(t)}$, and λ_1 and λ_2 control knowledge preservation and numerical regularization, respectively.

SPHERE instead regularizes the geometry of the parameter update. Given the edited weight matrix $W \in \mathbb{R}^{n\times d}$, it collects the top principal eigenvectors of $W^\top W$ into $U \in \mathbb{R}^{d\times r}$ and constructs

$$P_\perp = I - \alpha U U^\top, \qquad \widehat{W} = W + \Delta W P_\perp, \tag{22}$$

where ΔW is the update produced by the underlying editor, $P_\perp$ suppresses its components along the principal weight directions, and α controls the suppression strength.

We follow the original implementations and configurations of these methods. Their released implementations do not specify dedicated layer configurations for LLaMA3.1-8B or Qwen3-8B, so we adopt the layer setting used for LLaMA3-8B in the original implementation and edit MLP down-projection layers 4–8 for both models. For GPT-J-6B, we edit layers 3–8, and for GPT2-XL, layers 13–17. The official implementations are available at `https://github.com/jianghoucheng/AlphaEdit`, `https://github.com/lbq8942/BetaEdit`, and `https://github.com/PlusLabNLP/SPHERE`, respectively.

Regularized Fine-Tuning (RegFT). We implement a LoRA-based regularized fine-tuning baseline using the same edited layers as the subject-based editors above. For each edit batch, a new LoRA module with rank 8 is initialized and optimized on the current batch. In addition to the editing loss, we penalize the deviation of the edited weights from their values at the beginning of the batch:

$$\mathcal{L}_{\text{RegFT}} = \mathcal{L}_{\text{edit}} + 0.1 \sum_{\ell} \left\| W_\ell - W_\ell^{(0)} \right\|_F^2, \tag{23}$$

where ℓ indexes the edited layer and $W_\ell^{(0)}$ denotes its weight at the beginning of the current batch. After optimization, the LoRA parameters are merged into the base weights, and a new LoRA module is initialized for the next batch.

A.3 Challenging Editing Scenarios

Standard factual editing benchmarks mainly evaluate short target outputs, distinct subjects across edits, and generalization to paraphrased queries. We construct three additional settings to evaluate broader semantic generalization and more challenging forms of editing.

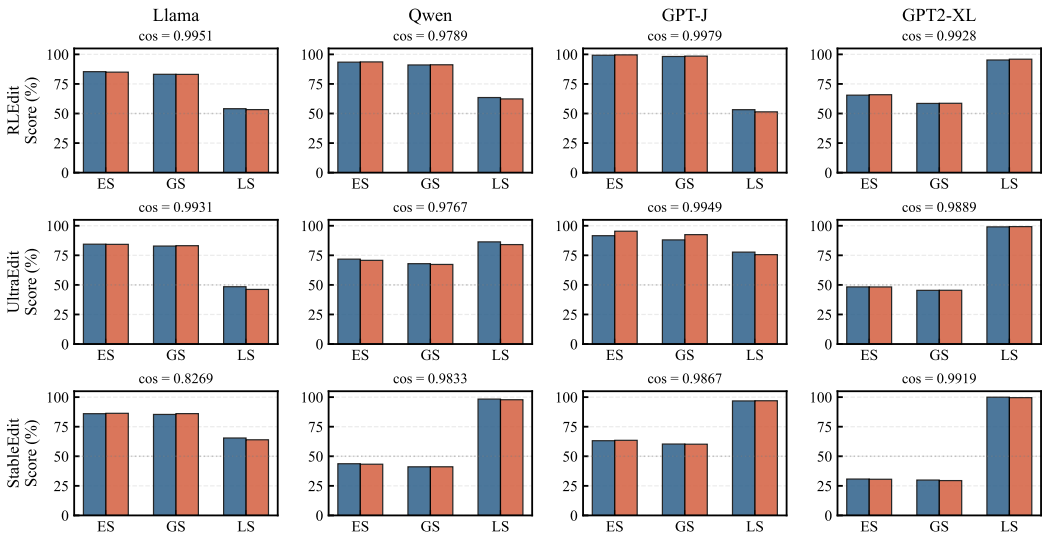


Figure 4: Geometric alignment and projection analysis of transformed and untransformed updates on WikiBigEdit. We report their cosine similarity, together with the editing performance of the original transformed update ΔW_{trans} (blue) and its projection $\Delta W_{\text{trans}}^{\parallel}$ (orange).

Broader Semantic Generalization. Beyond the standard paraphrase-based GS metric, WikiBigEdit provides two additional forms of generalization. **Persona generalization (PA)** rephrases the query in a substantially different linguistic style while preserving its semantic content. **Multi-hop reasoning (MH)** requires the model to use the edited fact as an intermediate step for answering a related question. PA and MH are evaluated on the same model after the full WikiBigEdit sequential editing process.

Same-Subject Editing. We construct 1,000 edits in which subjects are deliberately shared across different factual relations. The edits are divided into 10 independent batches of 100 edits each. Within each batch, 20 subjects appear in two edits, 10 appear in three edits, 5 appear in four edits, and 2 appear in five edits. All 100 edits in a batch are applied jointly, and the 10 batches are evaluated independently. We report the average efficacy, generality, and locality across the 10 batches, denoted as mES, mGS, and mLS.

