MetaKE: Meta-Learning for Knowledge Editing Toward a Better Accuracy-Editability Trade-off

Shuxin Liu **Di Gao** **Ou Wu***
liushuxin25@mails.ucas.ac.cn {gaodi,wuou}@ucas.ac.cn
Hangzhou Institute for Advanced Study
University of Chinese Academy of Sciences
Hangzhou, China

Abstract

Existing locate-then-edit Knowledge Editing (KE) methods typically decompose editing into two stages: upstream target representation optimization and downstream constrained parameter optimization. The optimization across the two stages is disconnected: upstream applies uniform regularization without observing downstream realization of the planned residual, hindering a refined accuracy–editability trade-off. Since this realization is request-specific and depends on downstream constraints, uniform regularization can over-shrink high-association requests, causing insufficient editing, while it can under-regularize low-association requests, producing over-large planned residuals that reduce downstream editability. To bridge this disconnect, we propose **MetaKE** (**Meta**-learning for **K**nowledge **E**diting), a new framework that unifies upstream and downstream stages into a bi-level optimization problem. The inner level optimizes parameter updates for the target representation, while the outer level optimizes representation using feedback from downstream constraints, achieving a better semantic accuracy-editability trade-off. To avoid costly multi-layer backpropagation, we introduce a Structural Gradient Proxy to approximate and propagate this feedback. Extensive experiments show that MetaKE outperforms strong baselines, offering a new perspective on KE. Code at: https://github.com/zhutizi23-coder/MetaKE.

1 Introduction

Large Language Models (LLMs) have become a core substrate for NLP systems [1–3], yet their parametric memory inevitably contains factual errors and outdated knowledge [4–10]. KE aims to update specific knowledge in LLMs while preserving general capabilities [11, 12]. Recent studies suggest that the *editing* process in locate-then-edit methods (e.g., MEMIT [13], AlphaEdit [14]) consists of two optimization stages [15–19]: an upstream stage that optimizes a target representation and induces a planned residual, and a downstream stage that realizes this residual through constrained parameter optimization. In the upstream stage, methods such as PMET [20] and STEAM [21] apply uniform KL regularization across different target knowledge to improve semantic accuracy and reduce interference with preserved knowledge. In the downstream stage, MEMIT uses covariance constraints, while AlphaEdit introduces projection constraints to protect preserved knowledge.

Despite these advances, the optimization across the two stages remains disconnected, which hinders a refined accuracy–editability trade-off. The upstream controls the planned residual through the target representation, whereas the downstream stage realizes only a request-specific fraction of it, governed by the request's association with the downstream constraints, i.e., how strongly the request interacts with those constraints. The shared upstream regularizer, however, is association-blind: it scales

*Corresponding author.

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

the planned residual without observing its request-specific realization. Since the realized update depends jointly on this residual and the request's association with the downstream constraints, the same regularization strength may fail to suit different requests: high-association requests may be over-shrunk, causing insufficient editing, while low-association requests may yield over-large planned residuals that are difficult to realize under downstream constraints, thereby reducing downstream editability. A request-specific regularization strength can rescale the planned residual, but such a scalar adjustment changes the residual magnitude without revealing to the upstream stage how much of that residual will be realized after downstream optimization. Intuitively, if the upstream stage could receive feedback on downstream constraints while optimizing the target representation, it could achieve a better semantic accuracy–editability trade-off.

To bridge this disconnect, we propose **MetaKE** (**Meta**-learning for **K**nowledge **E**diting), a new framework that unifies upstream and downstream stages into a bi-level optimization problem. The inner level optimizes the parameter update for a given target representation, while the outer level optimizes the target representation using feedback from the downstream constraints; the target representation sits at the outer level because it specifies what the parameter update should achieve, while the update is its constrained realization. To avoid expensive multi-layer backpropagation, we introduce a *Structural Gradient Proxy*, which efficiently approximates and propagates feedback from downstream constraints without full differentiation through the multi-layer process. Based on this mechanism, MetaKE first refines the target representation in the bi-level phase using downstream-aware feedback, and then performs the multi-layer parameter update based on the refined target.

Our contributions are summarized as follows:

- We analyze the disconnect between the two optimization stages and show that, under a local quadratic surrogate, downstream realization is request-specific while the shared upstream regularizer is association-blind, thereby hindering a refined accuracy–editability trade-off.
- We propose **MetaKE**, a new framework that unifies upstream and downstream stages into a bi-level optimization problem, enabling the upstream to perceive the downstream constraints and achieve a better semantic accuracy–editability trade-off.
- We introduce a *Structural Gradient Proxy* that provides a tractable approximation of downstream constraints feedback, avoiding costly full differentiation through multi-layer updates.
- Experiments on GPT-2 XL (1.5B), GPT-J (6B), and LLaMA3 (8B) show that MetaKE achieves SOTA performance and consistently improves the accuracy-editability trade-off.

2 Related Work

Knowledge Editing with Constraints. The locate-then-edit paradigm is central for KE, where factual associations are localized and modified through targeted updates [13, 22, 23]. Knowledge Neurons attributes factual knowledge to feed-forward neurons [23], while ROME edits factual recall via rank-one updates [22]. MEMIT extends this to multi-layer editing with covariance-based preservation constraints [13]. EMMET unifies ROME and MEMIT under a preservation–memorization objective and derives a closed-form constrained update [15]. AlphaEdit projects perturbations onto the null space of preserved knowledge to reduce disruption to preserved key-value associations [14], while LyapLock studies preservation under sequential editing [24]. These methods mainly strengthen downstream updates, constraints, or edit trajectories after target optimization. In contrast, MetaKE studies how downstream constraints provide feedback to upstream target optimization.

Meta-Learning and Target-Centric Optimization for Knowledge Editing. Another line of work improves KE by learning update mappings or optimizing targets. KE and MEND train hypernetworks to transform gradients into parameter updates [11, 25], while MALMEN scales this paradigm to batched edits via least-squares approximations [26]. InstructEdit improves editor generalization with task-specific instructions, and RLEdit formulates lifelong hypernetwork editing as reinforcement learning [27, 28]. Target-centric methods such as AdaEdit, AnyEdit, REP, and DAFNet refine editing targets, robust keys, or sequence-level representations [29–32]. These methods show the importance of learned update mappings and target optimization, but their signals are mainly semantic, gradient-based, instructional, or robustness-oriented. MetaKE instead couples upstream target refinement with downstream constrained realization, refining targets using downstream feedback.

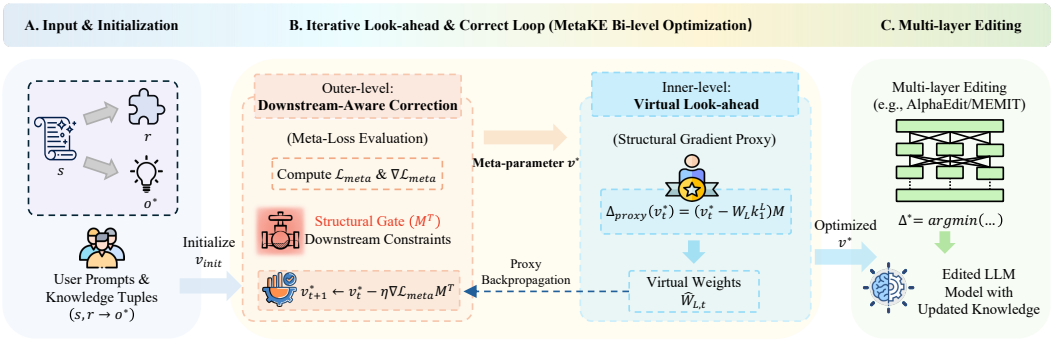

Figure 1: The architecture of the proposed method MetaKE.

3 Preliminaries

3.1 Knowledge Editing as Linear Associative Memory

KE typically treats Transformer FFNs as linear associative memories [9, 33, 34]. For an edited layer l, the FFN output matrix $\boldsymbol{W}_{\text{out}}^{l} \in \mathbb{R}^{d_1 \times d_0}$ maps a key $\boldsymbol{k}_l \in \mathbb{R}^{d_0}$, which encodes the subject-relation pair (s, r), to a value $\boldsymbol{m}_l \in \mathbb{R}^{d_1}$, which contributes to predicting the object o:

$$\boldsymbol{m}_l \approx \boldsymbol{W}_{\text{out}}^{l} \boldsymbol{k}_l, \qquad \boldsymbol{k}_l = \sigma(\boldsymbol{W}_{\text{in}}^{l} \gamma(\cdot)), \tag{1}$$

where $\boldsymbol{W}_{\text{in}}^{l}$ is the FFN input matrix, $\gamma(\cdot)$ denotes layer normalization, and $\sigma(\cdot)$ is the FFN activation function. In this paper, we use $\boldsymbol{W}_l := \boldsymbol{W}_{\text{out}}^{l}$. Editing seeks a parameter update such that the post-edit memory output aligns with the desired object o^*.

3.2 The Editing Process in Locate-then-Edit as Two Optimization Stages

In this paper, **semantic accuracy** measures how well a target representation $\boldsymbol{v}^*$ encodes the desired knowledge. **Editability** denotes how well $\boldsymbol{v}^*$ can be translated into parameter updates under downstream constraints. Recent analyses suggest that the editing process in locate-then-edit methods can be viewed as two optimization stages [15–19].

Stage I: Upstream Target Representation Optimization. MEMIT-style variants optimize $\boldsymbol{v}^*$ for the desired knowledge through a likelihood objective with KL regularization [20, 35–37]:

$$\boldsymbol{v}^* = \boldsymbol{m}_L + \arg\min_{\boldsymbol{\delta}} (-\log \mathbb{P}_{f_{L_{\text{out}}}(\boldsymbol{m}_L + \boldsymbol{\delta})}[o^* \mid s, r] + \omega D_{\text{KL}}(\mathbb{P}_f[\cdot \mid p'] \| \mathbb{P}_{f_{L_{\text{out}}}(\boldsymbol{m}_L + \boldsymbol{\delta})}[\cdot \mid p'])), \tag{2}$$

where $\boldsymbol{m}_L$ is the original representation at the last edited layer L, $f_{L_{\text{out}}}(\cdot)$ denotes the remaining forward, and p' is the prompt template. The likelihood term promotes semantic accuracy, while the KL term limits distributional shift. The upstream stage optimizes the semantic target under a uniform regularization strength without accounting for how editable it will be under downstream constraints.

Stage II: Downstream Constrained Parameter Optimization. Given the target representation $\boldsymbol{v}^*$, the downstream stage optimizes a constrained parameter update. Let $\boldsymbol{k}_1$ be the key of the to-be-updated knowledge. The prescribed residual is $\boldsymbol{r} = \boldsymbol{v}^* - \boldsymbol{W}\boldsymbol{k}_1$. For AlphaEdit, the preserved keys $\boldsymbol{K}_0$ are protected by a null-space projection matrix $\boldsymbol{P}$ satisfying $\boldsymbol{P}\boldsymbol{K}_0 = \boldsymbol{0}$, while $\boldsymbol{K}_p$ denotes the keys of previously edited knowledge. The objective can be written as

$$\arg\min_{\widetilde{\boldsymbol{\Delta}}} \left(\|(\boldsymbol{W} + \widetilde{\boldsymbol{\Delta}}\boldsymbol{P})\boldsymbol{k}_1 - \boldsymbol{v}^*\|_2^2 + \|\widetilde{\boldsymbol{\Delta}}\boldsymbol{P}\|_F^2 + \|\widetilde{\boldsymbol{\Delta}}\boldsymbol{P}\boldsymbol{K}_p\|_F^2 \right). \tag{3}$$

The applied update is

$$\boldsymbol{\Delta}_{\text{AlphaEdit}}^{*} = \boldsymbol{r}\boldsymbol{k}_1^T\boldsymbol{P} \left(\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P} + \boldsymbol{k}_1\boldsymbol{k}_1^T\boldsymbol{P} + \boldsymbol{I} \right)^{-1}. \tag{4}$$

4 Methodology

We first analyze the disconnect between the two optimization stages, and then describe our proposed method MetaKE. Fig. 1 illustrates the overall architecture.

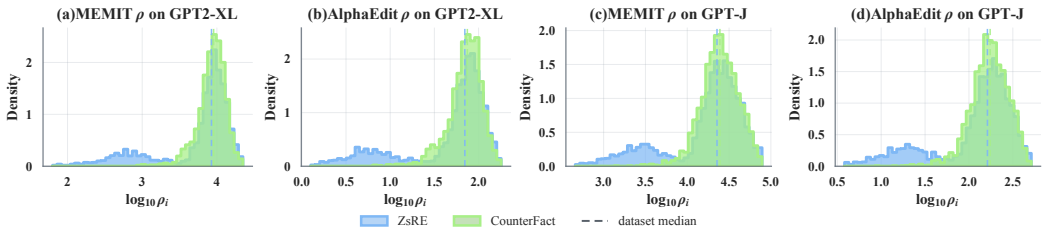

Figure 2: Distribution of request-specific downstream association ρ_i over 500 sampled edit requests. Dashed lines mark medians.

4.1 The Disconnect Between the Two Optimization Stages

Stage I optimizes $\boldsymbol{v}^*$ under a uniform regularization strength (e.g., the KL weight ω), while Stage II realizes this target through a constrained update induced by the downstream constraints. For the current edit, the key $\boldsymbol{k}_1$ is determined by the input subject–relation pair through the forward pass and is not an upstream variable; the quantity Stage I effectively controls is the planned residual $\boldsymbol{r} = \boldsymbol{v}^* - \boldsymbol{W}\boldsymbol{k}_1$. However, what Stage II writes into the model is not $\boldsymbol{r}$ itself.

Proposition 1 (Realized-Residual Attenuation)**.** *Consider a current edit with key $\boldsymbol{k}_1$ and planned residual $\boldsymbol{r}$. Under the single-request local quadratic surrogate of Stage II with fixed keys and constraints, the realized residual satisfies $\boldsymbol{\delta}^{\text{real}} = \beta\boldsymbol{r}$ with $\beta = \rho/(1+\rho) \in [0,1)$. We refer to ρ as the request-specific* association*, which quantifies how strongly $\boldsymbol{k}_1$ interacts with the downstream constraints. For MEMIT covariance constraints, $\rho = \boldsymbol{k}_1^T(\boldsymbol{C}_0 + \boldsymbol{K}_p\boldsymbol{K}_p^T)^{-1}\boldsymbol{k}_1$, where $\boldsymbol{C}_0 = \boldsymbol{K}_0\boldsymbol{K}_0^T$; for AlphaEdit projected updates, $\rho = (\boldsymbol{P}\boldsymbol{k}_1)^T(\boldsymbol{I} + \boldsymbol{P}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P})^{-1}\boldsymbol{P}\boldsymbol{k}_1$. For batch editing, the scalar attenuation is replaced by a request-interaction matrix. Proof in Appendix A.1.*

Proposition 1 reveals that Stage II writes $\beta\boldsymbol{r}$, not $\boldsymbol{r}$ itself, with β request-specific because ρ depends on the request key and downstream constraints. To assess this request-specificity, we measure ρ_i for each edit request. As shown in Fig. 2, the ratio between the 95th and 5th percentiles of ρ_i reaches 48.9× on ZsRE and remains around 4–5× on CounterFact. Since $\beta_i = \rho_i/(1+\rho_i)$, this dispersion induces request-specific realized and unrealized components, $\beta_i\boldsymbol{r}_i$ and $(1-\beta_i)\boldsymbol{r}_i$. In high-$\rho_i$ regimes, the variation manifests more clearly in $1-\beta_i = 1/(1+\rho_i)$, which governs the editability-side upper bound $\tau/(1-\beta_i)$ in Theorem 1. Hence upstream choice of $\boldsymbol{r}_i$ should account for both realized and unrealized downstream components. The shared KL weight ω, however, controls the planned residual scale: locally, $\|\boldsymbol{r}_i(\omega)\| \approx a_i/\omega$, where a_i depends on the local upstream objective. The effective edit signal is $\beta_i\|\boldsymbol{r}_i(\omega)\|$, whereas the unrealized part is $(1-\beta_i)\|\boldsymbol{r}_i(\omega)\|$. Thus, ω is association-blind: it scales the planned residual without observing this downstream decomposition.

Theorem 1 (Association-Dependent Editing Requirement)**.** *Under Proposition 1, suppose request i requires $\|\boldsymbol{\delta}_i^{\text{real}}\| \geq m$ for semantic success and $\|\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\text{real}}\| \leq \tau$ for downstream editability. Then any feasible planned residual satisfies $\frac{m}{\beta_i} \leq \|\boldsymbol{r}_i\| \leq \frac{\tau}{1-\beta_i}$, and the feasibility interval depends on the downstream association through $\beta_i = \rho_i/(1+\rho_i)$. In particular, if $\beta_i < m/(m+\tau)$, no choice of $\|\boldsymbol{r}_i\|$ meets both requirements within this local model. Proof in Appendix A.2.*

Theorem 1 shows that the admissible scale of $\boldsymbol{r}_i$ is governed by the residual decomposition induced by ρ_i: semantic success imposes the lower bound m/β_i, whereas editability imposes the upper bound $\tau/(1-\beta_i)$. A shared ω, however, imposes a shrinkage rule that does not observe this downstream-dependent decomposition. For high-association requests, $\boldsymbol{r}_i$ is largely realized, so excessive shrinkage may suppress otherwise feasible edits. For lower-association requests, a larger unrealized component can make an over-large planned residual difficult to realize under downstream constraints. These regimes impose different requirements on the residual scale, so a shared ω can be miscalibrated across requests. Appendix A.3 further shows that, in the bi-level formulation detailed in Sec. 4.2, upstream target optimization should account for downstream realization rather than being analyzed as an isolated optimization objective.

4.2 Bi-level Optimization for Downstream-Aware Editing

Sec. 4.1 concludes that a better accuracy–editability trade-off requires $\boldsymbol{v}^*$ to be optimized with access to the downstream constraints, so that $\boldsymbol{r}_i$ can anticipate, per request, how much of itself will survive the downstream constraints. We grant this access by recasting KE as a **Bi-level Optimization (BLO)** problem with $\boldsymbol{v}^*$ as a learnable *meta-parameter*. Let $\mathcal{W} := \{\boldsymbol{W}_l\}_{l\in\mathcal{L}}$ denote the key-layer weights and $\boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*) := \{\boldsymbol{\Delta}_l^*(\boldsymbol{v}^*)\}_{l\in\mathcal{L}}$ the multi-layer downstream updates. The BLO objective is

$$\begin{aligned} \boldsymbol{v}^*_{\text{meta}} = \arg\min_{\boldsymbol{v}^*} \quad & \mathcal{L}_{\text{meta}}\big(f(\cdot;\mathcal{W}+\boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*))\big) \\ \text{s.t.} \quad & \boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*) = \arg\min_{\widetilde{\boldsymbol{\Delta}}\mathcal{W}} \ \mathcal{J}_{\text{inner}}(\widetilde{\boldsymbol{\Delta}}\mathcal{W};\boldsymbol{v}^*). \end{aligned} \tag{5}$$

where $\widehat{\mathcal{W}} = \mathcal{W} + \boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*)$ denotes the virtual post-edit weights. The nesting embeds the downstream stage as a constraint of the upstream optimization, so that $\boldsymbol{v}^*$ is both semantically accurate and editable under the downstream constraints.

Inner Level. Given $\boldsymbol{v}^*$, the inner level reproduces the standard downstream constrained update by allocating a layer-specific target $\boldsymbol{t}_l(\boldsymbol{v}^*)$ detailed in Appendix B.2 and minimizing

$$\mathcal{J}_{\text{inner}} = \sum\nolimits_{l\in\mathcal{L}} \Big[\|(\boldsymbol{W}_l + \widetilde{\boldsymbol{\Delta}}_l\boldsymbol{P}_l)\boldsymbol{k}_1^l - \boldsymbol{t}_l(\boldsymbol{v}^*)\|_2^2 + \|\widetilde{\boldsymbol{\Delta}}_l\boldsymbol{P}_l\|_F^2 + \|\widetilde{\boldsymbol{\Delta}}_l\boldsymbol{P}_l\boldsymbol{K}_p^l\|_F^2 \Big], \tag{6}$$

where $\boldsymbol{P}_l$ is the projection induced by $\boldsymbol{K}_0^l$ and $\boldsymbol{K}_p^l$ collects previously edited keys. The minimizer $\boldsymbol{\Delta}_l^*(\boldsymbol{v}^*)$ is an *implicit function* of $\boldsymbol{v}^*$; the inner level introduces no new objective beyond the existing downstream stage but exposes its realization to the upstream.