Long-Form Editing. We construct 100 edits whose desired outputs contain approximately 200 tokens, substantially longer than the typically fewer than 10 tokens in ZSRE and WikiBigEdit. Each edit is applied independently, and the results are averaged over the 100 examples. We denote the corresponding metrics as lES, lGS, and lLS.

Superficial Editing. Following Xie et al. (2025), we construct a CF-a evaluation set for each model from CounterFact (Meng et al., 2023). Since the *Rep* attack requires the model to correctly recognize the original knowledge before editing, we first filter CounterFact examples for which the unedited model predicts the original target y_e^{old} correctly. We then sample 500 such examples separately for LLaMA3.1-8B and Qwen3-8B to form their respective CF-a sets. Each example is edited independently in a single-edit setting, allowing the target edit to be sufficiently established before testing whether it remains vulnerable to superficial reversal. After a successful edit changes the prediction from y_e^{old} to y_e^{new}, the *Rep* attack repeatedly inserts y_e^{old} into the input context and queries the edited model again, testing whether repeated exposure to the original answer can pull the model back toward its pre-edit prediction. We report ES and original-output matching (OM), where OM measures how often y_e^{old} receives the highest output probability after the attack; lower OM indicates less superficial editing. Both metrics are averaged over the 500 examples.

Representative examples of the three settings are shown in Table 6.

Table 7: Cosine similarity between the original and transformed input activations (z_i vs. $\tilde{z}_i$) and output gradients (z'_o vs. $\tilde{z}'_o$). Values are reported as mean $\pm$ standard deviation.

Model	Dataset	Editor	Input cosine	Output-gradient cosine
Llama 3.1	ZSRE	StableEdit	0.93 ± 0.02	0.84 ± 0.00
		UltraEdit	0.93 ± 0.02	0.88 ± 0.01
		RLEdit	0.95 ± 0.01	0.86 ± 0.01
	WikiBigEdit	StableEdit	0.95 ± 0.01	0.90 ± 0.00
		UltraEdit	0.95 ± 0.01	0.94 ± 0.01
		RLEdit	0.95 ± 0.01	0.82 ± 0.02
Qwen3	ZSRE	StableEdit	0.93 ± 0.01	0.87 ± 0.01
		UltraEdit	0.93 ± 0.01	0.84 ± 0.02
		RLEdit	0.93 ± 0.01	0.78 ± 0.03
	WikiBigEdit	StableEdit	0.93 ± 0.01	0.85 ± 0.02
		UltraEdit	0.93 ± 0.01	0.82 ± 0.03
		RLEdit	0.95 ± 0.01	0.86 ± 0.02
GPT-J	ZSRE	StableEdit	0.95 ± 0.00	0.96 ± 0.00
		UltraEdit	0.95 ± 0.00	0.96 ± 0.00
		RLEdit	0.94 ± 0.00	0.90 ± 0.01
	WikiBigEdit	StableEdit	0.95 ± 0.00	0.95 ± 0.00
		UltraEdit	0.95 ± 0.00	0.95 ± 0.00
		RLEdit	0.94 ± 0.00	0.85 ± 0.01
GPT2-XL	ZSRE	StableEdit	0.94 ± 0.02	0.93 ± 0.00
		UltraEdit	0.94 ± 0.02	0.92 ± 0.00
		RLEdit	0.94 ± 0.02	0.92 ± 0.00
	WikiBigEdit	StableEdit	0.95 ± 0.01	0.90 ± 0.01
		UltraEdit	0.95 ± 0.01	0.90 ± 0.01
		RLEdit	0.95 ± 0.01	0.88 ± 0.01

B Additional Analysis of Learned Transformations

B.1 Additional Update Alignment Results

Figure 4 reports the update-level alignment analysis on WikiBigEdit. Consistent with the ZSRE results in the main paper, the transformed and untransformed updates remain highly aligned, and removing the component orthogonal to the untransformed direction causes little change in editing performance.

We additionally compare the transformed factors themselves. For each edit, we measure the cosine similarity between the original and transformed input activations $(z_i, \tilde{z}_i)$ and between the original and transformed output gradients $(z'_o, \tilde{z}'_o)$. As shown in Table 7, the input activations are consistently highly aligned, with cosine similarities ranging from 0.93 to 0.95 across all editors, models, and datasets. The output gradients are somewhat more variable, but still remain strongly aligned overall, ranging from 0.78 to 0.96 with an average cosine similarity of approximately 0.89. This factor-level analysis is consistent with the update-level results and suggests that the learned transformations largely preserve the dominant direction of the raw edit signal.

B.2 Editing Location for Transformed Editors

The main paper analyzes the effect of editing location using the untransformed update. Here, we repeat the same layer-wise analysis for RLEdit, UltraEdit, and StableEdit. As shown in Figures 5, 6, and 7, UltraEdit exhibits a trend largely consistent with the untransformed results, whereas RLEdit and StableEdit show more pronounced fluctuations, possibly due to their training and whitening procedures. Nevertheless, very early layers generally perform worse, while the best performance is typically achieved at intermediate or late layers (Valeriani et al., 2023; Skean et al., 2025).

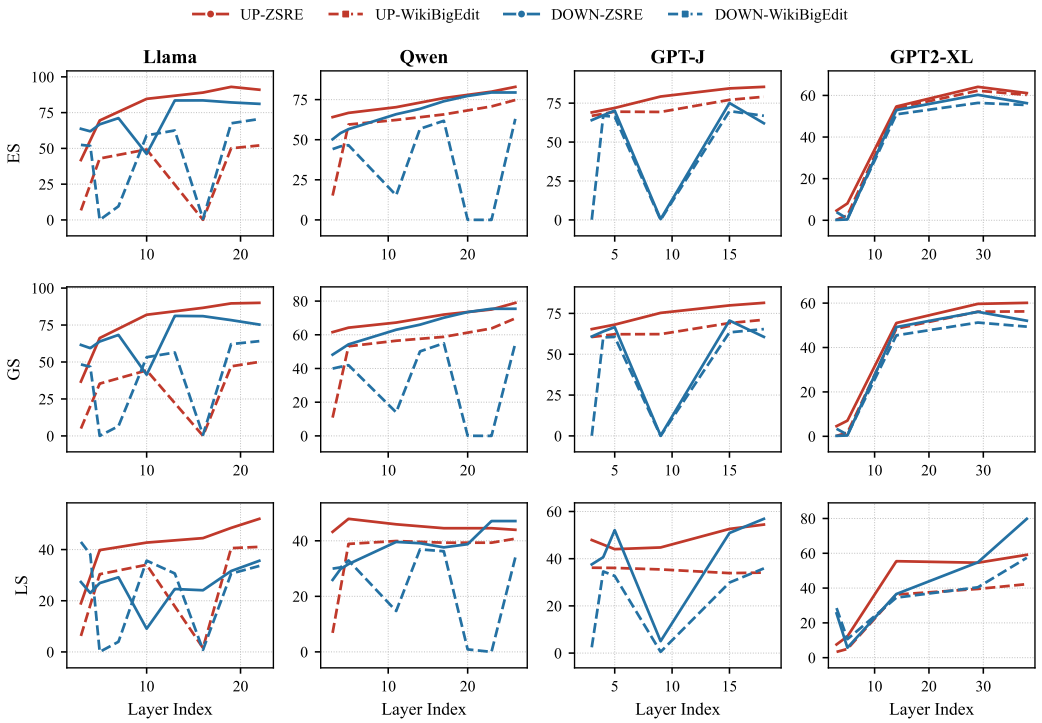


Figure 5: Effect of editing location on RLEdit across four language models and two datasets. We vary the edited layer from early to late positions for MLP down-projection and MLP up-projection modules, and report efficacy, generality, and locality.

Table 8: Effect of removing ridge regression on WikiBigEdit for transformed and untransformed editors. We norm-match the outer-product updates to their ridge-based counterparts to control for update magnitude and report efficacy, generality, and locality.

Method	LLaMA3.1-8B			Qwen3-8B			GPT-J-6B			GPT2-XL		
	ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS
RLEdit	42.1	38.6	8.8	38.7	36.1	12.5	55.7	51.4	21.5	48.7	43.6	51.3
UltraEdit	56.8	52.5	32.4	45.3	42.4	46.0	64.2	59.8	19.3	40.4	38.9	82.1
StableEdit	47.4	42.8	17.9	34.8	32.0	83.2	57.4	53.3	45.7	28.1	26.8	93.4
$\eta = -0.02$	58.6	52.8	30.7	46.8	42.3	40.3	54.7	50.2	41.1	28.1	27.9	93.4
$\eta = -0.05$	37.1	33.1	14.8	41.9	40.2	29.6	55.6	52.9	32.4	37.1	34.6	81.0
$\eta = -0.1$	0.0	0.0	0.0	30.8	27.5	5.3	55.8	54.4	21.6	45.9	41.6	61.2
$\eta = -0.2$	0.0	0.0	0.0	0.0	0.0	0.0	45.8	40.7	14.7	51.0	47.2	48.5

B.3 Removing Ridge Regression on WikiBigEdit

Table 8 complements the ZSRE results in the main paper with WikiBigEdit. Removing the ridge-based construction causes substantial performance degradation, particularly in locality, for both transformed and untransformed updates. This confirms the importance of ridge regularization.

B.4 Effect of Hypernetworks under MEND-Style Updates

The main experiments focus on sequential meta model editing. To examine how the benefit of the learned transformation depends on the editing setting, we further conduct single-edit experiments based on MEND (Mitchell et al., 2022a). Each example is edited and evaluated independently: after evaluating one edit, the model is restored to its original parameters before proceeding to the next