Outer level. The outer level refines $\boldsymbol{v}^*$ on the virtual post-edit model:

$$\mathcal{L}_{\text{meta}}(\boldsymbol{v}^*) = \mathcal{L}_{\text{edit}}(x_e, y_e; \widehat{\mathcal{W}}) + \mathcal{L}_{\text{loc}}(x_{\text{loc}}; \widehat{\mathcal{W}}, \mathcal{W}) + \|\boldsymbol{v}^* - \boldsymbol{v}_{\text{init}}\|_2^2, \tag{7}$$

The edit loss encourages the post-edit model to predict the target output, the locality loss penalizes unintended drift on unrelated prompts, and the meta-regularization keeps the refined target close to the initial semantic plan $\boldsymbol{v}_{\text{init}}$. All three terms are used with unit coefficients in our experiments.

4.3 Structural Gradient Proxy

Direct differentiation through the implicit map $\boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*)$ is computationally prohibitive, since each outer step requires either nested optimization or a layer-coupled linear solve. To make Eq. (5) tractable, we introduce a **Structural Gradient Proxy** that approximates the downstream-aware feedback through a single closed-form surrogate. The proxy rests on a local *Structural Consistency Hypothesis*: across the edited layers $\mathcal{L}$, the projection $\boldsymbol{P}_l$ and the previous-edit Gram $\boldsymbol{K}_p^l(\boldsymbol{K}_p^l)^T$ are aligned with their final-layer counterparts up to a bounded drift. Under this hypothesis, the dominant downstream signal is captured by the closed form at the final edited layer L:

$$\boldsymbol{\Delta}_{\text{proxy}}(\boldsymbol{v}^*) = (\boldsymbol{v}^* - \boldsymbol{W}_L\boldsymbol{k}_1^L)\,\boldsymbol{M}, \quad \boldsymbol{M} = (\boldsymbol{k}_1^L)^T\boldsymbol{P}_L\big(\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T\boldsymbol{P}_L + \boldsymbol{k}_1^L(\boldsymbol{k}_1^L)^T\boldsymbol{P}_L + \boldsymbol{I}\big)^{-1}. \tag{8}$$

Here $\boldsymbol{\Delta}_{\text{proxy}}$ is already the update applied to the virtual final layer, and $\boldsymbol{M}$ is request-specific through $\boldsymbol{k}_1^L$. Substituting $\boldsymbol{\Delta}_{\text{proxy}}$ into the chain rule yields a tractable hypergradient:

$$\nabla_{\boldsymbol{v}^*}\mathcal{L}_{\text{meta}} = \frac{\partial\mathcal{L}_{\text{meta}}}{\partial f}\cdot\frac{\partial f}{\partial\boldsymbol{\Delta}_{\text{proxy}}}\cdot\frac{\partial\boldsymbol{\Delta}_{\text{proxy}}}{\partial\boldsymbol{v}^*} + \left.\frac{\partial\mathcal{L}_{\text{meta}}}{\partial\boldsymbol{v}^*}\right|_f = \nabla_{\boldsymbol{\Delta}_{\text{proxy}}}\mathcal{L}_{\text{meta}}\cdot\boldsymbol{M}^T + 2(\boldsymbol{v}^* - \boldsymbol{v}_{\text{init}}). \tag{9}$$

For readability, Eqs. (8)–(9) use single-request notation. In implementation, current-batch keys and target residuals are column-stacked and updated in batched matrix form. The Task Signal denotes the downstream-mediated gradient induced by the semantic-accuracy and locality terms. The Structural Gate filters this signal through $\boldsymbol{P}_L$ and $\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T$, propagating approximate downstream constraints back to $\boldsymbol{v}^*$. We instantiate the gate in the main experiments with the AlphaEdit-style projected update, whose local response admits the closed form in Eq. (8). The same bi-level interface also admits operator-specific gates for native MEMIT and ridge-stabilized EMMET; their derivations and paired experiments are reported in Appendix D.6.

4.4 Optimization Algorithm: MetaKE

MetaKE refines the target $\boldsymbol{v}^*$ via an iterative *Look-ahead and Correct* loop.

Algorithm 1 MetaKE

Require: LR η, iterations T, data $(x_e, y_e, x_{\text{loc}})$, initial target $\boldsymbol{v}_{\text{init}}$
Ensure: Edited weights $\widehat{\boldsymbol{W}}$

Precompute Structural Gate $\boldsymbol{M}$ (Eq. 8) and initialize $\boldsymbol{v}^* \leftarrow \boldsymbol{v}_{\text{init}}$
for $t = 0$ **to** $T - 1$ **do**
  $\widehat{\boldsymbol{W}}_{L,t} \leftarrow \boldsymbol{W}_L + (\boldsymbol{v}^* - \boldsymbol{W}_L \boldsymbol{k}_1^L)\boldsymbol{M}$ ▷ Phase 1: Virtual Look-ahead
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{v}^*} \mathcal{L}_{\text{meta}}(\widehat{\boldsymbol{W}}_{L,t})$ ▷ Phase 2: hypergradient via Eq. 9
  $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* - \eta\, \boldsymbol{g}_t$ ▷ Refine target with downstream-aware feedback
end for
$\widehat{\boldsymbol{W}} \leftarrow$ Standard multi-layer editing$(\boldsymbol{W}, \boldsymbol{v}_T^*; \mathcal{L})$ ▷ Final downstream update (e.g., AlphaEdit)
return $\widehat{\boldsymbol{W}}$

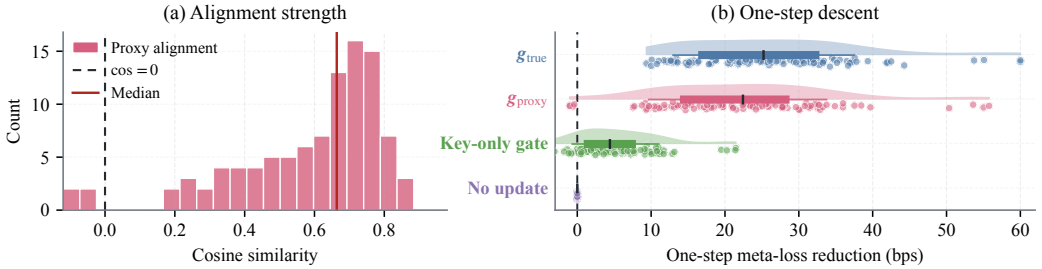

Figure 3: Fidelity of the Structural Gradient Proxy. (a) Cosine similarity between $\boldsymbol{g}_{\text{proxy}}$ and $\boldsymbol{g}_{\text{true}}$. $\boldsymbol{g}_{\text{proxy}}$ aligns with $\boldsymbol{g}_{\text{true}}$ in 96% of $N = 100$ edits, with 72% above $\cos > 0.5$. (b) One-step meta-loss reduction. The proxy achieves 22.2 bps reduction, recovering 85% of the $\boldsymbol{g}_{\text{true}}$ descent effect.

Phase 1: Virtual Look-ahead. At iteration t, MetaKE simulates the edit effect by forming virtual weights for the final edited layer as $\widehat{\boldsymbol{W}}_{L,t} = \boldsymbol{W}_L + \boldsymbol{\Delta}_{\text{proxy}}(\boldsymbol{v}_t^*)$. This step assesses how the current target would be realized under the AlphaEdit-style projection and previous-edit preservation structure, without differentiating through the full multi-layer editing process.

Phase 2: Downstream-Aware Feedback Correction. MetaKE evaluates $\mathcal{L}_{\text{meta}}$ under the virtual weights and backpropagates the signal through the structural gate $\boldsymbol{M}^T$. The resulting update refines $\boldsymbol{v}^*$ using an approximate downstream-aware feedback signal. After T iterations, the refined target is passed to the standard multi-layer editing procedure to perform the final parameter update across the edited layers $\mathcal{L}$. Under standard local regularity assumptions, the convergence analysis of the proxy-driven iteration is deferred to Appendix A.4.

5 Theoretical Justification

Sec. 4.1 identifies that the trade-off bottleneck stems from optimizing $\boldsymbol{v}^*$ without access to the downstream constraints that determine each β_i. MetaKE grants this access through the implicit map $\boldsymbol{\Delta}_l^*(\boldsymbol{v}^*)$: in Eq. (9), the Task Signal carries the requirements of Theorem 1, while the Structural Gate $\boldsymbol{M}^T$ exposes the request-specific association at layer L through $\boldsymbol{P}_L$ and $\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T$. The resulting feedback is no longer association-blind, and the residual scale is adapted per request.

Proxy fidelity. We next formalize when the structural proxy preserves the orientation of the full multi-layer hypergradient. Let $\boldsymbol{g}_{\text{true}}$ denote the exact full hypergradient of the outer objective, including the direct anchoring derivative, and let $\boldsymbol{g}_{\text{proxy}}$ be the corresponding full proxy hypergradient induced by the frozen structural gate. The following result shows that the proxy remains a descent direction when the non-final-layer contribution and final-layer proxy gap are jointly controlled.

Theorem 2 (Conditional Fidelity of the Structural Gradient Proxy)**.** *Let $F(\boldsymbol{v}^*)$ be the exact outer objective, with exact hypergradient $\boldsymbol{g}_{\text{true}}$ and structural proxy gradient $\boldsymbol{g}_{\text{proxy}}$. Under local fidelity conditions, if $\xi + \frac{K(1+\xi)}{\alpha_0}\varepsilon_{\text{AE}} < \chi(\boldsymbol{v}^*)$, where $\chi(\boldsymbol{v}^*) = \|\boldsymbol{g}_{\text{proxy}}\|_2 / \|\boldsymbol{h}_{\text{proxy}}\|_2$ and $\boldsymbol{h}_{\text{proxy}}$ is the downstream-mediated component of $\boldsymbol{g}_{\text{proxy}}$, then $\langle \boldsymbol{g}_{\text{true}}, \boldsymbol{g}_{\text{proxy}} \rangle > 0$. Thus $-\boldsymbol{g}_{\text{proxy}}$ is a descent direction for F; if F is locally smooth, small-enough steps along it decrease F. Proof in Appendix A.5.*

Table 1: Comparison of MetaKE with editing methods on the knowledge editing task. The best results are highlighted in red, while the second-best results are underlined.

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3 (8B)	MEND	$63.18_{\pm 0.34}$	$61.26_{\pm 0.32}$	$45.42_{\pm 0.41}$	$372.08_{\pm 0.75}$	$4.19_{\pm 0.06}$	$0.88_{\pm 0.05}$	$1.06_{\pm 0.05}$	$0.51_{\pm 0.03}$
	MALMEN	$38.82_{\pm 0.34}$	$49.01_{\pm 0.41}$	$56.64_{\pm 0.37}$	$524.48_{\pm 0.45}$	$19.83_{\pm 0.13}$	$39.65_{\pm 0.21}$	$40.56_{\pm 0.28}$	$30.47_{\pm 0.24}$
	ROME	$66.18_{\pm 0.47}$	$63.47_{\pm 0.41}$	$48.61_{\pm 0.25}$	$458.75_{\pm 0.23}$	$2.05_{\pm 0.06}$	$1.28_{\pm 0.04}$	$1.32_{\pm 0.03}$	$0.64_{\pm 0.03}$
	MEMIT	$65.52_{\pm 0.35}$	$63.94_{\pm 0.38}$	$51.26_{\pm 0.41}$	$452.55_{\pm 1.24}$	$5.94_{\pm 0.15}$	$36.15_{\pm 0.31}$	$31.42_{\pm 0.28}$	$18.57_{\pm 0.25}$
	PRUNE	$66.82_{\pm 0.42}$	$64.64_{\pm 0.37}$	$49.88_{\pm 0.25}$	$440.67_{\pm 1.12}$	$6.12_{\pm 0.09}$	$0.52_{\pm 0.05}$	$0.58_{\pm 0.07}$	$1.34_{\pm 0.04}$
	RECT	$64.53_{\pm 0.35}$	$62.35_{\pm 0.44}$	$60.78_{\pm 0.31}$	$520.94_{\pm 0.36}$	$19.21_{\pm 0.15}$	$87.41_{\pm 0.22}$	$82.35_{\pm 0.24}$	$32.18_{\pm 0.22}$
	PMET	$99.40_{\pm 0.21}$	$93.24_{\pm 0.79}$	$77.42_{\pm 0.80}$	$625.61_{\pm 0.52}$	$32.25_{\pm 0.35}$	$94.93_{\pm 0.27}$	$90.47_{\pm 0.54}$	$32.61_{\pm 0.66}$
	AlphaEdit	$99.23_{\pm 0.15}$	$86.21_{\pm 0.18}$	$77.16_{\pm 0.21}$	$624.95_{\pm 0.24}$	$30.25_{\pm 0.12}$	$94.27_{\pm 0.15}$	$88.35_{\pm 0.12}$	$32.65_{\pm 0.14}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.92_{\pm 0.05}$	$97.55_{\pm 0.46}$	$77.12_{\pm 0.86}$	$624.92_{\pm 0.24}$	$33.84_{\pm 0.42}$	$95.82_{\pm 0.46}$	$91.85_{\pm 0.42}$	$32.10_{\pm 0.68}$
	SPHERE	$99.75_{\pm 0.22}$	$88.38_{\pm 0.32}$	$74.84_{\pm 0.16}$	$616.53_{\pm 0.34}$	$28.67_{\pm 0.24}$	$94.25_{\pm 0.32}$	$88.35_{\pm 0.38}$	$32.02_{\pm 0.12}$
	MetaKE	$99.97_{\pm 0.02}$	$97.95_{\pm 0.21}$	$78.75_{\pm 0.35}$	$626.54_{\pm 0.42}$	$34.93_{\pm 0.35}$	$96.84_{\pm 0.14}$	$92.45_{\pm 0.16}$	$33.12_{\pm 0.21}$
GPT-J (6B)	MEND	$46.15_{\pm 0.50}$	$46.22_{\pm 0.51}$	$53.9_{\pm 0.48}$	$242.24_{\pm 0.39}$	$3.94_{\pm 0.04}$	$0.71_{\pm 0.04}$	$0.71_{\pm 0.04}$	$0.52_{\pm 0.03}$
	MALMEN	$63.37_{\pm 0.29}$	$59.87_{\pm 0.36}$	$41.40_{\pm 0.44}$	$229.56_{\pm 0.27}$	$20.35_{\pm 0.08}$	$33.3_{\pm 0.41}$	$31.68_{\pm 0.43}$	$3.81_{\pm 0.04}$
	ROME	$52.32_{\pm 0.24}$	$50.67_{\pm 0.32}$	$51.05_{\pm 0.65}$	$581.57_{\pm 0.22}$	$1.54_{\pm 0.04}$	$55.62_{\pm 0.24}$	$52.71_{\pm 0.33}$	$13.34_{\pm 0.26}$
	MEMIT	$98.12_{\pm 0.18}$	$95.71_{\pm 0.15}$	$63.74_{\pm 0.28}$	$594.15_{\pm 0.64}$	$39.33_{\pm 0.24}$	$96.85_{\pm 0.18}$	$93.52_{\pm 0.25}$	$31.42_{\pm 0.29}$
	PRUNE	$84.43_{\pm 0.29}$	$87.54_{\pm 0.26}$	$53.35_{\pm 0.34}$	$460.27_{\pm 0.65}$	$23.34_{\pm 0.21}$	$25.64_{\pm 0.16}$	$24.08_{\pm 0.22}$	$20.45_{\pm 0.13}$
	RECT	$98.83_{\pm 0.10}$	$86.25_{\pm 0.26}$	$72.54_{\pm 0.22}$	$616.68_{\pm 0.17}$	$41.43_{\pm 0.19}$	$96.18_{\pm 0.24}$	$91.75_{\pm 0.21}$	$28.25_{\pm 0.26}$
	PMET	$99.75_{\pm 0.16}$	$94.78_{\pm 0.53}$	$75.18_{\pm 0.78}$	$618.59_{\pm 0.49}$	$41.40_{\pm 0.32}$	$99.36_{\pm 0.19}$	$96.23_{\pm 0.51}$	$29.53_{\pm 0.80}$
	AlphaEdit	$99.75_{\pm 0.14}$	$96.47_{\pm 0.21}$	$76.14_{\pm 0.26}$	$618.41_{\pm 0.23}$	$42.36_{\pm 0.15}$	$99.56_{\pm 0.15}$	$95.84_{\pm 0.26}$	$28.31_{\pm 0.22}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.72_{\pm 0.17}$	$97.75_{\pm 0.53}$	$76.12_{\pm 0.86}$	$621.54_{\pm 0.75}$	$41.54_{\pm 0.35}$	$99.45_{\pm 0.25}$	$93.64_{\pm 0.48}$	$28.41_{\pm 0.72}$
	SPHERE	$99.80_{\pm 0.22}$	$95.40_{\pm 0.35}$	$76.12_{\pm 0.74}$	$611.21_{\pm 0.54}$	$43.20_{\pm 0.32}$	$99.69_{\pm 0.08}$	$97.10_{\pm 0.15}$	$28.09_{\pm 0.19}$
	MetaKE	$99.88_{\pm 0.10}$	$98.68_{\pm 0.24}$	$76.47_{\pm 0.38}$	$620.98_{\pm 0.35}$	$43.34_{\pm 0.26}$	$99.82_{\pm 0.12}$	$97.37_{\pm 0.15}$	$29.73_{\pm 0.20}$
GPT2-XL (1.5B)	MEND	$49.70_{\pm 0.49}$	$49.62_{\pm 0.46}$	$50.42_{\pm 0.52}$	$513.08_{\pm 0.25}$	$26.22_{\pm 0.08}$	$0.24_{\pm 0.05}$	$0.24_{\pm 0.05}$	$2.07_{\pm 0.03}$
	MALMEN	$52.60_{\pm 0.33}$	$50.99_{\pm 0.38}$	$49.76_{\pm 0.41}$	$536.67_{\pm 0.34}$	$12.84_{\pm 0.09}$	$32.99_{\pm 0.38}$	$33.00_{\pm 0.03}$	$2.62_{\pm 0.03}$
	ROME	$51.73_{\pm 0.38}$	$49.08_{\pm 0.21}$	$50.87_{\pm 0.24}$	$527.96_{\pm 0.89}$	$1.06_{\pm 0.05}$	$41.25_{\pm 0.45}$	$37.62_{\pm 0.38}$	$12.35_{\pm 0.22}$
	MEMIT	$94.83_{\pm 0.15}$	$85.62_{\pm 0.31}$	$60.21_{\pm 0.33}$	$472.14_{\pm 0.45}$	$22.08_{\pm 0.15}$	$79.58_{\pm 0.25}$	$72.45_{\pm 0.31}$	$26.12_{\pm 0.18}$
	PRUNE	$75.73_{\pm 0.28}$	$73.75_{\pm 0.21}$	$52.27_{\pm 0.16}$	$528.87_{\pm 0.19}$	$13.84_{\pm 0.21}$	$28.42_{\pm 0.26}$	$25.35_{\pm 0.24}$	$14.52_{\pm 0.18}$
	RECT	$92.43_{\pm 0.13}$	$81.35_{\pm 0.27}$	$65.06_{\pm 0.31}$	$483.17_{\pm 0.85}$	$20.46_{\pm 0.09}$	$83.45_{\pm 0.38}$	$75.28_{\pm 0.35}$	$24.93_{\pm 0.24}$
	PMET	$96.55_{\pm 0.58}$	$89.95_{\pm 0.72}$	$64.41_{\pm 0.66}$	$560.97_{\pm 1.74}$	$33.90_{\pm 0.49}$	$94.51_{\pm 0.38}$	$87.79_{\pm 0.62}$	$26.47_{\pm 0.68}$
	AlphaEdit	$99.42_{\pm 0.25}$	$94.35_{\pm 0.24}$	$66.05_{\pm 0.22}$	$592.17_{\pm 0.28}$	$38.85_{\pm 0.16}$	$93.52_{\pm 0.28}$	$83.65_{\pm 0.36}$	$25.85_{\pm 0.22}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.13_{\pm 0.18}$	$92.04_{\pm 0.52}$	$69.14_{\pm 0.69}$	$612.51_{\pm 0.68}$	$39.63_{\pm 0.35}$	$97.42_{\pm 0.45}$	$88.75_{\pm 0.82}$	$26.58_{\pm 0.75}$
	SPHERE	$99.30_{\pm 0.19}$	$89.10_{\pm 0.36}$	$72.35_{\pm 0.28}$	$616.41_{\pm 0.86}$	$39.43_{\pm 0.25}$	$91.41_{\pm 0.16}$	$81.92_{\pm 0.24}$	$26.74_{\pm 0.21}$
	MetaKE	$99.70_{\pm 0.18}$	$94.88_{\pm 0.31}$	$72.64_{\pm 0.26}$	$617.57_{\pm 0.25}$	$41.05_{\pm 0.28}$	$98.12_{\pm 0.18}$	$91.26_{\pm 0.32}$	$27.23_{\pm 0.25}$

The condition couples the two error sources: stronger non-final-layer leakage leaves less room for final-layer proxy error, with the admissible range decreasing as $(1-\xi)/(1+\xi)$, which gives a quantitative form of the Structural Consistency Hypothesis in Sec. 4.3. Fig. 3 evaluates the descent-direction implication of Theorem 2, across the tested edits, $\boldsymbol{g}_{\text{proxy}}$ is positively aligned with the true multi-layer hypergradient $\boldsymbol{g}_{\text{true}}$ and yields a positive one-step reduction of $\mathcal{L}_{\text{meta}}$ (Appendix C.5).

6 Experiments

In this section, we aim to answer three questions: (1) Does MetaKE improve editing performance over strong baselines? (2) Does MetaKE mitigate the disconnect between the two optimization stages, and which components are responsible for the gains? (3) Does MetaKE remain stable under long-horizon sequential editing while better preserving general capabilities and internal representation geometry?

6.1 Experimental Setup

Datasets & Evaluation Metrics. We evaluate our method on two benchmark datasets: ZsRE [38] and CounterFact [22]. We report five key metrics: Efficacy (success rate), Generalization (paraphrase robustness), Specificity (locality preservation), Fluency (generation entropy), and Consistency (reference mapping) to comprehensively assess the performance of MetaKE.

Baselines & Implementation Details. We compare MetaKE against KE baselines: ROME [22], MEMIT [13], PRUNE [39], RECT [40], PMET [20], AlphaEdit [14], BLUE [41], and SPHERE [42]. We also include meta-learning baselines MEND [25] and MALMEN [26]. Experiments are conducted on GPT2-XL (1.5B) [43], GPT-J (6B) [44], and LLaMA3 (8B) [45]. Unless otherwise noted, we use hyperparameters: shared settings $T = 15$, $\eta = 5 \times 10^{-3}$ for three models. The main results are averaged over three random seeds, and experiments are on a single NVIDIA A100 GPU.

6.2 Main Results

We assess MetaKE under the same sequential editing protocol established by [14], randomly sampling 2,000 editing instances with a batch size of 100. As shown in Table 1, MetaKE achieves the best results on most reported metrics across CounterFact and ZsRE for all three evaluated models. On ZsRE with GPT2-XL, MetaKE improves Generalization by 9.10% relative to AlphaEdit, indicating stronger robustness to paraphrased edit queries while maintaining competitive preservation performance. MetaKE attains notable gains on LLaMA3 over AlphaEdit$_{\text{BLUE}}$, improving Consistency by 3.22% and Specificity by 2.11% while preserving a near-perfect Efficacy of 99.97%. Overall, by aligning planned targets with the model's feasible geometry, MetaKE achieves stronger generalization while better preserving specificity, yielding a better trade-off between semantic accuracy and editability.

6.3 Analysis of the Two-Stage Optimization Disconnect

To assess the two-stage optimization disconnect in Sec. 4.1, we analyze association-dependent residual realization and compare high- and low-ρ efficacy curves under upstream KL sweeps. Details are provided in Appendix C.6.

Realized-residual attenuation. For each method, we sample 500 edit requests, compute the request-specific association ρ_i, and measure empirical realized coefficient $\widehat{\beta}_i$, which is the realized fraction of the planned residual after editing. Fig. 4 shows that MEMIT exhibits strong attenuation, especially in low-ρ regions; AlphaEdit improves the retained signal but leaves a clear realization gap. MetaKE achieves higher $\widehat{\beta}$ across association percentiles, suggesting that downstream-aware target optimization mitigates residual attenuation under downstream constraints.

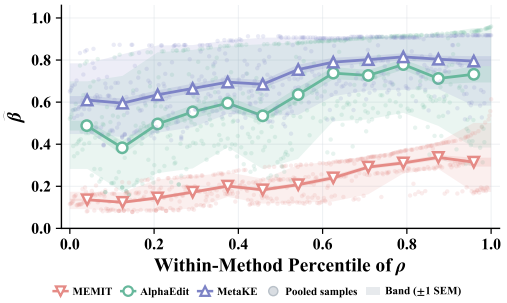

Figure 4: Empirical realization coefficient $\widehat{\beta}$ across association ρ.

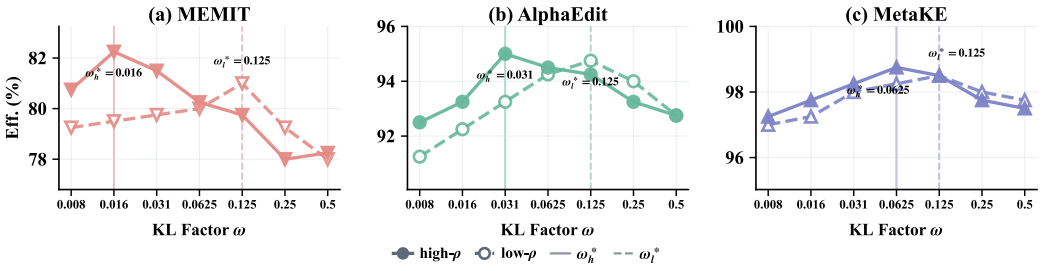

Figure 5: Efficacy of high- and low-ρ requests under different KL strengths.

Association-dependent requirements on KL strength. We further test whether requests with different downstream associations require different upstream KL strengths. For each method, we split 2,000 requests into high-ρ and low-ρ groups of 400 requests each, and sweep $\omega \in \{0.008, 0.016, 0.031, 0.0625, 0.125, 0.25, 0.5\}$. As shown in Fig. 5, MEMIT attains its best high-ρ Efficacy at a much smaller KL value than its best low-ρ Efficacy, indicating different favorable planning strengths. AlphaEdit reduces this discrepancy but still shows distinct group-wise optima. In contrast, MetaKE makes the two groups' favorable KL regions substantially overlap and maintains higher Efficacy across most KL values. Together with the residual-realization results above, this

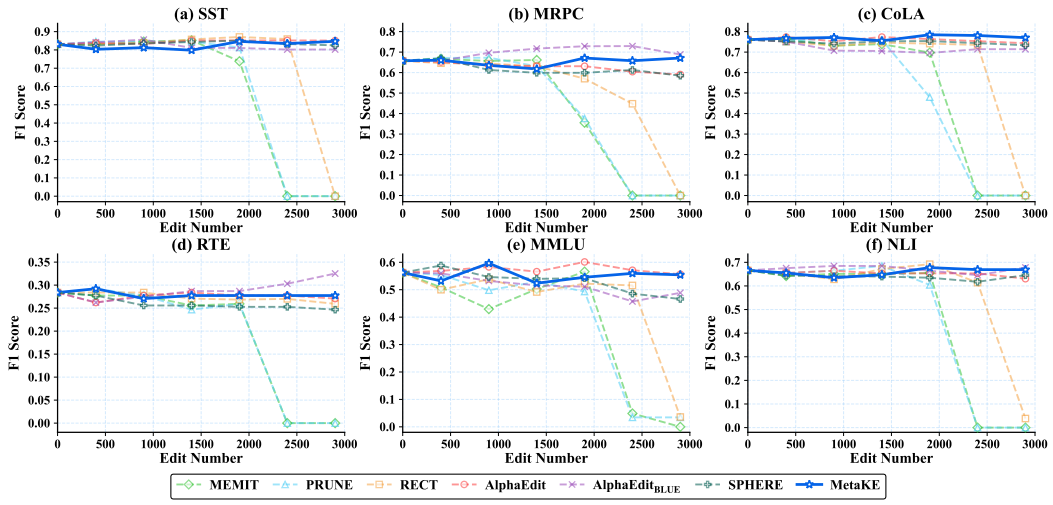

Figure 6: General-capability retention under long-horizon sequential editing on LLaMA3 (8B).

indicates that since the feasible residual scale depends on $\beta_i = \rho_i/(1+\rho_i)$, different association regimes favor different effective KL ranges; downstream-aware target optimization alleviates this limitation of association-blind planning, leading to a more favorable accuracy–editability trade-off. This comparison evaluates end-to-end sensitivity to the initial KL weight. The refinement-matched control in Appendix C.6 shows that generic refinement contributes to the gains, while the Structural Gate provides additional improvements and reduces KL sensitivity.

6.4 Ablation Study

We ablate MetaKE on LLaMA3 with ZsRE to identify its gain sources. We compare four variants: (i) w/o Bi-level Optimization, which directly uses $\boldsymbol{v}_{\text{init}}$ for final editing and removes iterative target refinement; (ii) w/o Structural Gate, which removes downstream structural filtering by replacing $\boldsymbol{M}$ with a dimension-compatible key-only gate obtained by dropping $\boldsymbol{P}_L$ and $\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T$ from Eq. (8); (iii) Stop-Grad Feedback, which keeps the virtual look-ahead forward pass but detaches $\boldsymbol{\Delta}_{\text{proxy}}(\boldsymbol{v}^*)$ before computing the edit and locality losses; and (iv) MetaKE.

Table 2: Ablation study on MetaKE.

Variant	Eff.↑	Gen.↑	Spe.↑
MetaKE	**96.84**	**92.45**	**33.12**
- w/o Bi-level Optimization	94.27	88.35	32.65
- w/o Structural Gate	95.88	90.75	32.14
- Stop-Grad Feedback	94.31	88.24	32.48

As shown in Table 2, removing bi-level optimization reduces Efficacy and Generalization, indicating that target refinement benefits from downstream constraints feedback. Stop-Grad Feedback performs close to w/o Bi-level Optimization, showing that virtual look-ahead alone is insufficient without differentiable downstream-aware feedback. Using the key-only gate also degrades all metrics, with a particularly notable relative drop in Specificity, suggesting that structural filtering through downstream constraints helps align the outer update with downstream constraints. Overall, MetaKE's gains come from target refinement driven by differentiable downstream-realization feedback and filtered by the Structural Gate. We further analyze the runtime and memory cost of this feedback path in Appendix D.4 and Appendix D.5; the results show that MetaKE improves the accuracy–editability trade-off with manageable overhead, while remaining within the same cost scale as existing locate-then-edit methods.

6.5 Sustainability of General Capabilities under Sequential Editing

To evaluate stability under sequential editing, we measure the post-edited model's general capability on six tasks from the GLUE benchmark [46], including SST [47], MRPC [48], CoLA [49], RTE [50], NLI [51], and MMLU [52]. We perform 3,000 sequential edit requests on LLaMA3, applying

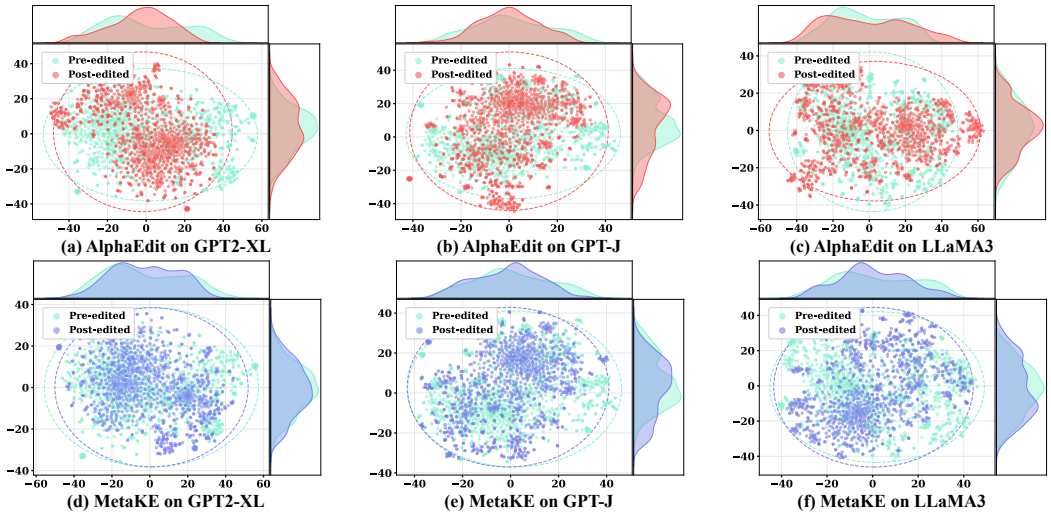

Figure 7: The distribution of hidden representations of pre-edited and post-edited LLMs.

batch size 100 and evaluating the general capability score every 500 edits. As shown in Fig. 6, MEMIT, PRUNE, and RECT all undergo sharp degradation, and in multiple tasks collapse to near-zero performance in the late editing stage. AlphaEdit, AlphaEdit$_{\text{BLUE}}$ and SPHERE remain considerably more stable. In contrast, MetaKE maintains nearly flat performance trajectories, and remains competitive with or better than the strongest baselines on most benchmarks.

6.6 Hidden Representations Analysis

The two-stage optimization disconnect can also manifest as shifts in internal representation geometry after editing. To examine this effect, we compare the hidden-state distributions of pre- and post-edit models. We extract hidden states for 1,000 randomly sampled factual queries and project them to a 2D plane using t-SNE [53]. As shown in Fig. 7, AlphaEdit exhibits a visible shift between pre- and post-edit representations across different backbones. In contrast, MetaKE produces a smaller distributional shift, suggesting that downstream-aware target optimization helps preserve the model's internal representation geometry while updating the target knowledge.

7 Conclusion

We identify a two-stage optimization disconnect in locate-then-edit KE, where upstream target optimization plans a residual without observing request-specific downstream constraints. MetaKE addresses this by refining the target representation with downstream-aware feedback, made tractable by a Structural Gradient Proxy. Experiments show that MetaKE mitigates residual attenuation, reduces association-blind KL limitations, and improves the accuracy–editability trade-off. Future work will extend this principle to more diverse and general nonlinear editing settings.

Acknowledgments and Disclosures

This work was supported by the National Natural Science Foundation of China under Grant Nos. 42550187 and 62476191, with computing resources provided by our research group. The authors declare no competing interests.

References

[1] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, and N. Zhang, "Editing large language models: Problems, methods, and opportunities," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 10 222–10 240.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[4] N. Zhang, Y. Yao, B. Tian, P. Wang, S. Deng, M. Wang, Z. Xi, S. Mao, J. Zhang, Y. Ni *et al.*, "A comprehensive study of knowledge editing for large language models," *arXiv preprint arXiv:2401.01286*, 2024.

[5] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM computing surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[6] E. Ferrara, "Should chatgpt be biased? challenges and risks of bias in large language models," *Challenges and Risks of Bias in Large Language Models (October 26, 2023)*, 2023.

[7] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 817–15 831.

[8] A. Liska, T. Kocisky, E. Gribovskaya, T. Terzi, E. Sezener, D. Agrawal, C. D. M. D'Autume, T. Scholtes, M. Zaheer, S. Young *et al.*, "Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models," in *International Conference on Machine Learning*. PMLR, 2022, pp. 13 604–13 622.

[9] J. A. Anderson, "A simple neural network generating an interactive memory," *Mathematical biosciences*, vol. 14, no. 3-4, pp. 197–220, 1972.

[10] A. Lazaridou, A. Kuncoro, E. Gribovskaya, D. Agrawal, A. Liska, T. Terzi, M. Gimenez, C. de Masson d'Autume, T. Kocisky, S. Ruder *et al.*, "Mind the gap: Assessing temporal generalization in neural language models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 348–29 363, 2021.

[11] N. De Cao, W. Aziz, and I. Titov, "Editing factual knowledge in language models," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6491–6506.

[12] T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim, and M. Ghassemi, "Aging with grace: Lifelong model editing with discrete key-value adaptors," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47 934–47 959, 2023.

[13] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," in *The Eleventh International Conference on Learning Representations*, 2023.

[14] J. Fang, H. Jiang, K. Wang, Y. Ma, J. Shi, X. Wang, X. He, and T.-S. Chua, "Alphaedit: Null-space constrained knowledge editing for language models," in *The Thirteenth International Conference on Learning Representations*, 2025.

[15] A. Gupta, D. Sajnani, and G. Anumanchipalli, "A unified framework for model editing," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 15 403–15 418.

[16] A. Gupta, P. Prateepamornkul, M. Lu, A. Alaa, T. Hartvigsen, and G. Anumanchipalli, "Lifelong knowledge editing requires better regularization," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 22 653–22 675. [Online]. Available: https://aclanthology.org/2025.findings-emnlp.1234/

[17] B. K. Baghel, E. Jordan, Z. R. Shi, and X. L. Li, "Resolving underedit & overedit with iterative & neighbor-assisted model editing," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025, pp. 14 786–14 808.

[18] P. Wang, Z. Tang, K. Zhou, J. Li, Q. Zhu, and M. Zhang, "Revealing and mitigating over-attention in knowledge editing," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=4l3AH8Bhmt

[19] Z. Zhao, G. Xu, X. Li, K. Wei, and J. Zhong, "Fedleke: Federated locate-then-edit knowledge editing for multi-client collaboration," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 14 247–14 258.

[20] X. Li, S. Li, S. Song, J. Yang, J. Ma, and J. Yu, "Pmet: Precise model editing in a transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 18 564–18 572.

[21] G. Jeong, J. Sun, S. Lee, and H. Kim, "STEAM: A semantic-level knowledge editing framework for large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2025*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 11 008–11 023. [Online]. Available: https://aclanthology.org/2025.findings-emnlp.585/

[22] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *Advances in neural information processing systems*, vol. 35, pp. 17 359–17 372, 2022.

[23] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 8493–8502.

[24] P. Wang, B. Zhou, X. Tang, J. Han, and S. Hu, "LyapLock: Bounded knowledge preservation in sequential large language model editing," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 6434–6459. [Online]. Available: https://aclanthology.org/2025.emnlp-main.327/

[25] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale," in *International Conference on Learning Representations*, 2022.

[26] C. Tan, G. Zhang, and J. Fu, "Massive editing for large language models via meta learning," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=L6L1CJQ2PE

[27] N. Zhang, B. Tian, S. Cheng, X. Liang, Y. Hu, K. Xue, Y. Gou, X. Chen, and H. Chen, "Instructedit: instruction-based knowledge editing for large language models," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024, pp. 6633–6641.

[28] Z. Li, H. Jiang, H. Chen, B. Bi, Z. Zhou, F. Sun, J. Fang, and X. Wang, "Reinforced lifelong editing for language models," in *International Conference on Machine Learning*. PMLR, 2025, pp. 34 920–34 942.

[29] Q. Li and X. 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)*, 2025, pp. 4127–4149.

[30] H. Jiang, J. Fang, N. Zhang, M. Wan, G. Ma, X. Wang, X. He, and T.-S. Chua, "Anyedit: Edit any knowledge encoded in language models," in *Forty-second International Conference on Machine Learning*, 2025.

[31] J. Yan, F. Wang, Y. Luo, Y. Li, and Y. Zhang, "Keys to robust edits: From theoretical insights to practical advances," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 22 545–22 560.

[32] T. Zhang, Q. Chen, D. Li, C. Wang, X. He, L. Huang, J. Huang *et al.*, "Dafnet: Dynamic auxiliary fusion for sequential model editing in large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 1588–1602.

[33] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5484–5495.

[34] T. Kohonen, "Correlation matrix memories," *IEEE transactions on computers*, vol. 100, no. 4, pp. 353–359, 1972.

[35] H. Park, G. Choi, M. Kim, and Y. Jo, "Context-robust knowledge editing for language models," in *Findings of the Association for Computational Linguistics: ACL 2025*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 10 360–10 385. [Online]. Available: https://aclanthology.org/2025.findings-acl.540/

[36] Y. Wei, X. Yu, R. Song, H. Peng, and A. Li, "Setke: knowledge editing for knowledge elements overlap," in *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, 2025, pp. 8295–8303.

[37] W. Fei, H. Shi, J. Xu, J. Peng, J. Li, J. Zhang, B. Bai, W. Han, Z. Chen, and X. Niu, "Scaling knowledge editing in LLMs to 100,000 facts with neural KV database," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=Z0CX62CSJQ

[38] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, R. Levy and L. Specia, Eds. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 333–342. [Online]. Available: https://aclanthology.org/K17-1034/

[39] J.-Y. Ma, H. Wang, H.-X. Xu, Z.-H. Ling, and J.-C. Gu, "Perturbation-restrained sequential model editing," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=bfI8cp8qmk

[40] J.-C. Gu, H.-X. Xu, J.-Y. Ma, P. Lu, Z.-H. Ling, K.-W. Chang, and N. 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*, 2024, pp. 16 801–16 819.

[41] X. Li, S. Wang, S. Li, S. Song, B. Ji, M. Jun, and J. Yu, "Rethinking residual distribution in locate-then-edit model editing," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. [Online]. Available: https://openreview.net/forum?id=P9gY05BDkW

[42] Q. Liu, J.-C. Gu, Y. Yao, H. Wang, and N. Peng, "Energy-regularized sequential model editing on hyperspheres," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=CHsdtzCip6

[43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[44] B. Wang and A. Komatsuzaki, "GPT-J-6B: A 6 billion parameter autoregressive language model," May 2021.

[45] Meta, "Llama 3," 2024, large language model release. [Online]. Available: https://llama.meta.com/llama3/

[46] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, 2018, pp. 353–355.

[47] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[48] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the third international workshop on paraphrasing (IWP2005)*, 2005.

[49] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 625–641, 2019.

[50] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo, "The fifth pascal recognizing textual entailment challenge." *TAC*, vol. 7, no. 8, p. 1, 2009.

[51] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)*, 2018, pp. 1112–1122.