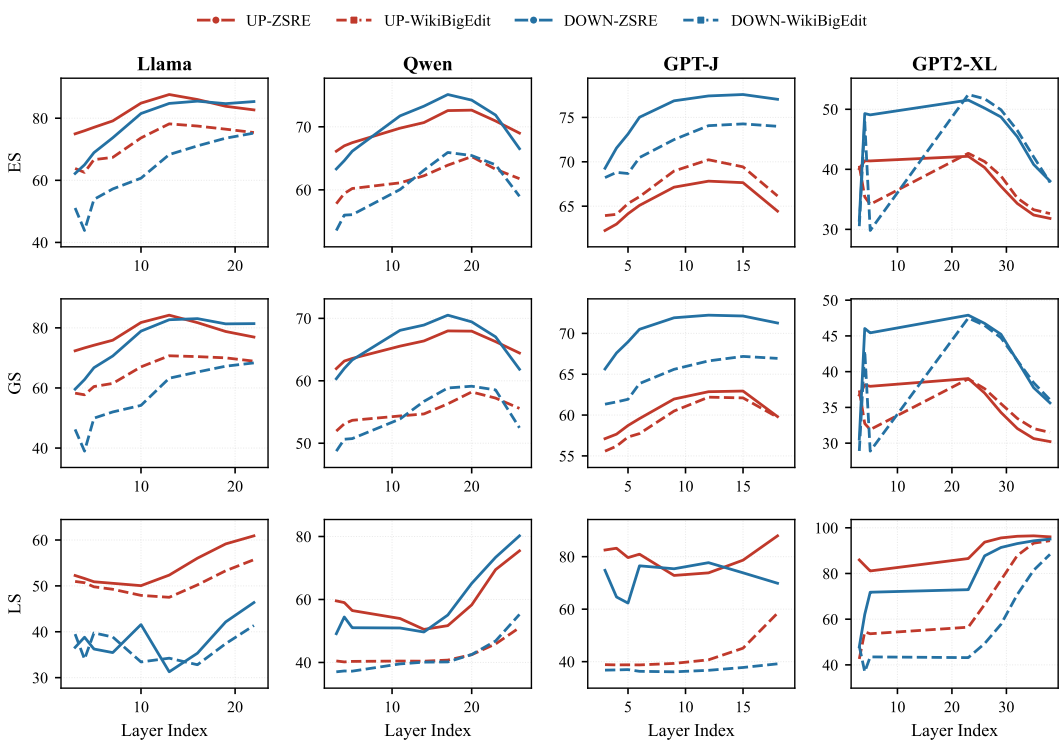


Figure 6: Effect of editing location on UltraEdit across four language models and two datasets. We vary the edited layer from early to late positions for MLP down-projection and MLP up-projection modules, and report efficacy, generality, and locality.

Table 9: Single editing performance under different update formulations. RawEdit[1] removes the transformation and uses a MEND-style update, whereas RawEdit[2] removes the transformation and uses a ridge-regression-based update.

Dataset	Method	Llama			Qwen			GPT-J			GPT2-XL		
		ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS
ZSRE	MEND	98.0	98.1	92.9	96.9	96.1	93.7	99.7	99.5	94.9	99.7	99.7	94.3
	RawEdit[1]	92.2	89.3	86.6	87.2	84.6	82.9	89.7	83.6	96.4	94.7	93.2	84.4
	RawEdit[2]	94.6	93.3	90.7	93.9	93.2	90.5	98.3	98.4	93.4	97.9	96.6	90.9
WikiBigEdit	MEND	94.5	94.5	76.7	92.6	90.7	74.4	92.6	93.5	78.0	84.4	86.9	82.5
	RawEdit[1]	85.6	85.2	75.8	86.3	81.3	75.4	90.8	88.4	77.8	84.8	81.1	81.9
	RawEdit[2]	92.0	91.1	76.9	90.3	87.3	75.3	93.9	92.9	77.5	84.7	84.1	82.8

example (Meng et al., 2022; Gu et al., 2024). We compare MEND with two untransformed variants that differ in their update construction.

RawEdit[1] removes the learned transformation and retains the MEND-style outer-product update, with only the update coefficient optimized. **RawEdit**[2] likewise removes the transformation but replaces the outer-product update with the ridge-regression-based construction and optimizes both the update coefficient and ridge parameter.

Table 9 shows that, under the original MEND-style update, the learned transformation provides a clear advantage over its untransformed counterpart. However, this gap becomes substantially smaller when the untransformed signal is combined with the ridge-regression-based update in RawEdit[2]. To further understand this effect, we compare the geometry of the resulting updates in Table 10. The input and output-gradient factors before and after the MEND transformation exhibit relatively low cosine similarity, indicating that the hypernetwork substantially changes the raw edit signal. Correspondingly, the update produced by MEND is only weakly aligned with that of RawEdit[1].