[52] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=d7KBjmI3GmQ

[53] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, pp. 2579–2605, Nov. 2008.

[54] Y. Li, S. Bubeck, R. Eldan, A. Del Giorno, S. Gunasekar, and Y. T. Lee, "Textbooks are all you need ii: phi-1.5 technical report," *arXiv preprint arXiv:2309.05463*, 2023.

[55] Q. Team, "Qwen3 technical report," 2025. [Online]. Available: https://arxiv.org/abs/2505.09388

A Theoretical Analysis

A.1 Proof of Proposition 1

Let $\boldsymbol{K}_1$ denote the keys of the current to-be-updated knowledge, $\boldsymbol{K}_0$ the keys of preserved knowledge, and $\boldsymbol{K}_p$ the keys of previously updated knowledge. For a current edit, let $\boldsymbol{k}_1 \in \boldsymbol{K}_1$ be the current edit key and let $\boldsymbol{r}$ be the planned residual induced by Stage I. $\boldsymbol{r} = \boldsymbol{v}^* - \boldsymbol{W}\boldsymbol{k}_1$, or, under multi-layer residual allocation, $\boldsymbol{r}$ denotes the effective residual assigned to the current edited layer.

Scope of the surrogate. The quadratic surrogate fixes current keys, planned residual, and constraint statistics, and analyzes the local downstream realization step rather than the full nonlinear Transformer. Nonlinearities affect the result through locally evaluated keys and higher-order remainders, so Proposition 1 should be read as a first-order local characterization. When hidden states and keys vary smoothly with the update, the deviation from the local surrogate can be written as a nonlinear residual $\boldsymbol{e}_{\mathrm{nl}}$, with $\|\boldsymbol{e}_{\mathrm{nl}}\| = O(\|\boldsymbol{\Delta}\|_F^2)$ in a sufficiently small neighborhood. Thus, $\beta = \rho/(1+\rho)$ captures the leading-order constrained realization, while activation nonlinearities such as SwiGLU are absorbed into the locally evaluated keys and the higher-order residual term.

MEMIT covariance constraints. For a current edit, consider the MEMIT local quadratic surrogate

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} \ \|\boldsymbol{\Delta}\boldsymbol{k}_1 - \boldsymbol{r}\|_2^2 + \mathrm{tr}(\boldsymbol{\Delta}\boldsymbol{C}_0\boldsymbol{\Delta}^T) + \|\boldsymbol{\Delta}\boldsymbol{K}_p\|_F^2, \tag{10}$$

where $\boldsymbol{C}_0$ is the covariance surrogate of preserved knowledge. Since

$$\|\boldsymbol{\Delta}\boldsymbol{K}_p\|_F^2 = \mathrm{tr}(\boldsymbol{\Delta}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{\Delta}^T), \tag{11}$$

define

$$\boldsymbol{A}_{\mathrm{M}} := \boldsymbol{C}_0 + \boldsymbol{K}_p\boldsymbol{K}_p^T. \tag{12}$$

Assume that $\boldsymbol{A}_{\mathrm{M}}$ is nonsingular on the considered subspace. The objective becomes

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} \ \|\boldsymbol{\Delta}\boldsymbol{k}_1 - \boldsymbol{r}\|_2^2 + \mathrm{tr}(\boldsymbol{\Delta}\boldsymbol{A}_{\mathrm{M}}\boldsymbol{\Delta}^T). \tag{13}$$

The first-order condition is

$$(\boldsymbol{\Delta}\boldsymbol{k}_1 - \boldsymbol{r})\boldsymbol{k}_1^T + \boldsymbol{\Delta}\boldsymbol{A}_{\mathrm{M}} = \boldsymbol{0}. \tag{14}$$

Therefore,

$$\boldsymbol{\Delta}^* = \boldsymbol{r}\boldsymbol{k}_1^T\left(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{k}_1\boldsymbol{k}_1^T\right)^{-1}. \tag{15}$$

The realized residual is

$$\boldsymbol{\delta}^{\mathrm{real}} = \boldsymbol{\Delta}^*\boldsymbol{k}_1 = \boldsymbol{r}\,\boldsymbol{k}_1^T\left(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{k}_1\boldsymbol{k}_1^T\right)^{-1}\boldsymbol{k}_1. \tag{16}$$

Define

$$\rho_{\mathrm{M}} = \boldsymbol{k}_1^T\boldsymbol{A}_{\mathrm{M}}^{-1}\boldsymbol{k}_1. \tag{17}$$

By the Sherman–Morrison identity,

$$\boldsymbol{k}_1^T\left(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{k}_1\boldsymbol{k}_1^T\right)^{-1}\boldsymbol{k}_1 = \frac{\rho_{\mathrm{M}}}{1+\rho_{\mathrm{M}}}. \tag{18}$$

Thus,

$$\boldsymbol{\delta}^{\mathrm{real}} = \frac{\rho_{\mathrm{M}}}{1+\rho_{\mathrm{M}}}\boldsymbol{r}. \tag{19}$$

AlphaEdit projected constraints. For AlphaEdit, preserved knowledge is handled by the null-space projector $\boldsymbol{P}$, which satisfies $\boldsymbol{P}^2 = \boldsymbol{P}$ and $\boldsymbol{P}^T = \boldsymbol{P}$. Let $\boldsymbol{B}$ denote the AlphaEdit update applied to the edited weights, i.e.,

$$\boldsymbol{B} := \boldsymbol{\Delta}_{\mathrm{AE}}.$$

Following the projected form of AlphaEdit, this applied update lies in the right-projected subspace, so

$$\boldsymbol{B} = \boldsymbol{B}\boldsymbol{P}.$$

This property implies $\boldsymbol{B}\boldsymbol{K}_0 = \boldsymbol{0}$, and therefore the preserved keys are unchanged by the applied perturbation.

Define the projected current-edit key as

$$\boldsymbol{q}_1 := \boldsymbol{P}\boldsymbol{k}_1.$$

Since $\boldsymbol{B} = \boldsymbol{B}\boldsymbol{P}$, we have

$$\boldsymbol{B}\boldsymbol{k}_1 = \boldsymbol{B}\boldsymbol{P}\boldsymbol{k}_1 = \boldsymbol{B}\boldsymbol{q}_1.$$

The single-request local objective can therefore be written over the applied

$$\boldsymbol{B}^* = \underset{\boldsymbol{B}=\boldsymbol{B}\boldsymbol{P}}{\arg\min}\ \|\boldsymbol{B}\boldsymbol{q}_1 - \boldsymbol{r}\|_2^2 + \|\boldsymbol{B}\|_F^2 + \|\boldsymbol{B}\boldsymbol{K}_p\|_F^2. \tag{20}$$

$$\|\boldsymbol{B}\boldsymbol{K}_p\|_F^2 = \operatorname{tr}\left(\boldsymbol{B}\boldsymbol{P}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P}\boldsymbol{B}^T\right). \tag{21}$$

Define

$$\boldsymbol{A}_{\mathrm{AE}} := \boldsymbol{I} + \boldsymbol{P}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P}. \tag{22}$$

Then

$$\boldsymbol{B}^* = \underset{\boldsymbol{B}=\boldsymbol{B}\boldsymbol{P}}{\arg\min}\ \|\boldsymbol{B}\boldsymbol{q}_1 - \boldsymbol{r}\|_2^2 + \operatorname{tr}(\boldsymbol{B}\boldsymbol{A}_{\mathrm{AE}}\boldsymbol{B}^T). \tag{23}$$

The first-order condition in the projected subspace is

$$(\boldsymbol{B}\boldsymbol{q}_1 - \boldsymbol{r})\boldsymbol{q}_1^T + \boldsymbol{B}\boldsymbol{A}_{\mathrm{AE}} = \boldsymbol{0}. \tag{24}$$

Hence,

$$\boldsymbol{B}^* = \boldsymbol{r}\boldsymbol{q}_1^T\left(\boldsymbol{A}_{\mathrm{AE}} + \boldsymbol{q}_1\boldsymbol{q}_1^T\right)^{-1}. \tag{25}$$

The realized residual on the original current edit key is

$$\boldsymbol{\delta}^{\mathrm{real}} = \boldsymbol{B}^*\boldsymbol{k}_1 = \boldsymbol{B}^*\boldsymbol{P}\boldsymbol{k}_1 = \boldsymbol{B}^*\boldsymbol{q}_1. \tag{26}$$

Therefore,

$$\boldsymbol{\delta}^{\mathrm{real}} = \boldsymbol{r}\,\boldsymbol{q}_1^T\left(\boldsymbol{A}_{\mathrm{AE}} + \boldsymbol{q}_1\boldsymbol{q}_1^T\right)^{-1}\boldsymbol{q}_1. \tag{27}$$

Define

$$\rho_{\mathrm{AE}} = \boldsymbol{q}_1^T\boldsymbol{A}_{\mathrm{AE}}^{-1}\boldsymbol{q}_1 = (\boldsymbol{P}\boldsymbol{k}_1)^T\left(\boldsymbol{I} + \boldsymbol{P}\boldsymbol{K}_p\boldsymbol{K}_p^T\boldsymbol{P}\right)^{-1}\boldsymbol{P}\boldsymbol{k}_1. \tag{28}$$

By the Sherman–Morrison identity,

$$\boldsymbol{q}_1^T\left(\boldsymbol{A}_{\mathrm{AE}} + \boldsymbol{q}_1\boldsymbol{q}_1^T\right)^{-1}\boldsymbol{q}_1 = \frac{\rho_{\mathrm{AE}}}{1+\rho_{\mathrm{AE}}}. \tag{29}$$

Thus,

$$\boldsymbol{\delta}^{\mathrm{real}} = \frac{\rho_{\mathrm{AE}}}{1+\rho_{\mathrm{AE}}}\boldsymbol{r}. \tag{30}$$

Remaining unrealized residual. Combining MEMIT and AlphaEdit derivations, both single-request updates satisfy

$$\boldsymbol{\delta}^{\mathrm{real}} = \beta\boldsymbol{r}, \qquad \beta = \frac{\rho}{1+\rho} \in [0,1). \tag{31}$$

Consequently, the unrealized residual is

$$\boldsymbol{r}^{\mathrm{rem}} := \boldsymbol{r} - \boldsymbol{\delta}^{\mathrm{real}} = (1-\beta)\boldsymbol{r} = \frac{1}{1+\rho}\boldsymbol{r}. \tag{32}$$

Batch-edit extension. For MEMIT-style constraints, let $\boldsymbol{K}_1 = [\boldsymbol{k}_{1,1}, \ldots, \boldsymbol{k}_{1,b}]$ and $\boldsymbol{R}$ collect the planned residuals of the current batch. The batch objective is

$$\boldsymbol{\Delta}^* = \arg\min_{\boldsymbol{\Delta}} \ \|\boldsymbol{\Delta}\boldsymbol{K}_1 - \boldsymbol{R}\|_F^2 + \mathrm{tr}(\boldsymbol{\Delta}\boldsymbol{A}_{\mathrm{M}}\boldsymbol{\Delta}^T). \tag{33}$$

The solution is

$$\boldsymbol{\Delta}^* = \boldsymbol{R}\boldsymbol{K}_1^T \left(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{K}_1\boldsymbol{K}_1^T\right)^{-1}. \tag{34}$$

Thus the realized batch residual is

$$\boldsymbol{\Delta}^*\boldsymbol{K}_1 = \boldsymbol{R}\boldsymbol{T}_{\mathrm{M}}, \tag{35}$$

where

$$\boldsymbol{T}_{\mathrm{M}} = \boldsymbol{K}_1^T \left(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{K}_1\boldsymbol{K}_1^T\right)^{-1} \boldsymbol{K}_1. \tag{36}$$

Let

$$\boldsymbol{S}_{\mathrm{M}} = \boldsymbol{K}_1^T \boldsymbol{A}_{\mathrm{M}}^{-1} \boldsymbol{K}_1. \tag{37}$$

By the Woodbury identity,

$$\boldsymbol{T}_{\mathrm{M}} = \boldsymbol{S}_{\mathrm{M}}(\boldsymbol{I} + \boldsymbol{S}_{\mathrm{M}})^{-1}. \tag{38}$$

For AlphaEdit-style constraints, define $\boldsymbol{Q}_1 := \boldsymbol{P}\boldsymbol{K}_1$. The batch projected solution is

$$\boldsymbol{B}^* = \boldsymbol{R}\boldsymbol{Q}_1^T \left(\boldsymbol{A}_{\mathrm{AE}} + \boldsymbol{Q}_1\boldsymbol{Q}_1^T\right)^{-1}. \tag{39}$$

The realized batch residual is

$$\boldsymbol{B}^*\boldsymbol{K}_1 = \boldsymbol{B}^*\boldsymbol{Q}_1 = \boldsymbol{R}\boldsymbol{T}_{\mathrm{AE}}, \tag{40}$$

where

$$\boldsymbol{T}_{\mathrm{AE}} = \boldsymbol{Q}_1^T \left(\boldsymbol{A}_{\mathrm{AE}} + \boldsymbol{Q}_1\boldsymbol{Q}_1^T\right)^{-1} \boldsymbol{Q}_1. \tag{41}$$

Let

$$\boldsymbol{S}_{\mathrm{AE}} = \boldsymbol{Q}_1^T \boldsymbol{A}_{\mathrm{AE}}^{-1} \boldsymbol{Q}_1. \tag{42}$$

By the Woodbury identity,

$$\boldsymbol{T}_{\mathrm{AE}} = \boldsymbol{S}_{\mathrm{AE}}(\boldsymbol{I} + \boldsymbol{S}_{\mathrm{AE}})^{-1}. \tag{43}$$

Thus, batch editing realizes planned residuals through a request-interaction matrix rather than independent scalar factors. When the batch contains a single current edit, the matrix form reduces to the scalar attenuation factor in Proposition 1. □

A.2 Proof of Theorem 1

By Proposition 1, the realized residual of request i satisfies

$$\boldsymbol{\delta}_i^{\mathrm{real}} = \beta_i \boldsymbol{r}_i, \qquad \beta_i = \frac{\rho_i}{1+\rho_i}. \tag{44}$$

Since $\rho_i \geq 0$, we have $\beta_i \in [0, 1)$. If $\beta_i = 0$ and $m > 0$, then $\|\boldsymbol{\delta}_i^{\mathrm{real}}\| = 0 < m$ for any finite $\|\boldsymbol{r}_i\|$, so semantic success is impossible. We therefore consider the non-degenerate case $\beta_i \in (0, 1)$.

Semantic success requires

$$\|\boldsymbol{\delta}_i^{\mathrm{real}}\| \geq m. \tag{45}$$

Substituting $\boldsymbol{\delta}_i^{\mathrm{real}} = \beta_i \boldsymbol{r}_i$ gives

$$\beta_i \|\boldsymbol{r}_i\| \geq m, \tag{46}$$

and hence

$$\|\boldsymbol{r}_i\| \geq \frac{m}{\beta_i}. \tag{47}$$

This gives the lower bound on the planned residual required for semantic success after downstream realization.

Downstream editability requires

$$\|\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\mathrm{real}}\| \leq \tau. \tag{48}$$

Using $\boldsymbol{\delta}_i^{\text{real}} = \beta_i \boldsymbol{r}_i$, we have

$$\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\text{real}} = (1 - \beta_i)\boldsymbol{r}_i. \tag{49}$$

Therefore,

$$\|\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\text{real}}\| = (1 - \beta_i)\|\boldsymbol{r}_i\|. \tag{50}$$

The downstream editability condition then implies

$$(1 - \beta_i)\|\boldsymbol{r}_i\| \le \tau, \tag{51}$$

and since $\beta_i < 1$,

$$\|\boldsymbol{r}_i\| \le \frac{\tau}{1 - \beta_i}. \tag{52}$$

This gives the upper bound on the planned residual imposed by downstream editability.

Combining the two requirements, any feasible planned residual must satisfy

$$\frac{m}{\beta_i} \le \|\boldsymbol{r}_i\| \le \frac{\tau}{1 - \beta_i}. \tag{53}$$

Thus, the feasible scale of $\boldsymbol{r}_i$ depends on β_i, and hence on the downstream association ρ_i through $\beta_i = \rho_i/(1 + \rho_i)$.

It remains to derive the infeasibility condition. The interval above is non-empty only if

$$\frac{m}{\beta_i} \le \frac{\tau}{1 - \beta_i}. \tag{54}$$

Since $\beta_i \in (0, 1)$, multiplying both sides by $\beta_i(1 - \beta_i) > 0$ gives

$$m(1 - \beta_i) \le \tau\beta_i. \tag{55}$$

Equivalently,

$$m \le (m + \tau)\beta_i, \tag{56}$$

which yields

$$\beta_i \ge \frac{m}{m + \tau}. \tag{57}$$

Therefore, if

$$\beta_i < \frac{m}{m + \tau}, \tag{58}$$

no finite choice of $\|\boldsymbol{r}_i\|$ can satisfy both semantic success and downstream editability under this local model. This completes the proof. □

Implication for shared KL regularization. If the shared upstream KL coefficient locally induces $\|\boldsymbol{r}_i(\omega)\| \approx a_i/\omega$, with $a_i > 0$ and $\omega > 0$, then Theorem 1 implies

$$\frac{m}{\beta_i} \le \frac{a_i}{\omega} \le \frac{\tau}{1 - \beta_i}. \tag{59}$$

Equivalently, the admissible range of ω for request i is

$$\Omega_i = [L_i, U_i] = \left[\frac{a_i(1 - \beta_i)}{\tau}, \frac{a_i\beta_i}{m}\right]. \tag{60}$$

If $\omega > U_i$, then

$$\|\boldsymbol{r}_i(\omega)\| < \frac{m}{\beta_i}, \tag{61}$$

so the realized residual satisfies

$$\|\boldsymbol{\delta}_i^{\text{real}}\| = \beta_i\|\boldsymbol{r}_i(\omega)\| < m. \tag{62}$$

Thus the planned residual is over-shrunk and semantic editing becomes insufficient.

If $\omega < L_i$, then

$$\|\boldsymbol{r}_i(\omega)\| > \frac{\tau}{1 - \beta_i}. \tag{63}$$

In this case, the upstream stage produces an over-large planned residual relative to the downstream realization factor. Since

$$\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\mathrm{real}} = (1-\beta_i)\boldsymbol{r}_i, \tag{64}$$

we obtain

$$\|\boldsymbol{r}_i - \boldsymbol{\delta}_i^{\mathrm{real}}\| = (1-\beta_i)\|\boldsymbol{r}_i(\omega)\| > \tau. \tag{65}$$

Therefore, under-regularization first manifests as an over-large planned residual, and this over-large planned residual reduces downstream editability by leaving an unrealized residual that exceeds the editability tolerance.

Now consider a high-association request h and a low-association request ℓ, with $\beta_h > \beta_\ell$. Their admissible coefficient intervals are

$$\Omega_h = \left[\frac{a_h(1-\beta_h)}{\tau}, \frac{a_h\beta_h}{m}\right], \qquad \Omega_\ell = \left[\frac{a_\ell(1-\beta_\ell)}{\tau}, \frac{a_\ell\beta_\ell}{m}\right]. \tag{66}$$

A single shared coefficient cannot adapt these intervals per request. In particular, whenever

$$U_h < L_\ell, \qquad \text{i.e.,} \qquad \frac{a_h\beta_h}{m} < \frac{a_\ell(1-\beta_\ell)}{\tau}, \tag{67}$$

any shared coefficient

$$\omega \in (U_h, L_\ell) \tag{68}$$

simultaneously over-shrinks the high-association request and under-regularizes the low-association request. The former yields insufficient realized editing, while the latter produces an over-large planned residual whose unrealized component violates the downstream editability tolerance.

A.3 A Conditional Necessity Analysis of Downstream-Aware Target Optimization

We provide a first-order necessity analysis for downstream-aware target optimization under the bi-level formulation in Sec. 4.2. The goal is not to claim that the upstream and downstream stages can never be decoupled. Instead, we show that such a decoupling is first-order valid only in special cases where the downstream-realization coupling term vanishes. In the generic case, optimizing the upstream target as an isolated semantic objective misses a term induced by the downstream solver.

For compactness, let $\boldsymbol{\delta}$ denote the active coordinates of the applied multi-layer downstream update, defined below. We write the inner downstream problem as

$$\boldsymbol{\delta}^*(\boldsymbol{v}) = \arg\min_{\boldsymbol{\delta}} \mathcal{J}_{\mathrm{inner}}(\boldsymbol{\delta}; \boldsymbol{v}). \tag{69}$$

The induced bi-level objective can be written as

$$F(\boldsymbol{v}) = \mathcal{M}(\boldsymbol{\delta}^*(\boldsymbol{v}), \boldsymbol{v}), \tag{70}$$

where $\mathcal{M}$ denotes the meta-loss evaluated after applying the virtual downstream update, including the edit loss, locality loss, and meta-regularization on $\boldsymbol{v}$. An isolated upstream analysis would keep only the direct target derivative $\nabla_{\boldsymbol{v}}\mathcal{M}$, thereby ignoring how $\boldsymbol{v}$ changes the downstream solution $\boldsymbol{\delta}^*(\boldsymbol{v})$.

Active-space parameterization. For projected downstream updates, the unconstrained matrix $\widetilde{\boldsymbol{\Delta}}_l$ is not identifiable because the inner objective depends on it only through the applied update $\boldsymbol{B}_l = \widetilde{\boldsymbol{\Delta}}_l \boldsymbol{P}_l$. Let $\boldsymbol{U}_l \in \mathbb{R}^{d_0 \times r_l}$ be an orthonormal basis of $\mathrm{range}(\boldsymbol{P}_l)$, so that $\boldsymbol{P}_l = \boldsymbol{U}_l \boldsymbol{U}_l^T$. We parameterize the applied update as

$$\boldsymbol{B}_l = \boldsymbol{A}_l \boldsymbol{U}_l^T, \qquad \boldsymbol{A}_l \in \mathbb{R}^{d_1 \times r_l},$$

and define

$$\boldsymbol{\delta} = \mathrm{vec}\left(\{\boldsymbol{A}_l\}_{l \in \mathcal{L}}\right).$$

The Hessian in Proposition 2 is therefore taken with respect to these active coordinates. This removes the structural null directions $\widetilde{\boldsymbol{\Delta}}_l(\boldsymbol{I} - \boldsymbol{P}_l)$, which do not affect the applied update or the inner objective.

Proposition 2 (First-order coupling induced by downstream realization)**.** *Assume that $\mathcal{J}_{\mathrm{inner}}$ and $\mathcal{M}$ are twice continuously differentiable in a neighborhood of $(\boldsymbol{\delta}^*(\boldsymbol{v}), \boldsymbol{v})$, and that the active-space*

Hessian $\boldsymbol{H}(\boldsymbol{v}) = \nabla^2_{\boldsymbol{\delta}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}(\boldsymbol{\delta}^*(\boldsymbol{v}); \boldsymbol{v})$ is nonsingular. Then $\boldsymbol{\delta}^*(\boldsymbol{v})$ is locally differentiable in $\boldsymbol{v}$, and the gradient of the bi-level objective satisfies

$$\nabla_{\boldsymbol{v}} F(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\mathcal{M} - \nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}\boldsymbol{H}(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{\delta}}\mathcal{M}, \tag{71}$$

where all derivatives on the right-hand side are evaluated at $(\boldsymbol{\delta}^*(\boldsymbol{v}), \boldsymbol{v})$. Consequently, isolated upstream optimization and bi-level target optimization have the same first-order target gradient at $\boldsymbol{v}$ if and only if

$$\nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}\boldsymbol{H}(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{\delta}}\mathcal{M} = \mathbf{0}. \tag{72}$$

Proof. The first-order optimality condition of the inner problem is

$$\nabla_{\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}(\boldsymbol{\delta}^*(\boldsymbol{v}); \boldsymbol{v}) = \mathbf{0}. \tag{73}$$

Since $\boldsymbol{H}(\boldsymbol{v})$ is nonsingular, the implicit function theorem implies that $\boldsymbol{\delta}^*(\boldsymbol{v})$ is locally differentiable. Differentiating the inner optimality condition with respect to $\boldsymbol{v}$ gives

$$\boldsymbol{H}(\boldsymbol{v})\frac{\partial \boldsymbol{\delta}^*}{\partial \boldsymbol{v}} + \nabla^2_{\boldsymbol{\delta}\boldsymbol{v}}\mathcal{J}_{\text{inner}} = \mathbf{0}. \tag{74}$$

Therefore,

$$\frac{\partial \boldsymbol{\delta}^*}{\partial \boldsymbol{v}} = -\boldsymbol{H}(\boldsymbol{v})^{-1}\nabla^2_{\boldsymbol{\delta}\boldsymbol{v}}\mathcal{J}_{\text{inner}}. \tag{75}$$

Applying the chain rule to $F(\boldsymbol{v}) = \mathcal{M}(\boldsymbol{\delta}^*(\boldsymbol{v}), \boldsymbol{v})$ yields

$$\nabla_{\boldsymbol{v}} F(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\mathcal{M} + \left(\frac{\partial \boldsymbol{\delta}^*}{\partial \boldsymbol{v}}\right)^{\top}\nabla_{\boldsymbol{\delta}}\mathcal{M}. \tag{76}$$

Substituting the expression for $\partial\boldsymbol{\delta}^*/\partial\boldsymbol{v}$, we obtain

$$\nabla_{\boldsymbol{v}} F(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\mathcal{M} - \left(\nabla^2_{\boldsymbol{\delta}\boldsymbol{v}}\mathcal{J}_{\text{inner}}\right)^T\left(\boldsymbol{H}(\boldsymbol{v})^{-1}\right)^T\nabla_{\boldsymbol{\delta}}\mathcal{M}. \tag{77}$$

Since $\boldsymbol{H}(\boldsymbol{v}) = \nabla^2_{\boldsymbol{\delta}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}$ is the Hessian of a twice continuously differentiable scalar objective in the active coordinates, it is symmetric. Hence

$$\left(\boldsymbol{H}(\boldsymbol{v})^{-1}\right)^T = \boldsymbol{H}(\boldsymbol{v})^{-1}.$$

Moreover,

$$\left(\nabla^2_{\boldsymbol{\delta}\boldsymbol{v}}\mathcal{J}_{\text{inner}}\right)^T = \nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}.$$

Therefore,

$$\nabla_{\boldsymbol{v}} F(\boldsymbol{v}) = \nabla_{\boldsymbol{v}}\mathcal{M} - \nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}\boldsymbol{H}(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{\delta}}\mathcal{M}, \tag{78}$$

which gives Eq. (71). The isolated upstream gradient is $\nabla_{\boldsymbol{v}}\mathcal{M}$, whereas the bi-level gradient is $\nabla_{\boldsymbol{v}} F$. These two gradients are equal exactly when the coupling term in Eq. (72) is zero. □

Proposition 2 shows that downstream-aware target optimization is generally required at first order: the upstream target $\boldsymbol{v}$ affects the meta-objective through the downstream solution $\boldsymbol{\delta}^*(\boldsymbol{v})$. However, the result also makes clear that decoupling is possible in special cases. For example, the coupling term vanishes if the downstream solution is locally insensitive to the target, i.e.,

$$\nabla^2_{\boldsymbol{\delta}\boldsymbol{v}}\mathcal{J}_{\text{inner}} = \mathbf{0};$$

if the meta-objective is already stationary with respect to the downstream update, i.e.,

$$\nabla_{\boldsymbol{\delta}}\mathcal{M} = \mathbf{0};$$

or if the outer sensitivity lies in the null space of the downstream-response operator,

$$\nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}\boldsymbol{H}(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{\delta}}\mathcal{M} = \mathbf{0}.$$

These are locally decoupled regimes in which isolated upstream optimization and bi-level target optimization share the same first-order target direction. Outside such cases, the downstream solver contributes a nonzero realization-dependent term, so treating $\boldsymbol{v}$ as an isolated semantic target misses part of the bi-level target gradient.

Corollary 1 (Isolated target stationarity is generally insufficient)**.** *Let $\boldsymbol{v}_0$ be stationary for the isolated upstream target gradient, i.e.,*

$$\nabla_{\boldsymbol{v}}\mathcal{M}(\boldsymbol{\delta}^*(\boldsymbol{v}_0), \boldsymbol{v}_0) = \mathbf{0}.$$

Define the downstream-realization coupling term at $\boldsymbol{v}_0$ as

$$\boldsymbol{c}(\boldsymbol{v}_0) = \nabla^2_{\boldsymbol{v}\boldsymbol{\delta}}\mathcal{J}_{\text{inner}}\boldsymbol{H}(\boldsymbol{v}_0)^{-1}\nabla_{\boldsymbol{\delta}}\mathcal{M}\Big|_{(\boldsymbol{\delta}^*(\boldsymbol{v}_0),\boldsymbol{v}_0)}.$$

If $\boldsymbol{c}(\boldsymbol{v}_0) \neq \mathbf{0}$, then $\boldsymbol{v}_0$ is not a stationary point of the bi-level objective F. Moreover, there exists a sufficiently small $\eta > 0$ such that

$$F(\boldsymbol{v}_0 + \eta\boldsymbol{c}(\boldsymbol{v}_0)) < F(\boldsymbol{v}_0).$$

Proof. By Proposition 2 and the isolated stationarity condition $\nabla_{\boldsymbol{v}}\mathcal{M} = \mathbf{0}$, we have

$$\nabla_{\boldsymbol{v}}F(\boldsymbol{v}_0) = -\boldsymbol{c}(\boldsymbol{v}_0).$$

Therefore, the directional derivative of F at $\boldsymbol{v}_0$ along $\boldsymbol{c}(\boldsymbol{v}_0)$ is

$$\frac{d}{d\eta}F(\boldsymbol{v}_0 + \eta\boldsymbol{c}(\boldsymbol{v}_0))\Big|_{\eta=0} = \nabla_{\boldsymbol{v}}F(\boldsymbol{v}_0)^\top\boldsymbol{c}(\boldsymbol{v}_0) = -\|\boldsymbol{c}(\boldsymbol{v}_0)\|_2^2 < 0.$$

Thus, by differentiability of F, there exists a sufficiently small $\eta > 0$ such that $F(\boldsymbol{v}_0 + \eta\boldsymbol{c}(\boldsymbol{v}_0)) < F(\boldsymbol{v}_0)$. □

This corollary formalizes the limitation of analyzing the upstream target as an isolated semantic objective. Such an analysis is first-order valid only when the coupling term vanishes. When the coupling is nonzero, the isolated stationary target can still be improved by moving along the downstream-realization direction, which motivates optimizing $\boldsymbol{v}^*$ as a meta-parameter through the downstream solver.

A.4 Optimization Properties of Algorithm 1

We analyze the optimization properties of Algorithm 1. The analysis is stated for the tractable objective induced by the Virtual Look-ahead step and then connected to the original multi-layer bi-level objective through the Structural Gradient Proxy error characterized by Theorem 2.

A.4.1 Virtual Look-ahead Objective

Following Sec. 4.3, define the Virtual Look-ahead objective as

$$\widehat{F}(\boldsymbol{v}^*) := \mathcal{L}_{\text{meta}}\left(f(\cdot; \boldsymbol{W}_L + \boldsymbol{\Delta}_{\text{proxy}}(\boldsymbol{v}^*))\right), \tag{79}$$

where

$$\boldsymbol{\Delta}_{\text{proxy}}(\boldsymbol{v}^*) = (\boldsymbol{v}^* - \boldsymbol{W}_L\boldsymbol{k}_1^L)\boldsymbol{M}, \tag{80}$$

and

$$\boldsymbol{M} = (\boldsymbol{k}_1^L)^T\boldsymbol{P}_L\left(\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T\boldsymbol{P}_L + \boldsymbol{k}_1^L(\boldsymbol{k}_1^L)^T\boldsymbol{P}_L + \boldsymbol{I}\right)^{-1}, \tag{81}$$

where $\boldsymbol{k}_1^L$ denotes the current edit key at the final edited layer L. Under this definition, the Downstream-Aware Feedback Correction in Algorithm 1 is gradient descent on $\widehat{F}$:

$$\boldsymbol{v}_{t+1}^* = \boldsymbol{v}_t^* - \eta\nabla\widehat{F}(\boldsymbol{v}_t^*). \tag{82}$$

Assumption 1 (Local smoothness and lower boundedness)**.** *The Virtual Look-ahead objective $\widehat{F}$ is continuously differentiable and lower bounded on a neighborhood $\mathcal{N}$ containing all iterates generated by Algorithm 1. Moreover, $\widehat{F}$ is $L_{\widehat{F}}$-smooth on $\mathcal{N}$, i.e.,*

$$\|\nabla\widehat{F}(\boldsymbol{v}_1^*) - \nabla\widehat{F}(\boldsymbol{v}_2^*)\|_2 \leq L_{\widehat{F}}\|\boldsymbol{v}_1^* - \boldsymbol{v}_2^*\|_2 \tag{83}$$

for all $\boldsymbol{v}_1^*, \boldsymbol{v}_2^* \in \mathcal{N}$.

Assumption 2 (Uniform Structural Gradient Proxy error)**.** *Let*

$$F(\boldsymbol{v}^*) = \mathcal{L}_{\text{meta}}\left(f(\cdot; \mathcal{W} + \boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*))\right) \tag{84}$$

denote the original outer level objective induced by the full multi-layer downstream constrained parameter optimization. There exists $\varepsilon_{\text{SGP}} \geq 0$ such that

$$\|\nabla F(\boldsymbol{v}^*) - \nabla\widehat{F}(\boldsymbol{v}^*)\|_2 \leq \varepsilon_{\text{SGP}}, \qquad \forall\boldsymbol{v}^* \in \mathcal{N}. \tag{85}$$

Under the conditions of Theorem 2, ε_{SGP} can be taken as a uniform upper bound of the Structural Gradient Proxy error over $\mathcal{N}$.

A.4.2 Convergence of Downstream-Aware Feedback Correction

Theorem 3 (Descent and first-order stationarity of Algorithm 1)**.** *Suppose Assumption 1 holds and Algorithm 1 uses a constant step size*

$$0 < \eta \leq \frac{1}{L_{\widehat{F}}}. \tag{86}$$

Then the iterates generated by Eq. (82) *satisfy*

$$\widehat{F}(\boldsymbol{v}_{t+1}^*) \leq \widehat{F}(\boldsymbol{v}_t^*) - \frac{\eta}{2}\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2. \tag{87}$$

Consequently,

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2 \leq \frac{2(\widehat{F}(\boldsymbol{v}_0^*) - \widehat{F}_{\inf})}{\eta T}, \tag{88}$$

and

$$\min_{0\leq t<T} \|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2 \leq \frac{2(\widehat{F}(\boldsymbol{v}_0^*) - \widehat{F}_{\inf})}{\eta T}, \tag{89}$$

where $\widehat{F}_{\inf}$ *is any lower bound of* $\widehat{F}$ *on* $\mathcal{N}$*.*

Proof. Since $\widehat{F}$ is $L_{\widehat{F}}$-smooth, the descent lemma gives

$$\begin{aligned}\widehat{F}(\boldsymbol{v}_{t+1}^*) \leq \widehat{F}(\boldsymbol{v}_t^*) + \left\langle \nabla \widehat{F}(\boldsymbol{v}_t^*), \boldsymbol{v}_{t+1}^* - \boldsymbol{v}_t^* \right\rangle \\ + \frac{L_{\widehat{F}}}{2}\|\boldsymbol{v}_{t+1}^* - \boldsymbol{v}_t^*\|_2^2.\end{aligned} \tag{90}$$

Substituting Eq. (82) yields

$$\widehat{F}(\boldsymbol{v}_{t+1}^*) \leq \widehat{F}(\boldsymbol{v}_t^*) - \eta\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2 + \frac{L_{\widehat{F}}\eta^2}{2}\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2. \tag{91}$$

Thus,

$$\widehat{F}(\boldsymbol{v}_{t+1}^*) \leq \widehat{F}(\boldsymbol{v}_t^*) - \eta\left(1 - \frac{L_{\widehat{F}}\eta}{2}\right)\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2. \tag{92}$$

Since $\eta \leq 1/L_{\widehat{F}}$, Eq. (87) follows. Summing Eq. (87) from $t = 0$ to $T - 1$ gives

$$\widehat{F}(\boldsymbol{v}_T^*) \leq \widehat{F}(\boldsymbol{v}_0^*) - \frac{\eta}{2}\sum_{t=0}^{T-1}\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2. \tag{93}$$

Using $\widehat{F}(\boldsymbol{v}_T^*) \geq \widehat{F}_{\inf}$ and rearranging proves Eq. (88). Eq. (89) follows because the minimum is upper bounded by the average. □ □

A.4.3 Connection to the Original Multi-layer Bi-level Objective

The previous theorem establishes convergence for the Virtual Look-ahead objective optimized during Algorithm 1. We next connect this result to the original multi-layer bi-level objective in Eq. (5).

Corollary 2 (Approximate stationarity for the original multi-layer bi-level objective)**.** *Suppose Assumptions 1 and 2 hold. Let* $\{\boldsymbol{v}_t^*\}_{t=0}^{T-1}$ *be the iterates generated by Algorithm 1. Then*

$$\min_{0\leq t<T} \|\nabla F(\boldsymbol{v}_t^*)\|_2^2 \leq \frac{4(\widehat{F}(\boldsymbol{v}_0^*) - \widehat{F}_{\inf})}{\eta T} + 2\varepsilon_{\mathrm{SGP}}^2. \tag{94}$$

Thus Algorithm 1 returns an $O(\varepsilon_{\mathrm{SGP}})$*-stationary point for the original multi-layer bi-level objective, up to the Structural Gradient Proxy error.*

Proof. By Assumption 2,

$$\|\nabla F(\boldsymbol{v}_t^*)\|_2 \leq \|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2 + \varepsilon_{\mathrm{SGP}}. \tag{95}$$

Using $(a + b)^2 \leq 2a^2 + 2b^2$, we obtain

$$\|\nabla F(\boldsymbol{v}_t^*)\|_2^2 \leq 2\|\nabla \widehat{F}(\boldsymbol{v}_t^*)\|_2^2 + 2\varepsilon_{\mathrm{SGP}}^2. \tag{96}$$

Taking the minimum over $t = 0, \ldots, T - 1$ and applying Eq. (89) proves Eq. (94). □

Theorem 3 shows that Algorithm 1 has the standard non-convex convergence guarantee on the Virtual Look-ahead objective optimized during target refinement. Corollary 2 further shows that the optimized target is approximately stationary for the original multi-layer bi-level objective, with the residual term controlled by the Structural Gradient Proxy error. Therefore, Theorem 2 does not claim exact equivalence between the Structural Gradient Proxy and full differentiation through the multi-layer process; it characterizes when the Structural Gate preserves a useful downstream constraint feedback signal for optimizing $\boldsymbol{v}^*$.

A.5 Proof for Theorem 2

We use $\|\cdot\|_2$ for the spectral norm of matrices and the Euclidean norm of vectors, and $\|\cdot\|_F$ for the Frobenius norm. Let

$$F(\boldsymbol{v}^*) = \mathcal{L}_{\text{meta}}(f(\cdot; \mathcal{W} + \boldsymbol{\Delta}\mathcal{W}^*(\boldsymbol{v}^*)))$$

be the exact outer-level objective. The direct derivative of the target anchoring term is

$$\boldsymbol{a}(\boldsymbol{v}^*) = 2(\boldsymbol{v}^* - \boldsymbol{v}_{\text{init}}).$$

We decompose the exact full multi-layer hypergradient as

$$\boldsymbol{g}_{\text{true}} = \boldsymbol{h}_{\text{true}} + \boldsymbol{a}(\boldsymbol{v}^*),$$

where the downstream-mediated component is written in vectorized form as

$$\boldsymbol{h}_{\text{true}} = \sum_{l \in \mathcal{L}} \boldsymbol{J}_l^T \boldsymbol{s}_l^\flat, \qquad \boldsymbol{s}_l^\flat := \text{vec}(\nabla_{\boldsymbol{\Delta}_l} \mathcal{L}_{\text{meta}}), \qquad \boldsymbol{J}_l := \frac{\partial \text{ vec}(\boldsymbol{\Delta}_l^*(\boldsymbol{v}^*))}{\partial \boldsymbol{v}^*}.$$

Here $\boldsymbol{s}_l^\flat \in \mathbb{R}^{d_1 d_0}$ and $\boldsymbol{J}_l \in \mathbb{R}^{d_1 d_0 \times d_1}$, so $\boldsymbol{J}_l^T \boldsymbol{s}_l^\flat \in \mathbb{R}^{d_1}$, matching the dimension of $\boldsymbol{v}^*$. Since the anchoring term does not depend on $\boldsymbol{\Delta}_l$, $\boldsymbol{s}_l^\flat$ is induced by the edit and locality losses.

Let

$$\boldsymbol{h}_L := \boldsymbol{J}_L^T \boldsymbol{s}_L^\flat$$

denote the final-layer downstream-mediated contribution. For the structural proxy, we keep the gradient in matrix form:

$$\boldsymbol{h}_{\text{proxy}} = \boldsymbol{S}_{\text{p}} \widetilde{\boldsymbol{M}}^T, \qquad \boldsymbol{S}_{\text{p}} := \nabla_{\boldsymbol{\Delta}_{\text{proxy}}} \mathcal{L}_{\text{meta}},$$

where $\boldsymbol{S}_{\text{p}} \in \mathbb{R}^{d_1 \times d_0}$ and $\widetilde{\boldsymbol{M}}^T \in \mathbb{R}^{d_0 \times 1}$, giving $\boldsymbol{h}_{\text{proxy}} \in \mathbb{R}^{d_1}$. Equivalently, this is the vectorized Jacobian contraction with $\partial \,\text{vec}(\boldsymbol{\Delta}_{\text{proxy}})/\partial \boldsymbol{v}^* = \widetilde{\boldsymbol{M}}^T \otimes \boldsymbol{I}_{d_1}$. The corresponding full proxy hypergradient is

$$\boldsymbol{g}_{\text{proxy}} = \boldsymbol{h}_{\text{proxy}} + \boldsymbol{a}(\boldsymbol{v}^*).$$

Thus the anchoring derivative is added exactly to both the true and proxy hypergradients, and

$$\boldsymbol{g}_{\text{true}} - \boldsymbol{g}_{\text{proxy}} = \boldsymbol{h}_{\text{true}} - \boldsymbol{h}_{\text{proxy}}.$$

Local fidelity conditions. We use three local conditions on the downstream-mediated components.