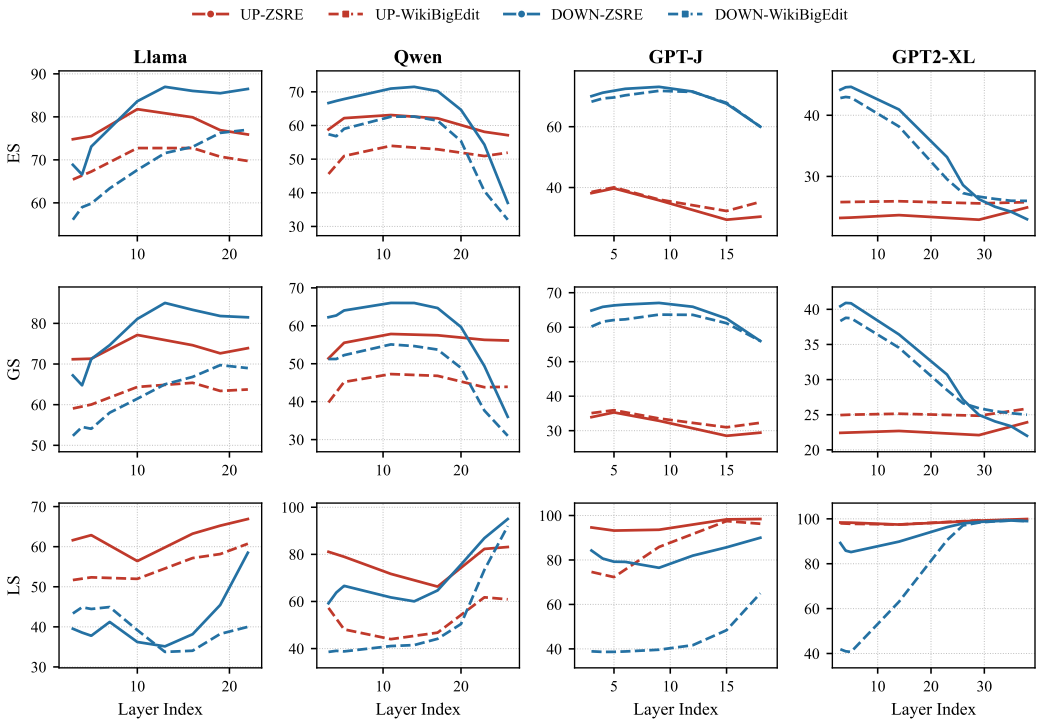


Figure 7: Effect of editing location on StableEdit across four language models and two datasets. We vary the edited layer from early to late positions for MLP down-projection and MLP up-projection modules, and report efficacy, generality, and locality.

In contrast, replacing the outer-product construction with the ridge-based formulation in RawEdit[2] substantially increases its alignment with the MEND update across all models and datasets.

These results suggest that part of the effect introduced by the learned transformation can be recovered through a stronger update formulation, even without transforming the raw edit signal itself. Although MEND still retains an advantage in single-edit performance, this advantage is considerably reduced once ridge regression is introduced. This further supports our conclusion that the incremental utility of a hypernetwork depends strongly on the underlying update rule, rather than being an intrinsic property of the transformation alone.

C RawEdit Derivation and Training Details

C.1 Derivation of the Desired Residual

We provide a detailed comparison between the desired residual used by the conventional formulation and that used by RawEdit.

Recall that, for a linear weight matrix W, the first-step edit gradient can be factorized as

$$\nabla_W \mathcal{L}_{\text{edit}} = z'_o z_i^\top, \tag{24}$$

where z_i denotes the input activation and $z'_o = \nabla_{z_o} \mathcal{L}_{\text{edit}}$ the corresponding output gradient. This outer-product structure is an exact property of the gradient and does not itself impose an assumption on the final parameter update.

The conventional desired residual is obtained by additionally assuming that the parameter update follows the gradient form:

$$\Delta W_{\text{grad}} = \eta z'_o z_i^\top. \tag{25}$$

Table 10: Cosine similarity between MEND and its untransformed variants across models and datasets. "Input" and "Output-gradient" measure the similarity between the original and MEND-transformed edit factors. M–R^1 and M–R^2 denote the cosine similarity between the weight update produced by MEND and those produced by RawEdit^1 and RawEdit^2, respectively. Values are reported as mean $\pm$ standard deviation.

Model	Dataset	Input cosine	Output-gradient cosine	M-R^1 cosine	M-R^2 cosine
Llama3.1-8B	ZSRE	0.25 ± 0.05	0.29 ± 0.12	0.03 ± 0.02	0.41 ± 0.19
	WikiBigEdit	0.24 ± 0.05	0.22 ± 0.11	0.02 ± 0.02	0.40 ± 0.12
Qwen3-8B	ZSRE	0.31 ± 0.06	0.28 ± 0.15	0.05 ± 0.03	0.48 ± 0.14
	WikiBigEdit	0.24 ± 0.05	0.24 ± 0.12	0.02 ± 0.02	0.39 ± 0.09
GPT2-XL	ZSRE	0.48 ± 0.07	0.44 ± 0.14	0.14 ± 0.05	0.58 ± 0.16
	WikiBigEdit	0.39 ± 0.09	0.36 ± 0.14	0.07 ± 0.04	0.51 ± 0.14
GPT-J	ZSRE	0.35 ± 0.05	0.26 ± 0.09	0.06 ± 0.03	0.44 ± 0.14
	WikiBigEdit	0.40 ± 0.06	0.16 ± 0.07	0.03 ± 0.02	0.35 ± 0.12

Applying this assumed update to the edit key z_i gives

$$\Delta W_{\text{grad}} z_i = \eta (z_i^\top z_i) z_o'. \tag{26}$$

Thus, once the gradient-form parameter update is assumed, the desired residual is necessarily scaled by the sample-dependent factor $z_i^\top z_i$. This is the origin of $\eta(z_i^\top z_i)z_o'$ in the conventional formulation.

Importantly, this factor is not required by the subsequent ridge-regression problem itself; it arises from first specifying a particular parameter-update form and then deriving the residual induced by that update. RawEdit removes this intermediate step and directly defines

$$r_{\text{RawEdit}} = \eta z_o' \tag{27}$$

as the desired residual. The regularized optimization in Eq. 10 then determines the corresponding parameter update.

C.2 RawEdit Training Details

RawEdit learns two scalar parameters for each edited layer: an update coefficient η_ℓ and a ridge coefficient β_ℓ. We parameterize them as

$$\eta_\ell = -\exp(s_\ell), \qquad \beta_\ell = \exp(\mu_\ell), \tag{28}$$

which ensures $\eta_\ell < 0$ and $\beta_\ell > 0$.

Training follows the sequential reinforcement strategy of RLEdit (Li et al., 2025b). Rather than optimizing each batch of edits independently (Tan et al., 2024), the training objective at step t evaluates not only the current batch of edits but also a window of recent batches. In this way, the parameters learned for the current update are influenced by whether previously edited knowledge remains effective after subsequent edits.