Condition 1 (Final-layer dominance). There exists $\xi \geq 0$ such that

$$\Big\| \sum_{l \in \mathcal{L},\, l \neq L} \boldsymbol{J}_l^T \boldsymbol{s}_l^\flat \Big\|_2 \leq \xi \|\boldsymbol{h}_L\|_2.$$

This bounds the non-final-layer downstream-mediated contribution relative to the final-layer signal.

Condition 2 (Final-layer proxy gap). There exist $K > 0$ and $\varepsilon_{\text{AE}} \geq 0$ such that

$$\|\boldsymbol{h}_L - \boldsymbol{h}_{\text{proxy}}\|_2 \leq K \varepsilon_{\text{AE}} \|\boldsymbol{S}_{\text{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2.$$

Here ε_{AE} summarizes the local AlphaEdit-form approximation error, including the gate-freezing discrepancy, the discarded gate-Jacobian term, and the sensitivity drift between the exact final-layer path and the proxy path. If the structural geometry is fixed within a local outer step, the gate-Jacobian component vanishes and ε_{AE} mainly captures sensitivity drift.

Condition 3 (Structural non-degeneracy). Assume $\boldsymbol{S}_{\mathrm{p}} \neq \mathbf{0}$, $\widetilde{\boldsymbol{M}} \neq \mathbf{0}$, and

$$\alpha(\boldsymbol{v}^*) := \frac{\|\boldsymbol{h}_{\mathrm{proxy}}\|_2}{\|\boldsymbol{S}_{\mathrm{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2} \geq \alpha_0 > 0.$$

This excludes the degenerate case in which the structural gate annihilates the downstream-mediated task signal.

Finally, define the full-gradient compatibility factor

$$\chi(\boldsymbol{v}^*) := \frac{\|\boldsymbol{g}_{\mathrm{proxy}}\|_2}{\|\boldsymbol{h}_{\mathrm{proxy}}\|_2}.$$

This factor accounts for the exact anchoring derivative in the full proxy hypergradient. When the anchoring term is zero or does not reduce the proxy direction norm, $\chi(\boldsymbol{v}^*) \geq 1$, and the condition reduces to the downstream-mediated form.

Proof. By adding and subtracting $\boldsymbol{h}_L$, we have

$$\|\boldsymbol{h}_{\mathrm{true}} - \boldsymbol{h}_{\mathrm{proxy}}\|_2 \leq \|\boldsymbol{h}_{\mathrm{true}} - \boldsymbol{h}_L\|_2 + \|\boldsymbol{h}_L - \boldsymbol{h}_{\mathrm{proxy}}\|_2.$$

Using Conditions 1 and 2,

$$\|\boldsymbol{h}_{\mathrm{true}} - \boldsymbol{h}_{\mathrm{proxy}}\|_2 \leq \xi \|\boldsymbol{h}_L\|_2 + K\varepsilon_{\mathrm{AE}} \|\boldsymbol{S}_{\mathrm{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2.$$

Condition 3 gives

$$\|\boldsymbol{S}_{\mathrm{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2 \leq \frac{\|\boldsymbol{h}_{\mathrm{proxy}}\|_2}{\alpha_0}.$$

Therefore,

$$\|\boldsymbol{h}_L\|_2 \leq \|\boldsymbol{h}_{\mathrm{proxy}}\|_2 + \|\boldsymbol{h}_L - \boldsymbol{h}_{\mathrm{proxy}}\|_2 \leq \left(1 + \frac{K\varepsilon_{\mathrm{AE}}}{\alpha_0}\right) \|\boldsymbol{h}_{\mathrm{proxy}}\|_2.$$

Substituting this bound yields

$$\|\boldsymbol{h}_{\mathrm{true}} - \boldsymbol{h}_{\mathrm{proxy}}\|_2 \leq \left[\xi + \frac{K(1+\xi)}{\alpha_0}\varepsilon_{\mathrm{AE}}\right] \|\boldsymbol{h}_{\mathrm{proxy}}\|_2.$$

Since the anchoring derivative is shared exactly by the true and proxy full hypergradients,

$$\|\boldsymbol{g}_{\mathrm{true}} - \boldsymbol{g}_{\mathrm{proxy}}\|_2 = \|\boldsymbol{h}_{\mathrm{true}} - \boldsymbol{h}_{\mathrm{proxy}}\|_2.$$

Using the definition of $\chi(\boldsymbol{v}^*)$, we obtain

$$\|\boldsymbol{g}_{\mathrm{true}} - \boldsymbol{g}_{\mathrm{proxy}}\|_2 \leq \frac{\xi + \frac{K(1+\xi)}{\alpha_0}\varepsilon_{\mathrm{AE}}}{\chi(\boldsymbol{v}^*)} \|\boldsymbol{g}_{\mathrm{proxy}}\|_2.$$

If

$$\xi + \frac{K(1+\xi)}{\alpha_0}\varepsilon_{\mathrm{AE}} < \chi(\boldsymbol{v}^*),$$

then

$$\|\boldsymbol{g}_{\mathrm{true}} - \boldsymbol{g}_{\mathrm{proxy}}\|_2 < \|\boldsymbol{g}_{\mathrm{proxy}}\|_2.$$

Hence, by Cauchy–Schwarz,

$$\begin{aligned}\langle \boldsymbol{g}_{\mathrm{true}}, \boldsymbol{g}_{\mathrm{proxy}} \rangle &= \|\boldsymbol{g}_{\mathrm{proxy}}\|_2^2 + \langle \boldsymbol{g}_{\mathrm{true}} - \boldsymbol{g}_{\mathrm{proxy}}, \boldsymbol{g}_{\mathrm{proxy}} \rangle \\ &\geq \|\boldsymbol{g}_{\mathrm{proxy}}\|_2 \left(\|\boldsymbol{g}_{\mathrm{proxy}}\|_2 - \|\boldsymbol{g}_{\mathrm{true}} - \boldsymbol{g}_{\mathrm{proxy}}\|_2\right) > 0.\end{aligned}$$

Thus $-\boldsymbol{g}_{\mathrm{proxy}}$ is a descent direction for the exact outer objective F. If F is locally smooth, the standard descent lemma implies that

$$F(\boldsymbol{v}^* - \eta \boldsymbol{g}_{\mathrm{proxy}}) < F(\boldsymbol{v}^*)$$

for all sufficiently small $\eta > 0$.

Equivalently, when $\chi(\boldsymbol{v}^*) > \xi$, the condition can be written as

$$\varepsilon_{\mathrm{AE}} < \frac{\alpha_0(\chi(\boldsymbol{v}^*) - \xi)}{K(1+\xi)}.$$

When $\chi(\boldsymbol{v}^*) \geq 1$, this recovers the simpler sufficient condition

$$\varepsilon_{\mathrm{AE}} < \frac{\alpha_0(1-\xi)}{K(1+\xi)}.$$

Table 3: Locality-constraint ablation on GPT-J/CounterFact

Locality constraint	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Cons.↑	Runtime/100 edits (s)↓
No Outer Locality Loss	99.95	98.82	75.97	620.43	43.16	459.33
Subject-Essence	99.90	98.75	76.76	620.63	43.53	517.94
Neighborhood-1	99.75	98.55	76.87	620.56	43.63	550.83
Neighborhood-5	99.50	98.35	77.21	620.14	43.69	773.26

Interpretation under long sequential editing. The bound also clarifies the role of editable capacity. As sequential edits accumulate, preservation and previous-edit constraints may occupy more projected directions, weakening the structural gate. This effect is captured by $\alpha(\boldsymbol{v}^*)$: a smaller α_0 tightens the admissible proxy-gap threshold $\alpha_0(\chi(\boldsymbol{v}^*) - \xi)/(K(1+\xi))$. Thus, the theorem is a conditional guarantee requiring the projected gate to retain an informative direction, rather than an unconditional guarantee under unlimited editing capacity. In practice, $\alpha(\boldsymbol{v}^*)$ or $\|\widetilde{\boldsymbol{M}}\|_2$ can be monitored, and the edit batch size or edited layer set can be adjusted when projected capacity becomes limited.

B Methodology Details

B.1 Target Optimization

The meta-loss in Eq. (7) consists of three terms.

Edit success loss. This term encourages the post-edit model to produce the desired target output y_e given the edit input x_e:

$$\mathcal{L}_{\text{edit}}(x_e, y_e; \widehat{\mathcal{W}}) = -\log \mathbb{P}_{\widehat{\mathcal{W}}}[y_e \mid x_e]. \tag{97}$$

Locality preservation loss. Following MEMIT-style target-optimization variants that construct $\boldsymbol{v}^*$ with likelihood and KL-preservation objectives [20, 35–37], we use subject-essence KL probe to regularize the meta-refined target. For an edit request with subject s_i, we instantiate the locality prompt as $x_{\text{loc}} =$ `{`s_i`} is a`. This prompt is used to preserve the pre-edit output distribution around the subject while optimizing the target representation. We match the output distribution of the post-edit model to that of the pre-edit model on a locality set x_{loc}:

$$\mathcal{L}_{\text{loc}}(x_{\text{loc}}; \widehat{\mathcal{W}}, \mathcal{W}) = D_{\text{KL}}\left(P_{\mathcal{W}}(y \mid x_{\text{loc}}) \| P_{\widehat{\mathcal{W}}}(y \mid x_{\text{loc}})\right). \tag{98}$$

Locality-Constraint Ablation. The Subject-Essence prompt in Eq. (98) is a lightweight subject-centered locality proxy, not a substitute for general locality evaluation. The main experiments measure Specificity on benchmark-native prompts distinct from this probe. We compare no outer locality loss, Subject-Essence, Neighborhood-1, and Neighborhood-5 on GPT-J/CounterFact (2,000 edits, batch size 100, seed 0), keeping all other optimization and realization settings fixed. For each edit, relation-matched neighborhood prompts are drawn from CounterFact records disjoint from the editing stream, validation set, and benchmark-provided Specificity prompts; the latter are used only for final evaluation. Neighborhood-m averages KL losses over m fixed prompts, keeping the locality-loss scale unchanged. As shown in Table 3, removing the outer locality loss changes Efficacy and Generalization only marginally but decreases Specificity and Consistency by 0.79 and 0.37 points, while reducing runtime by 11.32%. Relative to Subject-Essence, Neighborhood-1/5 improve Specificity by only $0.11/0.45$ points at $6.35\%/49.30\%$ higher runtime; Neighborhood-5 also lowers Efficacy and Generalization by 0.40 points each. Subject-Essence thus provides a favorable lightweight trade-off between locality preservation and efficiency.

Target anchoring. The term $\|\boldsymbol{v}^* - \boldsymbol{v}_{\text{init}}\|_2^2$ keeps the refined target close to the initial semantic plan. This prevents the proxy refinement from drifting too far from the target representation produced by the upstream target optimization stage. The downstream-aware information is introduced through the structural gate in Eq. (9), while this term serves as a semantic anchor and optimization stabilizer.

B.2 Layer-Specific Target Allocation

We clarify how the layer-specific target $\boldsymbol{t}_l(\boldsymbol{v}^*)$ in Eq. (6) is constructed from the global target $\boldsymbol{v}^*$. Following the standard multi-layer residual allocation in MEMIT and AlphaEdit-style editors, we first define the last edited layer as $L = \max(\mathcal{L})$. During the sequential update over edited layers,

let $\boldsymbol{h}_{1,L}^{(<l)}$ denote the current hidden state of the edit request at layer L, after applying the updates to edited layers before l and re-running the forward pass. The remaining target displacement before updating layer l is

$$\boldsymbol{d}_l(\boldsymbol{v}^*) = \boldsymbol{v}^* - \boldsymbol{h}_{1,L}^{(<l)}. \tag{99}$$

Let $\mathcal{L}_{\geq l} = \{j \in \mathcal{L} : j \geq l\}$ be the set of edited layers not yet processed. We allocate an equal portion of the remaining displacement to the current layer:

$$\boldsymbol{r}_l(\boldsymbol{v}^*) = \frac{1}{|\mathcal{L}_{\geq l}|}\boldsymbol{d}_l(\boldsymbol{v}^*). \tag{100}$$

The layer-specific target is then

$$\boldsymbol{t}_l(\boldsymbol{v}^*) = \boldsymbol{W}_l\boldsymbol{k}_1^l + \boldsymbol{r}_l(\boldsymbol{v}^*). \tag{101}$$

For consecutive edited layers, this recovers the MEMIT allocation $\boldsymbol{r}_l = (\boldsymbol{v}^* - \boldsymbol{h}_{1,L})/(L - l + 1)$. In the AlphaEdit-style realization used by MetaKE, the same $\boldsymbol{t}_l(\boldsymbol{v}^*)$ defines the residual

$$\boldsymbol{R}_l(\boldsymbol{v}^*) = \boldsymbol{t}_l(\boldsymbol{v}^*) - \boldsymbol{W}_l\boldsymbol{k}_1^l, \tag{102}$$

which is then realized through the projected downstream update with $\boldsymbol{P}_l$. For a batch of edits, the vectors above are stacked as columns, giving $\boldsymbol{R}_l = \boldsymbol{T}_l - \boldsymbol{W}_l\boldsymbol{K}_l$.

B.3 Batched Virtual Look-ahead

For clarity, the Structural Gradient Proxy in Sec. 4.3 and Algorithm 1 is written in single-request notation. In the batched implementation, the current edit keys at the final edited layer are stacked as

$$\boldsymbol{K}_1^L = [\boldsymbol{k}_{1,1}^L, \ldots, \boldsymbol{k}_{1,b}^L],$$

and the corresponding target residuals are stacked as

$$\boldsymbol{R}^L = [\boldsymbol{r}_1^L, \ldots, \boldsymbol{r}_b^L], \qquad \boldsymbol{r}_i^L = \boldsymbol{v}_i^* - \boldsymbol{W}_L\boldsymbol{k}_{1,i}^L.$$

The batched structural gate is

$$\boldsymbol{M}_{\mathcal{B}} = (\boldsymbol{K}_1^L)^T\boldsymbol{P}_L\left(\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T\boldsymbol{P}_L + \boldsymbol{K}_1^L(\boldsymbol{K}_1^L)^T\boldsymbol{P}_L + \boldsymbol{I}\right)^{-1},$$

and the virtual look-ahead update is computed as

$$\boldsymbol{\Delta}_{\text{proxy}}^{\mathcal{B}} = \boldsymbol{R}^L\boldsymbol{M}_{\mathcal{B}}.$$

The term $\boldsymbol{K}_1^L(\boldsymbol{K}_1^L)^T\boldsymbol{P}_L$ is the current-batch interaction term in the original AlphaEdit closed-form update. During each outer refinement step, $\boldsymbol{M}_{\mathcal{B}}$ acts as a frozen structural gate. Thus, the proxy accounts for dominant batch interaction in virtual look-ahead response while avoiding differentiation through changes in batch key statistics. This approximation may be less accurate for very large batches or batches containing highly correlated edit requests.

C Experimental Setup

This section supplements the main paper with a detailed account of the experimental setup. We summarize the benchmarks, evaluation criteria, comparison methods, and additional assessments used throughout our study. Our goal is to make the empirical protocol transparent, reproducible, and directly comparable to prior work on knowledge editing.

C.1 Datasets

ZsRE. ZsRE [38] is a widely used benchmark for factual editing in a question-answering format. Each sample corresponds to a prompt encoding a subject–relation query (s_i, r_i) together with an expected answer o_i. Following the standard evaluation pipeline in prior editing studies [22], the benchmark provides three types of inputs: the original query for measuring edit success, paraphrased queries for measuring semantic transfer, and locality probes for testing whether the edit remains confined to the intended fact. As a result, ZsRE is particularly suitable for evaluating paraphrase robustness and locality preservation.

CounterFact. CounterFact [22] is a counterfactual factual editing benchmark designed to test whether a model can replace an existing fact with a new one. Its locality examples are constructed by substituting the subject entity with semantically related alternatives while preserving the same relation, thereby enabling a fine-grained assessment of collateral changes. Beyond the standard editing criteria, CounterFact also includes natural language generation prompts, which makes it possible to evaluate the quality of generated text using fluency- and consistency-oriented metrics.

C.2 Evaluation Metrics

C.2.1 Metrics on ZsRE

On ZsRE, we adopt the standard triad of *efficacy*, *generalization*, and *specificity* [13, 22, 25].

Efficacy. Efficacy measures whether the edited model returns the desired target object when queried with the original edit prompt. Let f_θ denote the edited model, let x_i be the original query for the i-th edit request, and let o_i be the desired edited target. We compute efficacy as:

$$\mathbb{E}_i \left[\arg\max_o P_{f_\theta}(o \mid x_i) = o_i \right]. \tag{103}$$

Generalization. Generalization evaluates whether the updated fact is expressed consistently under semantically equivalent rephrasings. Let $N(x_i)$ denote the paraphrase set associated with x_i. The corresponding metric is the average accuracy over these paraphrased queries:

$$\mathbb{E}_i \, \mathbb{E}_{\tilde{x}_i \in N(x_i)} \left[\arg\max_o P_{f_\theta}(o \mid \tilde{x}_i) = o_i \right]. \tag{104}$$

Specificity. Specificity quantifies whether the modification stays local and avoids changing predictions on unrelated prompts. Let $O(x_i)$ denote out-of-scope neighborhood prompts for the i-th example, and let o_i^c be the model's original top-1 prediction on those prompts before editing. We define specificity as:

$$\mathbb{E}_i \, \mathbb{E}_{\bar{x}_i \in O(x_i)} \left[\arg\max_o P_{f_\theta}(o \mid \bar{x}_i) = o_i^c \right]. \tag{105}$$

C.2.2 Metrics on CounterFact

For CounterFact, we follow the common practice of comparing the edited (counterfactual) object against the original factual object [13, 22]. The evaluation focuses on whether the model's probability mass is successfully shifted toward the new fact across different prompt variants. For the i-th instance, let x_i be the original prompt, o_i the desired counterfactual target, and o_i^c the original fact.

Efficacy. We first measure whether the edited model prefers the target object to the original factual object on the original prompt:

$$\mathbb{E}_i \left[P_{f_\theta}(o_i \mid x_i) > P_{f_\theta}(o_i^c \mid x_i) \right]. \tag{106}$$

Generalization. We then evaluate whether this preference is preserved across paraphrased variants $\tilde{x}_i \in N(x_i)$:

$$\mathbb{E}_i \, \mathbb{E}_{\tilde{x}_i \in N(x_i)} \left[P_{f_\theta}(o_i \mid \tilde{x}_i) > P_{f_\theta}(o_i^c \mid \tilde{x}_i) \right]. \tag{107}$$

Specificity. To evaluate locality, we consider neighborhood prompts $\bar{x}_i \in O(x_i)$ that should remain unaffected by the edit. Specificity is defined as the proportion of such cases in which the model still assigns higher probability to the original fact than to the edited target:

$$\mathbb{E}_i \, \mathbb{E}_{\bar{x}_i \in O(x_i)} \left[P_{f_\theta}(o_i^c \mid \bar{x}_i) > P_{f_\theta}(o_i \mid \bar{x}_i) \right]. \tag{108}$$

Fluency. Since CounterFact also includes generation-based evaluation, we assess fluency by measuring repetitive patterns in generated text. Following prior work, we use an entropy-based score computed from the empirical bi-gram and tri-gram distributions:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) - \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \tag{109}$$

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

Consistency. We evaluate whether generated text remains aligned with external knowledge about the edited fact. Specifically, we compute cosine similarity between the TF–IDF representation of model output y_i and that of a reference Wikipedia passage y_i^{wiki} associated with the edited target:

$$\text{Consis.} = \cos\Big(\text{tfidf}(y_i), \text{tfidf}(y_i^{\text{wiki}})\Big). \tag{110}$$

C.3 Baseline Methods

MEND. MEND [25] is a meta-learning-based editing method that learns small auxiliary networks to transform the standard fine-tuning gradient into a targeted parameter update. By exploiting the low-rank structure of feed-forward gradients, it enables fast and efficient one-shot edits. It is a strong baseline for scalable editing, but its performance can degrade under long sequential editing streams.

MALMEN. MALMEN [26] extends gradient-based meta-editing to the multi-fact setting. Instead of directly summing parameter shifts from different edits, it formulates their aggregation as a least-squares problem and solves for a joint update. This makes it effective for large batched edits, though it is less tailored to long-horizon sequential editing.