Let $\mathcal{E}_t$ denote the paraphrased edits associated with the t-th edit batch. We retain the most recent B edit groups and optimize

$$\mathcal{L}_t = \sum_{b=0}^{\min(B-1,t-1)} \gamma^b \frac{1}{|\mathcal{E}_{t-b}|} \sum_{(x,y)\in\mathcal{E}_{t-b}} \text{CE}(f_{\theta+\Delta\theta_t}(x), y) + \lambda_{\text{loc}} \sum_\ell \|\Delta W_{t,\ell}\|_F^2, \tag{29}$$

where B is the backward depth and γ controls temporal decay. Following Li et al. (2026), we use an update-norm regularizer instead of the KL-divergence-based locality objective, which avoids evaluating additional unrelated examples and substantially reduces training cost.

We use $B = 10$ and $\gamma = 0.95$ (Li et al., 2025b). The locality coefficient is set to $\lambda_{\text{loc}} = 0.025$ for LLaMA3.1-8B and Qwen3-8B, and $\lambda_{\text{loc}} = 0.01$ for GPT-J-6B and GPT2-XL. For ZSRE, the training dataset contains 20,000 edits, which are processed sequentially in 200 batches of 100 edits;

Algorithm 1 Training Procedure of RawEdit

Require: Language model f_θ, sequential edit batches $\{\mathcal{T}_t\}_{t=1}^{T}$, backward depth B, temporal decay γ, and locality coefficient λ_{loc}

Initialize s_ℓ and μ_ℓ for each edited layer ℓ
Set $H_{0,\ell} \leftarrow 0$ for each edited layer
Initialize Adam over $\{s_\ell, \mu_\ell\}_\ell$
for $t = 1, \ldots, T$ **do**
    Compute the edit loss $\mathcal{L}_{\text{edit}}^{(t)}$ (Eq. 1) on the current edit batch $\mathcal{T}_t$
    Backpropagate $\mathcal{L}_{\text{edit}}^{(t)}$ to collect, for each edited layer ℓ, the layer inputs $K_{t,\ell}$ and output gradients $G_{t,\ell}$
    for each edited layer ℓ **do**
        $\eta_\ell \leftarrow -\exp(s_\ell)$, $\beta_\ell \leftarrow \exp(\mu_\ell)$
        $D_{t,\ell} \leftarrow \eta_\ell G_{t,\ell}$
        $A_{t,\ell} \leftarrow K_{t,\ell}^\top K_{t,\ell} + H_{t-1,\ell} + \beta_\ell I$
        Compute the RawEdit update $\Delta W_{t,\ell}$ from the ridge solve (Eq. 11)
    end for
    Apply $\{\Delta W_{t,\ell}\}_\ell$ to the model for meta-objective evaluation
    for each edited layer ℓ **do**
        Commit $W_\ell \leftarrow W_\ell + \Delta W_{t,\ell}$
        Update $H_{t,\ell} \leftarrow H_{t-1,\ell} + K_{t,\ell}^\top K_{t,\ell}$
    end for
    Compute $\mathcal{L}_t$ (Eq. 29) on the current and previous $B-1$ groups of equivalent expressions
    Backpropagate $\mathcal{L}_t$ through $\{\Delta W_{t,\ell}\}_\ell$ to obtain gradients for $\{s_\ell, \mu_\ell\}_\ell$ (Tan et al., 2024)
    Update $\{s_\ell, \mu_\ell\}_\ell$ with one Adam step
end for

for WikiBigEdit, we use 17,000 training edits organized into 170 batches of 100 edits, following Ma et al. (2026). Training and evaluation use disjoint training and test splits, respectively; no test edit is observed during training. Training proceeds continuously along this sequential edit stream, and the entire training set is traversed once (Li et al., 2025b). Adam updates $\{s_\ell, \mu_\ell\}$ for all edited layers once after every sequential edit batch.

Algorithm 1 summarizes the training procedure. For each sequential edit batch, RawEdit first extracts the layer inputs and output gradients, constructs the current update using the accumulated historical Gram matrix, and evaluates the current and recent batches of paraphrased edits. The resulting gradients are then propagated to the layer-wise scalar parameters before proceeding to the next edit batch.

We examine sensitivity to λ_{loc} by sweeping $\{0.01, 0.015, 0.02, 0.025, 0.03\}$ across all four models and both datasets. As shown in Figure 8, RawEdit is robust across this range. Increasing λ_{loc} generally improves locality while gradually reducing efficacy and generality, reflecting the expected trade-off between editing strength and preservation.

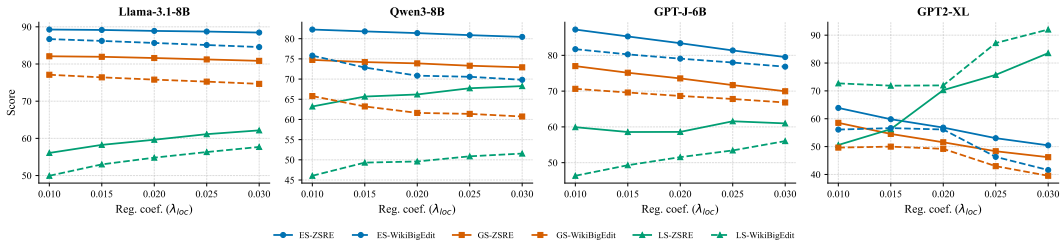


Figure 8: Sensitivity of RawEdit to the locality regularization coefficient λ_{loc} across four models and two datasets. We sweep λ_{loc} from 0.01 to 0.03 and observe stable performance over a broad range, with larger values generally improving locality while slightly reducing efficacy and generality.