ROME. ROME [22] is a representative locate-and-edit method that modifies a single feed-forward layer believed to be responsible for recalling the target fact. After identifying a causally important layer, it applies a rank-one parameter update to write the edited association into the model. Because it performs sparse intervention, ROME often exhibits strong locality, although its robustness on more complex or multi-fact settings can be limited.

MEMIT. MEMIT [13] generalizes ROME from a single-layer intervention to a multi-layer editing framework. Instead of concentrating the update at one location, MEMIT distributes a shared editing residual across a range of critical layers and solves the corresponding key–value adjustment via least squares. This makes it a strong baseline for large-scale or batched editing, but it can accumulate interference under long sequential editing streams.

PRUNE. PRUNE [39] is designed for more stable sequential editing by explicitly controlling spectral updates in the edited weight matrices. Its key idea is to preserve numerically stable directions while suppressing harmful ones, thereby reducing edit interference and slowing down performance degradation over repeated updates.

RECT. RECT [40] introduces regularization during the editing process to keep the updated parameters close to the pre-trained model. Compared with methods that optimize primarily for factual overwrite, RECT places greater emphasis on balancing edit success with the retention of broader language and reasoning capabilities.

PMET. PMET [20] is an optimization-based editing method that refines the key–value view of Transformer feed-forward networks. It jointly optimizes hidden states from both MHSA and FFN, but only uses the optimized FFN-side representations to update FFN weights. This more precise formulation often improves reliability, although it relies on stronger assumptions about the functional roles of Transformer components.

AlphaEdit. AlphaEdit [14] builds on MEMIT by incorporating null-space constraints into the residual update. Concretely, it projects the editing signal away from protected or previously occupied directions, so that newly written knowledge causes less interference with preserved content. This mechanism substantially improves stability in sequential editing while still retaining the flexibility of distributed multi-layer updates.

AlphaEdit$_{\text{BLUE}}$. AlphaEdit$_{\text{BLUE}}$ [41] further refines the AlphaEdit framework by limiting edits to a small set of empirically identified boundary layers. This design is motivated by the observation that factual interventions may exhibit depth-dependent sensitivity in Transformer models. Although such restriction can improve efficiency and reduce disturbance, it also imposes a stronger architectural bias that may be suboptimal when different facts rely on heterogeneous layerwise mechanisms.

SPHERE. SPHERE [42] is a sequential editing method designed to improve stability by preserving the hyperspherical uniformity of neuron weights. It projects editing signals into a sparse complementary subspace, reducing interference with dominant pretrained directions. This design improves robustness in long editing streams, though it depends on a geometric assumption about knowledge preservation in weight space.

C.4 Evaluation of General Capability

To assess whether sequential knowledge editing harms general language competence, we evaluate the edited models on several standard natural language understanding benchmarks. These benchmarks are excluded from the editing data and cover a range of linguistic abilities, including sentiment analysis, semantic matching, grammatical judgment, and inference.

SST. We adopt SST [47] to measure the effect of sequential editing on sentence-level semantic understanding. The task asks the model to assign binary sentiment labels to short sentences with diverse syntactic patterns. Since SST can be affected by small distributional changes, performance shifts may indicate degradation in basic language comprehension. We report classification accuracy.

MRPC. MRPC [48] tests whether the model identifies semantic equivalence between sentence pairs. It is useful for evaluating whether editing damages fine-grained paraphrase understanding. Following common evaluation protocols for this label-imbalanced task, we report F1 score.

MMLU. We include MMLU [52] to evaluate the preservation of broad-domain knowledge and reasoning ability. The benchmark spans many subject areas and requires factual recall as well as reasoning over task-specific information. We evaluate the models in both zero-shot and few-shot settings to separate the effect of editing from task-specific adaptation.

RTE. RTE [50] is used to examine whether repeated edits affect textual entailment ability. The task requires predicting whether a hypothesis is entailed by a given premise. We report accuracy as the measure of inference performance.

CoLA. We use CoLA [49] to assess grammatical acceptability judgment. Because CoLA targets subtle syntactic phenomena, it helps reveal whether editing introduces unintended changes to linguistic form. Following standard practice, we report Matthews correlation coefficient (MCC).

NLI. We further evaluate the edited models on a natural language inference benchmark [51]. The task requires classifying sentence-pair relations into entailment, contradiction, or neutral. This benchmark measures robustness in semantic reasoning, and we report accuracy.

C.5 Empirical Details for Structural Proxy Validation

This section details the empirical validation protocol for the structural proxy. All quantities are computed per edit; we report distributions over $N{=}100$ edits sampled uniformly from ZsRE, evaluated on GPT2-XL after the target representation $\boldsymbol{v}^*$ has converged.

True multi-layer hypergradient. For each edit, we compute the exact full multi-layer hypergradient $\boldsymbol{g}_{\text{true}}$ by backpropagating $\mathcal{L}_{\text{meta}}$ through the full sequential multi-layer downstream realization over the editable layer set $\mathcal{L}$. The downstream applies the update layer by layer and recomputes the layer-specific key statistics and current target representation after preceding layer updates, thereby preserving cross-layer dependencies in the lower-level response. We then differentiate the outer-level objective with respect to $\boldsymbol{v}^*$:

$$\boldsymbol{g}_{\text{true}} = \nabla_{\boldsymbol{v}^*} \mathcal{L}_{\text{meta}}(f(\cdot; \mathcal{W} + \Delta\mathcal{W}^*(\boldsymbol{v}^*))) \,.$$

Proxy hypergradient. The structural proxy gradient is computed as the full proxy hypergradient

$$\boldsymbol{g}_{\text{proxy}} = \boldsymbol{h}_{\text{proxy}} + 2(\boldsymbol{v}^* - \boldsymbol{v}_{\text{init}}), \qquad \boldsymbol{h}_{\text{proxy}} = \boldsymbol{S}_{\text{p}} \widetilde{\boldsymbol{M}}^T,$$

where $\boldsymbol{S}_{\text{p}}$ is the meta-loss gradient with respect to the proxy update and $\widetilde{\boldsymbol{M}}$ is the frozen structural gate at layer L. The first term $\boldsymbol{h}_{\text{proxy}}$ is the downstream-mediated component, while the second term is the exact direct derivative of the target anchoring term.

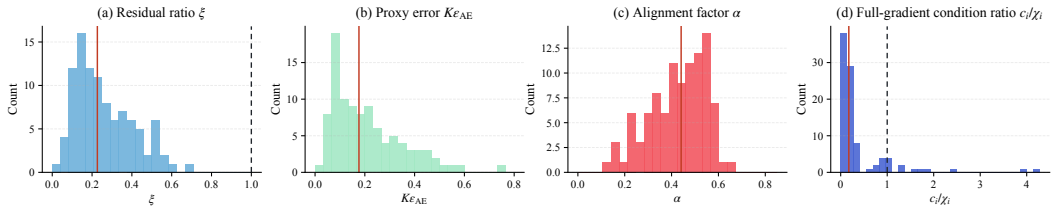

Figure 8: Per-edit diagnostics of the local fidelity conditions of Theorem 2 on $N = 100$ uniformly sampled ZsRE edits using GPT2-XL, evaluated after $\boldsymbol{v}^*$ converges. (a) Non-final-layer leakage ratio ξ. (b) Final-layer proxy gap $K\varepsilon_{\mathrm{AE}}$. (c) Structural non-degeneracy factor α. (d) Full-gradient sufficient-condition ratio c_i/χ_i, and values below 1 satisfy the full-gradient sufficient local descent-direction condition. Red lines mark medians, and dashed lines mark the threshold 1 where shown.

One-step descent. For each edit, we apply a single outer-level update

$$\boldsymbol{v}^{*+} = \boldsymbol{v}^* - \eta \boldsymbol{g}$$

along four directions: (i) $\boldsymbol{g}_{\mathrm{true}}$, (ii) $\boldsymbol{g}_{\mathrm{proxy}}$, (iii) a key-only-gate variant that replaces $\widetilde{\boldsymbol{M}}$ with $\widetilde{\boldsymbol{M}}_{\mathrm{key}}$, obtained by removing $\boldsymbol{P}_L$ and $\boldsymbol{K}_p^L(\boldsymbol{K}_p^L)^T$ from the structural gate, and (iv) No update. We report the relative one-step reduction in basis points of the current meta-loss:

$$\Delta_{\mathrm{rel}} = 10^4 \cdot \frac{\mathcal{L}_{\mathrm{meta}}(\boldsymbol{v}^*) - \mathcal{L}_{\mathrm{meta}}(\boldsymbol{v}^{*+})}{|\mathcal{L}_{\mathrm{meta}}(\boldsymbol{v}^*)| + \epsilon}.$$

The reported value is a one-step relative meta-loss reduction measured in basis points; it is used only to compare descent strength across update directions under the same step size.

Assumption diagnostics. The structural quantities ξ, $K\varepsilon_{\mathrm{AE}}$, and α are computed on the downstream-mediated components, because the anchoring derivative is exact and shared by the true and proxy full hypergradients. Per edit, we estimate

$$\xi = \frac{\left\| \sum_{l \neq L} \boldsymbol{J}_l^T \boldsymbol{s}_l^\flat \right\|_2}{\|\boldsymbol{h}_L\|_2 + \epsilon},$$

$$K\varepsilon_{\mathrm{AE}} = \frac{\|\boldsymbol{h}_L - \boldsymbol{h}_{\mathrm{proxy}}\|_2}{\|\boldsymbol{S}_{\mathrm{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2 + \epsilon},$$

and

$$\alpha = \frac{\|\boldsymbol{h}_{\mathrm{proxy}}\|_2}{\|\boldsymbol{S}_{\mathrm{p}}\|_F \|\widetilde{\boldsymbol{M}}\|_2 + \epsilon}.$$

We also compute the structural condition value

$$c = \xi + \frac{(1+\xi) K\varepsilon_{\mathrm{AE}}}{\alpha},$$

and the full-gradient compatibility factor

$$\chi = \frac{\|\boldsymbol{g}_{\mathrm{proxy}}\|_2}{\|\boldsymbol{h}_{\mathrm{proxy}}\|_2 + \epsilon}.$$

According to Theorem 2, the full-gradient sufficient condition is $c < \chi$.

As shown in Fig. 8, the diagnostics characterize the local fidelity regime underlying Theorem 2. Panel (a) shows that the non-final-layer leakage ratio stays below 1 for the sampled edits, with median $\xi = 0.23$, indicating that the final edited layer captures the dominant downstream-mediated contribution in this diagnostic setting. Panel (b) shows a moderate final-layer proxy gap, with median $K\varepsilon_{\mathrm{AE}} = 0.18$ and 95% quantile 0.51. Panel (c) shows that the structural non-degeneracy factor is not concentrated near zero, with 5% quantile $\alpha = 0.22$. Panel (d) combines these quantities with χ into the full-gradient condition ratio c/χ. Among finite diagnostic ratios, 88% satisfy $c/\chi < 1$, with median 0.17, while 12% exceed the sufficient-condition threshold and 6% of edits yield NaN or failed ratio estimates. These undefined cases arise from non-finite diagnostic quantities in the ratio computation and are reported separately. Since Theorem 2 gives a sufficient but not necessary condition, finite cases with $c/\chi \geq 1$ indicate that this certificate is inconclusive for those edits.

Table 4: Long-horizon proxy fidelity on GPT2-XL/ZsRE. Cosine is computed between the true and proxy hypergradients. The last two columns report one-step relative meta-loss reduction in basis points (bps).

H	Median cosine ↑	Positive alignment (%) ↑	Cosine > 0.5 (%) ↑	True-gradient reduction (bps) ↑	Proxy-gradient reduction (bps) ↑
100	0.664	96	70	25.61	21.73
1,000	0.648	95	70	25.74	21.89
3,000	0.653	96	71	26.20	22.01
5,000	0.635	94	68	25.55	21.42

Long-horizon proxy fidelity. We extend the diagnostic in this section to a 5,000-edit GPT2-XL/ZsRE stream with batch size 1. At edit counts $H = 100$, 1000, 3000, and 5000, we evaluate the same 100 held-out requests, used only for measurement and excluded from the edit history. We compare the full multi-layer hypergradient $\boldsymbol{g}_{\text{true}}$ with the structural proxy gradient $\boldsymbol{g}_{\text{proxy}}$, following the preceding protocol. Table 4 shows broadly stable alignment through 5,000 edits. At $H = 5000$, the proxy retains 83.84% of the true-gradient one-step reduction (21.42 versus 25.55 bps). Together with the sequential robustness results in Appendix D.3, this supports proxy utility over the tested horizon. Beyond 5,000 edits, accumulated constraints may reduce realizable update magnitude, while stronger cross-layer drift could impair gradient alignment.

C.6 Empirical Details for Sec 6.3

We provide the detailed protocols for the two diagnostics in Sec. 6.3. Both diagnostics are conducted on ZsRE with GPT-2 XL as the base model. We compare MEMIT, AlphaEdit, and MetaKE. MEMIT and AlphaEdit follow the standard locate-then-edit pipeline, while MetaKE builds on AlphaEdit and introduces downstream-aware bi-level target optimization. For each method, we keep the default editing layers, target objective, and Stage-II update structure from its implementation.

For every editing request i, we compute the request-specific association ρ_i according to the method-specific downstream constraint. For MEMIT, we use the covariance-based association

$$\rho_i = \boldsymbol{k}_{1,i}^T (\boldsymbol{C}_0 + \boldsymbol{K}_p \boldsymbol{K}_p^T)^{-1} \boldsymbol{k}_{1,i}, \tag{111}$$

where $\boldsymbol{C}_0 = \boldsymbol{K}_0 \boldsymbol{K}_0^T$. For AlphaEdit and MetaKE, we use the projected association

$$\rho_i = (\boldsymbol{P}\boldsymbol{k}_{1,i})^T (\boldsymbol{I} + \boldsymbol{P}\boldsymbol{K}_p \boldsymbol{K}_p^T \boldsymbol{P})^{-1} \boldsymbol{P}\boldsymbol{k}_{1,i}. \tag{112}$$

Empirical residual attenuation. This diagnostic tests the request-specific attenuation predicted by Proposition 1 and examines whether downstream-aware target optimization reduces the empirical loss of executable signal. We randomly sample 500 editing requests from ZsRE. For each request i and each method, we run the full editing pipeline and record the planned residual

$$\boldsymbol{r}_i = \boldsymbol{v}_i^* - \boldsymbol{W}\boldsymbol{k}_{1,i}, \tag{113}$$

and the realized residual

$$\boldsymbol{\delta}_i^{\text{real}} = \boldsymbol{\Delta}^* \boldsymbol{k}_{1,i}, \tag{114}$$

where $\boldsymbol{v}_i^*$ is the upstream target representation and $\boldsymbol{\Delta}^*$ is the downstream constrained update.

We estimate the empirical scalar realization coefficient by projecting the realized residual onto the planned residual:

$$\widehat{\beta}_i = \frac{\langle \boldsymbol{\delta}_i^{\text{real}}, \boldsymbol{r}_i \rangle}{\|\boldsymbol{r}_i\|_2^2 + \epsilon}, \tag{115}$$

where ϵ is a small numerical constant. Under the single-request local quadratic surrogate in Proposition 1, the corresponding scalar realization coefficient reduces to the theoretical value

$$\beta_i^\rho = \frac{\rho_i}{1 + \rho_i}. \tag{116}$$

Thus, $\widehat{\beta}_i$ measures how much of the planned residual is empirically realized along its intended direction. Because MEMIT and AlphaEdit use different downstream constraints, their raw ρ_i values

Table 5: Efficacy (%) in the refinement-matched KL sweep on GPT2-XL and ZsRE. H/L denote the high-/low-association groups selected using AlphaEdit's ρ_i and held fixed across methods.

ω	AlphaEdit		MetaKE w/o Structural Gate		MetaKE	
	H	L	H	L	H	L
0.008	91.92	91.25	93.08	91.47	97.00	97.00
0.016	93.00	92.08	94.42	93.00	97.67	97.17
0.031	94.33	93.33	96.25	94.92	98.17	97.83
0.0625	93.75	94.00	96.00	96.08	98.67	98.17
0.125	93.67	94.42	95.50	96.25	98.42	98.50
0.25	92.92	94.00	94.58	94.92	97.92	97.92
0.5	92.17	92.92	93.50	93.25	97.58	97.50

are not directly comparable in scale. Therefore, for each method independently, we sort requests by ρ_i and map them to a within-method percentile axis. We then divide the sorted requests into 13 quantile bins and report the mean $\widehat{\beta}$ in each bin, together with a ± 1 standard-error band. Pooled per-request samples are overlaid as translucent points.

Association-dependent requirements on KL strength. This diagnostic tests that if the feasible planned residual scale depends on the request-specific realization factor $\beta_i = \rho_i/(1+\rho_i)$, then high- and low-association requests may prefer different upstream KL strengths under a shared KL control.

We randomly sample 2,000 editing requests from ZsRE. For each method, we compute the request-specific association ρ_i at final edited layer using the corresponding association score defined above. We then form two disjoint request groups. The high-ρ group contains the top 20% of requests, i.e., 400 requests whose keys interact most strongly with the downstream constraints. The low-ρ group contains the bottom 20%, i.e., 400 requests with the weakest association. We sweep the upstream KL coefficient over the discrete grid $\omega \in \{0.008,\ 0.016,\ 0.031,\ 0.0625,\ 0.125,\ 0.25,\ 0.5\}$. The grid is centered at default $\omega = 0.0625$ and covers both weak and strong regularization regimes. For every method and every value of ω, we re-run the full editing pipeline on high-ρ and low-ρ requests, while leaving all other hyperparameters fixed. We use Efficacy as the main editing-success metric. Following the standard knowledge-editing evaluation protocol, Efficacy measures whether the post-edit model predicts the target answer on the editing prompt. In this diagnostic, Efficacy is used as an operational proxy for the semantic-success side of Theorem 1: the theorem's lower-bound condition concerns whether the realized update is sufficient to induce the target prediction.

For each method, group, and KL coefficient, we compute the group-wise average Efficacy:

$$\mathrm{Eff}_H(\omega) = \frac{1}{|H|}\sum_{i\in H}\mathrm{Eff}_i(\omega), \qquad \mathrm{Eff}_L(\omega) = \frac{1}{|L|}\sum_{i\in L}\mathrm{Eff}_i(\omega), \tag{117}$$

where H and L denote the high-ρ and low-ρ groups, respectively. We estimate the empirical optimal KL coefficient for each group by

$$\omega_H^* = \arg\max_{\omega\in\mathcal{G}} \mathrm{Eff}_H(\omega), \qquad \omega_L^* = \arg\max_{\omega\in\mathcal{G}} \mathrm{Eff}_L(\omega), \tag{118}$$

where $\mathcal{G} = \{0.008, 0.016, 0.031, 0.0625, 0.125, 0.25, 0.5\}$. When multiple KL values yield the same maximum Efficacy, we select the smaller ω as the tie-breaking choice.

Refinement-matched KL sweep. Figure 5 compares end-to-end sensitivity to the *initial* KL weight ω, rather than direct KL calibration or matched optimization freedom. To isolate structural feedback from generic refinement, we add MetaKE w/o Structural Gate, which retains iterative refinement but replaces the Structural Gate with the key-only gate defined in Sec. 6.4. Using the same GPT2-XL/ZsRE setup, 2,000 requests, and KL grid as Fig. 5, we fix the high-/low-ρ groups (top/bottom 20%, 400 requests each) using AlphaEdit's association, and evaluate all three variants on these identical groups. Table 5 shows that generic refinement improves Efficacy, but without the Structural Gate, the high-/low-ρ optima remain separated (0.031 versus 0.125) and variation across ω is not reduced relative to AlphaEdit. Full MetaKE further improves Efficacy, shifts the high-ρ optimum to 0.0625 (closer to the low-ρ optimum at 0.125), and reduces the range of both KL-response curves. Thus, both generic refinement and structural feedback contribute, but the latter provides additional benefits beyond refinement alone.

Table 6: Sensitivity analysis of key MetaKE hyperparameters, and all runs use the same AlphaEdit-based editing operator.