Table 11: Editor-side overhead and computational efficiency of meta-learning-based editors. Editor size denotes the method-specific learned parameters or maintained statistics. Edit time is averaged across the four models, and all runtime measurements are obtained on a single NVIDIA A100 GPU.

Method	Editor Size				Edit Time (s)	Training / Warm-up Time
	GPT2-XL	LLaMA3.1-8B	Qwen3-8B	GPT-J-6B		
RLEdit	66.032M	152.138M	135.234M	169.042M	0.08	1.1 h
UltraEdit	0.016M	0.037M	0.033M	0.041M	0.07	2.3 min
StableEdit	2.576M	16.814M	16.810M	16.818M	0.11	3.6 min
RawEdit	14 scalars	14 scalars	14 scalars	14 scalars	0.07	0.7 h

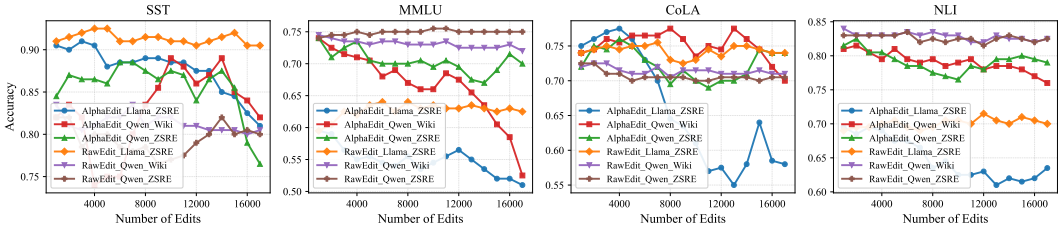


Figure 9: Additional general capability results during 17,000 sequential edits. RawEdit and AlphaEdit are evaluated every 1,000 edits on SST, MMLU, CoLA, and NLI using LLaMA3.1-8B and Qwen3-8B on ZSRE and WikiBigEdit.

D Additional Experimental Results

D.1 Efficiency Comparison with Meta-Learning Editors

We compare RawEdit with existing meta-learning-based editors in terms of method-specific editor size, editing time, and training or warm-up time. All runtime measurements are obtained on a single NVIDIA A100 GPU. The reported editor sizes are not directly equivalent across methods: RLEdit contains a learned hypernetwork, UltraEdit and StableEdit maintain normalization- or whitening-related statistics, whereas RawEdit learns only two scalar parameters per edited layer. We report these quantities to characterize the method-specific editor-side overhead of each approach. As shown in Table 11, RawEdit requires only 14 learned scalars for the seven edited layers and achieves an editing speed comparable to UltraEdit while being faster than RLEdit and StableEdit. RawEdit training takes approximately 0.7 hours in our setup, compared with 1.1 hours for RLEdit.

UltraEdit and StableEdit require only a brief warm-up because their transformations rely on running normalization/whitening statistics rather than a separately trained hypernetwork. However, this low setup cost does not necessarily translate into robust editing across models. As shown in Table 1, both methods exhibit substantial performance degradation on GPT2-XL. For example, StableEdit achieves very high locality (99.4 LS on both datasets), but its efficacy is only 23.7 on ZSRE and 25.8 on WikiBigEdit. UltraEdit shows a similar pattern, with high locality but substantially lower efficacy and generality. In these cases, the high locality reflects a degenerate editing regime, where the model remains close to its original behavior because the intended edits are only weakly established.

D.2 Strengthening Existing Editors with RawEdit's Update Formulation

RawEdit strengthens the conventional update formulation through two main changes: a direct residual and history-aware regularization. We further examine whether these improvements can benefit existing meta-learning-based editors.

For RLEdit, UltraEdit, and StableEdit, the gradient-induced residual cannot be removed independently of their transformation mechanism as their transformed weight inputs directly participate in constructing the residual; replacing this term with RawEdit's direct residual would therefore simultaneously remove part of the learned transformation, preventing a controlled comparison. We

Table 12: Effect of adding RawEdit's history-aware regularization (HAR) to existing meta-learning-based editors.

Method	Dataset	LLaMA3.1-8B			Qwen3-8B			GPT-J-6B			GPT2-XL		
		ES	GS	LS	ES	GS	LS	ES	GS	LS	ES	GS	LS
RLEdit	ZSRE (20k)	84.4	80.5	32.3	77.8	73.1	54.3	72.1	67.4	56.3	56.0	52.1	79.9
w/ HAR		86.7	81.6	53.0	80.4	73.3	59.0	73.9	66.5	64.4	58.0	52.5	81.6
UltraEdit		81.3	75.5	62.7	66.6	62.0	79.5	77.0	71.2	70.1	37.7	35.4	95.2
w/ HAR		84.8	74.9	67.0	68.3	61.6	81.0	78.9	72.9	72.4	37.3	35.0	96.1
StableEdit		86.1	81.2	51.8	36.9	35.6	95.4	59.5	55.5	90.2	23.7	22.9	99.4
w/ HAR		88.0	81.8	59.1	36.0	34.7	96.3	61.6	56.3	92.9	23.9	22.5	99.6
RLEdit	WikiBigEdit (17k)	70.0	64.2	33.6	63.7	57.6	34.5	66.2	60.9	35.4	55.7	49.8	57.5
w/ HAR		75.4	66.5	40.5	67.6	57.4	42.4	69.8	60.1	38.5	56.8	50.1	61.1
UltraEdit		76.1	69.2	40.4	58.4	51.7	56.0	74.1	67.0	39.9	37.2	35.6	89.9
w/ HAR		77.8	69.9	43.4	60.0	51.3	58.1	74.4	68.2	43.1	37.3	36.0	91.4
StableEdit		77.2	69.7	40.9	32.4	31.2	92.9	60.5	55.3	65.4	25.8	25.0	99.4
w/ HAR		79.0	70.6	42.3	35.8	32.7	93.4	63.0	56.9	69.0	25.5	24.9	99.6

Table 13: Comparison with setting-specific editors for same-subject and long-form editing.

Method	mES	mGS	mLS	lES	lGS	lLS	mES	mGS	mLS	lES	lGS	lLS
	LLaMA3.1-8B						Qwen3-8B					
MEMIT-Merge	78.9	64.9	86.0	82.0	78.9	100	76.7	55.1	86.0	65.2	64.7	100
AnyEdit	68.9	57.8	91.9	91.0	89.3	94.3	45.8	37.9	99.0	91.7	90.2	92.4
	GPT-J						GPT2-XL					
MEMIT-Merge	81.8	64.8	95.3	82.7	67.1	100	74.5	65.1	92.9	51.6	50.1	100
AnyEdit	61.9	45.8	97.1	91.8	90.1	96.6	62.1	53.9	96.1	85.0	75.1	89.6

consequently retain their original transformations and residual definitions, and incorporate only the separable history-aware Gram term into their update construction.

As shown in Table 12, this modification generally improves editing performance across models and datasets, with particularly consistent gains in locality. These results further support our conclusion that strengthening the update formulation can benefit editing performance even when the original transformation mechanism is retained.

D.3 Additional General Capability Results

Beyond the locality metric, we evaluate whether accumulated edits affect broader model capabilities (Fang et al., 2025a; Li et al., 2025b). We consider four benchmarks covering different abilities: SST for sentiment classification, MMLU for broad knowledge and reasoning, CoLA for linguistic acceptability, and NLI for natural language inference. Models are evaluated every 1,000 sequential edits. We omit GPT-J and GPT2-XL from this analysis because their pre-edit performance on these benchmarks is already very poor, making post-edit degradation less informative.

Figure 9 extends the main-paper analysis to LLaMA3.1-8B and Qwen3-8B on both ZSRE and WikiBigEdit. For consistent comparison, ZSRE is truncated to the first 17,000 edits. Across most model–dataset–benchmark combinations, RawEdit preserves general capabilities more consistently than AlphaEdit as editing proceeds, in line with its strong locality performance.

D.4 Additional Results on Challenging Editing Scenarios

The main paper reports challenging-setting results on LLaMA3.1-8B and Qwen3-8B. Table 14 complements these results with GPT-J and GPT2-XL. Across broader semantic generalization, same-

Table 14: Performance on challenging editing scenarios using GPT-J and GPT2-XL. PA and MH denote persona-style and multi-hop generalization; mES/mGS/mLS and lES/lGS/lLS denote ES/GS/LS in the same-subject and long-form settings, respectively.

Method	GPT-J								GPT2-XL							
	PA	MH	mES	mGS	mLS	lES	lGS	lLS	PA	MH	mES	mGS	mLS	lES	lGS	lLS
RegFT	29.3	20.5	96.6	84.4	90.5	99.7	98.9	90.8	29.7	21.0	92.8	62.5	59.7	93.5	89.8	75.8
AlphaEdit	41.3	17.2	61.6	43.0	98.9	81.0	70.5	100	8.8	3.6	59.7	51.7	98.9	50.0	50.5	100
BetaEdit	40.3	17.0	59.7	48.2	99.4	80.5	68.5	100	19.6	15.9	62.6	42.5	98.1	49.0	50.0	100
RLEdit	55.4	26.1	92.4	81.7	93.6	98.7	98.9	92.5	37.6	24.6	82.1	64.2	70.7	90.5	83.5	78.1
UltraEdit	57.0	30.1	91.6	83.5	93.2	97.9	97.4	93.1	28.7	21.6	81.8	65.9	73.1	88.9	83.1	83.8
RawEdit	56.1	29.4	91.7	82.2	94.1	98.1	97.4	94.0	39.8	23.5	84.7	67.8	70.3	90.5	83.2	81.3

subject editing, and long-form editing, RawEdit continues to achieve the best performance among the baselines, demonstrating its robustness across these diverse evaluation settings.

D.5 Comparison with Setting-Specific Editors

We further compare RawEdit with methods developed specifically for the two challenging settings. MEMIT-Merge (Dong et al., 2025) is designed for same-subject editing, while AnyEdit (Jiang et al., 2025) targets long-form editing. As shown in Table 13, RawEdit remains competitive with, and often outperforms, these setting-specific methods. Both MEMIT-Merge and AnyEdit construct edits around selected token representations, such as those associated with the subject, whereas RawEdit uses edit signals from all answer-token positions (Tan et al., 2024). This avoids relying on a single localized token and makes RawEdit naturally applicable to both same-subject and long-form edits. Fine-tuning similarly shows effectiveness as it optimizes over all token positions.