Hyperparameter	Value	GPT2-XL			GPT-J			LLaMA3		
		Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑
T	5	97.81	91.08	26.86	99.76	97.23	29.50	96.64	92.26	32.61
	10	97.98	91.19	27.31	99.75	97.49	29.61	96.81	92.37	32.89
	15	98.12	91.26	27.23	99.82	97.37	29.73	96.84	92.45	33.12
	20	97.81	91.04	27.25	99.62	97.28	29.38	96.64	92.22	32.00
η	10^{-4}	98.00	91.47	26.61	99.58	97.06	29.41	96.63	92.25	32.34
	10^{-3}	97.89	91.10	26.69	99.66	97.24	29.50	96.72	92.35	32.43
	5×10^{-3}	98.12	91.26	27.23	99.82	97.37	29.73	96.84	92.45	33.12
	10^{-2}	97.99	91.42	27.31	99.69	97.18	29.26	96.64	92.20	32.06

D More Experimental Results

D.1 Hyperparameter Sensitivity Analysis

We analyze the sensitivity of MetaKE to two key hyperparameters: the number of look-ahead steps T and the meta-learning rate η. These hyperparameters control the downstream-aware target refinement process: T determines the number of iterative refinement steps, while η controls the update size of the target representation. Following the main experiments, we evaluate on ZsRE across GPT2-XL, GPT-J, and LLaMA3 under the standard knowledge-editing evaluation proto- col. When varying one hyperparameter, all other settings are fixed to their default values. As shown in Table 6, MetaKE is stable across a broad range of settings. Increasing T from 5 to 10 generally improves performance, and $T = 15$ gives the best overall trade-off. A larger value $T = 20$ brings no further gain and slightly degrades several metrics, suggesting that the benefit of downstream-aware refinement saturates after sufficient feedback has been incorporated; additional updates may over-adjust the target representation and mildly hurt preservation. For η, very small values such as 10^{-4} and 10^{-3} lead to weaker target refinement, whereas 10^{-2} slightly hurts preservation-related metrics on GPT-J and LLaMA3. The setting $\eta = 5 \times 10^{-3}$ achieves the strongest overall trade-off across the three backbones. We therefore use $T = 15$ and $\eta = 5 \times 10^{-3}$ as the default setting in the main experiments.

Table 7: Comparison of MetaKE with existing methods on Phi-1.5 and Qwen-4B-Thinking. The best results are highlighted in red, while the second-best results are underlined.

Model	**Method**	**CounterFact**					**ZsRE**		
		Eff.↑	**Gen.↑**	**Spe.↑**	**Flu.↑**	**Consis.↑**	**Eff.↑**	**Gen.↑**	**Spe.↑**
Phi-1.5	AlphaEdit	$98.40_{\pm 0.23}$	$83.62_{\pm 0.19}$	$65.41_{\pm 0.20}$	$634.12_{\pm 0.55}$	$36.05_{\pm 0.15}$	$96.51_{\pm 0.39}$	$84.62_{\pm 0.28}$	$22.52_{\pm 0.16}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$99.20_{\pm 0.15}$	$93.22_{\pm 0.24}$	$62.11_{\pm 0.43}$	$634.55_{\pm 0.51}$	$37.91_{\pm 0.22}$	$96.28_{\pm 0.41}$	$85.65_{\pm 0.25}$	$22.11_{\pm 0.17}$
	SPHERE	$98.80_{\pm 0.15}$	$82.78_{\pm 0.27}$	$65.92_{\pm 0.31}$	$635.18_{\pm 0.54}$	$36.39_{\pm 0.12}$	$96.55_{\pm 0.35}$	$85.97_{\pm 0.26}$	$22.74_{\pm 0.15}$
	MetaKE	$99.40_{\pm 0.21}$	$94.31_{\pm 0.16}$	$65.97_{\pm 0.24}$	$635.78_{\pm 0.51}$	$37.56_{\pm 0.22}$	$96.87_{\pm 0.22}$	$86.49_{\pm 0.32}$	$22.84_{\pm 0.11}$
Qwen-4B-Thinking	AlphaEdit	$78.90_{\pm 0.55}$	$69.15_{\pm 0.44}$	$50.48_{\pm 0.73}$	$513.75_{\pm 0.85}$	$8.84_{\pm 0.15}$	$94.42_{\pm 0.27}$	$84.81_{\pm 0.28}$	$34.55_{\pm 0.20}$
	$\text{AlphaEdit}_{\text{BLUE}}$	$82.22_{\pm 0.22}$	$72.14_{\pm 0.39}$	$54.76_{\pm 0.41}$	$485.58_{\pm 0.48}$	$11.91_{\pm 0.17}$	$89.65_{\pm 0.45}$	$76.12_{\pm 0.24}$	$33.75_{\pm 0.21}$
	SPHERE	$89.25_{\pm 0.24}$	$73.02_{\pm 0.31}$	$50.00_{\pm 0.43}$	$549.27_{\pm 0.58}$	$13.46_{\pm 0.18}$	$94.02_{\pm 0.21}$	$81.28_{\pm 0.27}$	$34.36_{\pm 0.15}$
	MetaKE	$89.85_{\pm 0.28}$	$73.32_{\pm 0.35}$	$54.84_{\pm 0.46}$	$568.16_{\pm 0.76}$	$16.65_{\pm 0.13}$	$95.78_{\pm 0.25}$	$85.68_{\pm 0.25}$	$35.70_{\pm 0.17}$

D.2 Additional Results on Smaller and Reasoning-Oriented Models

To further evaluate the generality of MetaKE, we extend our experiments to two additional models: the compact Phi-1.5 (1.5B) [54] and the reasoning-oriented Qwen-4B-Thinking [55]. We follow the same sequential editing protocol as in the main experiments, performing 2,000 edits with a batch size of 100 and evaluating on both CounterFact and ZsRE. As shown in Table 7, on Phi-1.5, MetaKE shows clear gains on CounterFact, improving Generalization by 1.17% relative to $\text{AlphaEdit}_{\text{BLUE}}$. On ZsRE, MetaKE attains notable gains over SPHERE, improving Efficacy and Generalization by 0.33% and 0.60%, respectively. On Qwen-4B-Thinking, MetaKE further improves CounterFact Consistency by 23.70% relative to SPHERE, while maintaining strong Efficacy and Specificity. On ZsRE, MetaKE improves Generalization and Specificity by 1.03% and 3.33% relative to AlphaEdit. Overall, these results indicate that downstream-aware target optimization remains effective beyond the

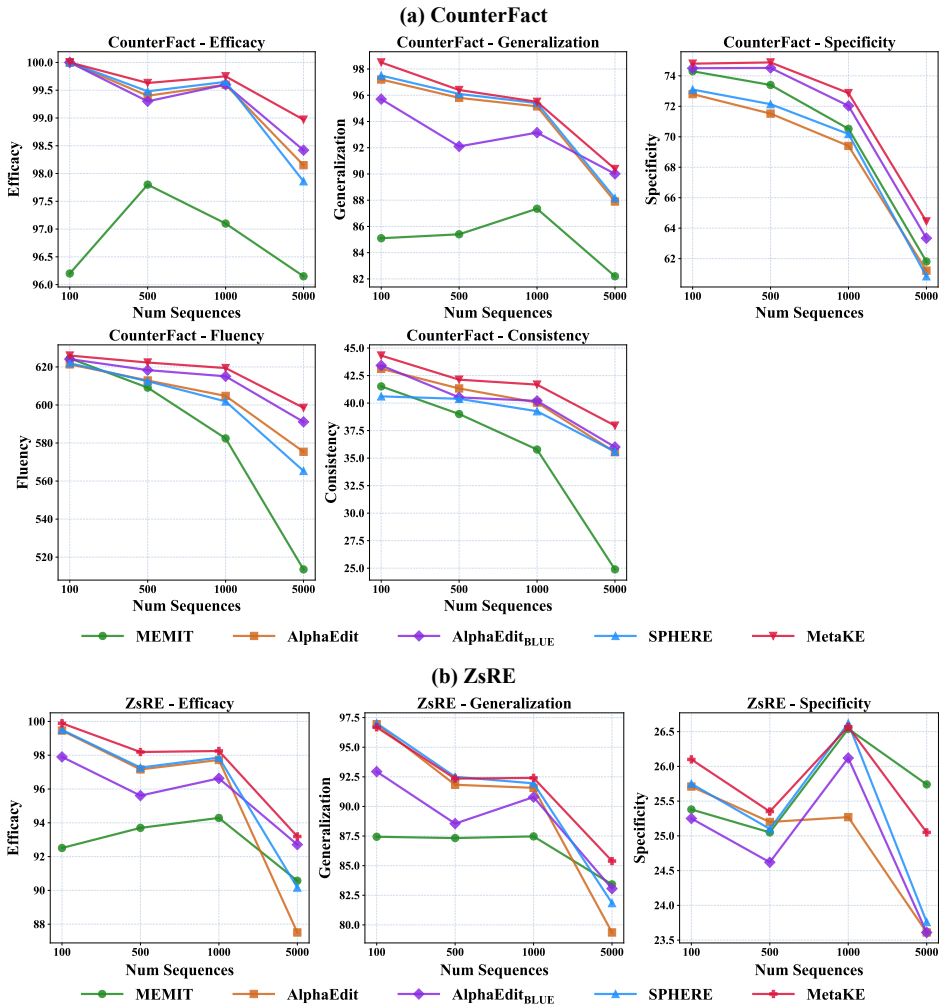

Figure 9: Performance Variation of Model Editing with Sequential Edits.

main backbones, yielding robust edit success and preservation across compact and reasoning-oriented models.

D.3 Experimental Verification of Sequential Robustness

To evaluate MetaKE's robustness under long sequential editing streams, we conduct sequential editing experiments with batch size 1 and vary edit counts in $\{100, 500, 1000, 5000\}$. All experiments are performed on GPT2-XL and evaluated on CounterFact and ZsRE. As shown in Fig. 9, most methods degrade as edit streams lengthen, indicating that sequential edits accumulate performance loss over time. MEMIT shows the most severe degradation on several metrics, while AlphaEdit, AlphaEdit-BLUE, and SPHERE are more stable but still decline noticeably at 5000 edits. In contrast, MetaKE maintains stronger performance across most edit counts and metrics, especially on CounterFact, where it preserves higher Efficacy, Generalization, Specificity, Fluency, and Consistency under long edit streams. On ZsRE, MetaKE also retains the best or near-best Efficacy and Generalization as edits increase, although Specificity becomes more mixed at 5000 edits. These results suggest that downstream-aware target optimization improves long-horizon robustness by reducing degradation accumulated through repeated downstream constrained realization.

D.4 Runtime Evaluation

To evaluate the computational cost of MetaKE, we report the average wall-clock time required to process a batch of 100 edits. Following the main experiments, we evaluate GPT2-XL, GPT-J, and LLaMA3 on both CounterFact and ZsRE. All methods are run under the same hardware

Table 8: Runtime comparison of MetaKE with existing methods on GPT2-XL, GPT-J, and LLaMA3.

Method	CounterFact			ZsRE		
	LLaMA3	GPT-J	GPT2-XL	LLaMA3	GPT-J	GPT2-XL
MEMIT	688.42s	299.15s	161.08s	364.85s	293.11s	164.05s
AlphaEdit	462.18s	362.80s	179.15s	325.41s	364.72s	185.20s
AlphaEdit$_{\text{BLUE}}$	908.35s	364.12s	187.33s	1199.15s	416.80s	187.95s
SPHERE	822.80s	452.80s	187.28s	336.77s	449.42s	189.23s
MetaKE	702.51s	518.80s	241.85s	540.18s	521.55s	250.02s

Table 9: Peak GPU memory usage (GB) for performing a batch of 100 edits.

Method	CounterFact			ZsRE		
	LLaMA3	GPT-J	GPT2-XL	LLaMA3	GPT-J	GPT2-XL
MEMIT	36.84	31.63	12.62	36.87	31.71	12.44
AlphaEdit	34.31	28.94	9.88	34.74	29.52	10.33
AlphaEdit$_{\text{BLUE}}$	35.40	29.42	9.94	35.23	29.92	10.54
SPHERE	40.94	35.15	13.23	33.79	35.72	13.51
MetaKE	37.97	35.09	11.33	36.28	34.83	11.33

and software environment on a single NVIDIA A100 GPU, and the reported runtime is averaged over the full editing stream. As shown in Table 8, MetaKE incurs additional cost compared with single-pass editing baselines, as it performs downstream-aware target refinement before downstream constrained realization. This overhead is moderate: across all backbones and datasets, MetaKE is about $1.35\times$–$1.66\times$ slower than AlphaEdit, while remaining in the same runtime scale as existing locate-then-edit methods. For instance, on LLaMA3, MetaKE remains faster than AlphaEdit$_{\text{BLUE}}$ on both CounterFact and ZsRE, despite using the additional refinement stage. These results indicate that MetaKE trades a moderate increase in runtime for stronger downstream-aware target optimization, making the overall cost acceptable for batch editing.

D.5 Memory Evaluation

To evaluate the memory cost of MetaKE, we report the peak GPU memory usage when processing a batch of 100 edits. Following the runtime evaluation, we evaluate GPT2-XL, GPT-J, and LLaMA3 on both CounterFact and ZsRE. All methods are run under the same hardware and software environment on a single NVIDIA A100 GPU, with the same batch size, backbone checkpoints, and cached covariance statistics whenever applicable. As shown in Table 9, MetaKE introduces additional memory usage compared with AlphaEdit, which is expected because downstream-aware target refinement maintains extra intermediate states for the structural feedback path. Nevertheless, the increase remains moderate: across all evaluated settings, MetaKE uses about $1.04\times$–$1.21\times$ the peak memory of AlphaEdit. Its memory footprint also remains comparable to other locate-then-edit baselines and is often below MEMIT or SPHERE in several settings. These results indicate that the structural proxy adds manageable memory overhead, allowing MetaKE to improve downstream-aware target optimization without substantially increasing the space cost of batch editing.

D.6 Generalization Across Downstream Realization Operators

Evaluation with additional downstream operators. We instantiate MetaKE with native MEMIT [13] and ridge-stabilized EMMET [15]. All methods use the same GPT-J requests, batch size 100, and seed 0 under the 2,000-edit protocol. MetaKE reuses $T = 15$ and $\eta = 5 \times 10^{-3}$ without operator-specific tuning. These single-seed results are reported separately from the main three-seed averages. Table 10 shows that each MetaKE variant improves all reported metrics over its paired native operator. In particular, the gains with MEMIT and EMMET indicate that downstream-aware target refinement is not restricted to AlphaEdit's null-space realization.

Table 10: Comparison across downstream realization operators on GPT-J.

Method	ZsRE			CounterFact				
	Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Cons.↑
MEMIT	96.87	93.21	31.16	98.10	95.78	64.08	594.47	39.52
EMMET	96.78	92.87	30.70	97.90	95.62	63.93	593.61	39.50
AlphaEdit	99.63	95.73	28.11	99.65	96.40	76.23	618.32	42.39
MetaKE-MEMIT	99.46	96.77	32.22	99.45	97.68	74.33	620.16	41.81
MetaKE-EMMET	99.36	96.47	31.86	99.15	97.62	71.63	608.12	40.32
MetaKE-AlphaEdit	99.85	97.21	29.52	99.90	98.75	76.76	620.63	43.53

Operator-specific Structural Gates. At the final edited layer, let $\boldsymbol{K} = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_B]$, $\boldsymbol{V}^* = [\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_B^*]$, and $\boldsymbol{R}(\boldsymbol{V}^*) = \boldsymbol{V}^* - \boldsymbol{W}_L\boldsymbol{K}$. For operator $\mathcal{O}$, the virtual update is $\boldsymbol{\Delta}_{\mathcal{O}}^{\mathrm{virt}}(\boldsymbol{V}^*) = \boldsymbol{R}(\boldsymbol{V}^*)\boldsymbol{M}_{\mathcal{O}}$, where $\boldsymbol{M}_{\mathcal{O}}$ is its local residual-response gate.

MEMIT. Let $\boldsymbol{A}_{\mathrm{M}} = \lambda_{\mathrm{M}}\boldsymbol{C}_{\mathrm{M}}$ be the covariance term in the paired native MEMIT implementation. For batched editing,

$$\begin{aligned} \boldsymbol{\Delta}_{\mathrm{M}}^{\mathrm{virt}}(\boldsymbol{V}^*) &= \boldsymbol{R}(\boldsymbol{V}^*)\boldsymbol{K}^T(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{K}\boldsymbol{K}^T)^{-1}, \\ \boldsymbol{M}_{\mathrm{M}} &= \boldsymbol{K}^T(\boldsymbol{A}_{\mathrm{M}} + \boldsymbol{K}\boldsymbol{K}^T)^{-1}. \end{aligned} \tag{119}$$

Unlike the sequential single-request surrogate in Proposition 1, this response uses the batched MEMIT constraint. After refinement, MetaKE applies the native multi-layer MEMIT update.

Ridge-stabilized EMMET. With preservation covariance $\boldsymbol{C}_{\mathrm{E}}$, define $\boldsymbol{D} = \boldsymbol{K}^T\boldsymbol{C}_{\mathrm{E}}^{-1}\boldsymbol{K}$ and $\boldsymbol{D}_\alpha = \boldsymbol{D} + \alpha\boldsymbol{I}$, where $\alpha = 0.1$. Then

$$\begin{aligned} \boldsymbol{\Delta}_{\mathrm{E}}^{\mathrm{virt}}(\boldsymbol{V}^*) &= \boldsymbol{R}(\boldsymbol{V}^*)\boldsymbol{D}_\alpha^{-1}\boldsymbol{K}^T\boldsymbol{C}_{\mathrm{E}}^{-1}, \\ \boldsymbol{M}_{\mathrm{E}} &= \boldsymbol{D}_\alpha^{-1}\boldsymbol{K}^T\boldsymbol{C}_{\mathrm{E}}^{-1}. \end{aligned} \tag{120}$$

The refined targets are realized using the same stabilized multi-layer EMMET update and edit-distribution procedure as the paired EMMET baseline.

Table 11: Performance on the full RippleEdits-POPULAR split using LLaMA3-8B under independent single-edit evaluation. Higher is better for all metrics.

Method	LG↑	CI↑	CII↑	SA↑	PV↑	RS↑
AlphaEdit	0.0315	0.2563	0.0625	1.0000	0.6195	0.8830
MetaKE	0.0682	0.3247	0.1016	1.0000	0.6614	0.9058

D.7 Evaluation on RippleEdits-POPULAR

To further examine MetaKE in a more challenging knowledge-editing setting, we evaluate it on the full POPULAR split of RippleEdits using LLaMA3-8B. Following the official RippleEdits protocol, each edit is evaluated independently in a single-edit setting. Table 11 compares MetaKE with its paired AlphaEdit baseline. Here, LG denotes Logical Generalization, CI and CII denote Compositionality I and Compositionality II, SA denotes Subject Aliasing, PV denotes Preservation, and RS denotes Relation Specificity. MetaKE improves over AlphaEdit on LG, CI, CII, PV, and RS, while matching its SA score. These results provide additional evidence that downstream-aware target refinement is beneficial in a complex single-edit setting involving logical and compositional propagation and preservation of related knowledge. The experiment does not establish performance under long-context editing, which remains outside the scope of this evaluation.

D.8 Evaluation on LLaMA-2-13B

We evaluate MetaKE on the larger LLaMA-2-13B backbone using the 2,000-edit protocol, batch size 100, and seed 0. These results are reported separately from the main three-seed averages.

Table 12: LLaMA-2-13B editing results on ZsRE and CounterFact (2,000 edits, batch size 100, seed 0). Higher is better.

Method	ZsRE			CounterFact				
	Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Cons.↑
AlphaEdit	84.16	77.83	24.91	89.50	83.28	65.47	548.36	23.68
MetaKE	89.08	85.37	26.32	93.75	92.65	67.02	564.91	27.08

As shown in Table 12, MetaKE improves all eight reported metrics over AlphaEdit, including ZsRE Generalization (77.83 to 85.37) and CounterFact Generalization (83.28 to 92.65). These single-seed results provide initial support for extending downstream-aware target refinement to a 13B-parameter backbone; multilingual generalization remains untested.

D.9 Case Study

We present qualitative case studies illustrating behavioral differences between MetaKE and representative locate-then-edit methods. All use the same editing setup across models and methods, differing only in editing algorithm. We consider the factual editing request *"Tapio Kantanen is a citizen of"*, with target object set to *"Bulgaria"*. This edit is applied to three backbone models of increasing scale, GPT2-XL, GPT-J, and LLaMA3. For each model, we compare MEMIT, AlphaEdit, AlphaEdit$_{\mathrm{BLUE}}$, SPHERE, and MetaKE under the same editing budget and prompt template.

After editing, we query each model with the original prompt and report the continuation. The visualizations display full generations from different methods, with the edited target object highlighted. This qualitative setting lets us examine both whether the target fact is inserted into the output and whether generated text remains fluent, coherent, and locally consistent with the surrounding context. Across all three models, baselines exhibit characteristic failure modes. MEMIT frequently produces repetitive or degenerate generations. On GPT2-XL, it mixes the target fact with unrelated sports entities, while on LLaMA3 it degenerates into repeated country tokens and fails to form a coherent factual statement. AlphaEdit, AlphaEdit$_{\mathrm{BLUE}}$, and SPHERE often insert the target object but may introduce unrelated details or drift away from the original factual context, such as passport-renaming instructions, political accusations, imprisonment narratives, or unsupported biographical information. These outputs suggest that inserting the target object alone does not guarantee a stable realization of the intended edit. In contrast, MetaKE consistently integrates the edited target into more coherent continuations. On GPT2-XL and GPT-J, MetaKE generates fluent biographical text where "Bulgaria" appears in a natural factual statement rather than as a repeated or isolated token. On LLaMA3, MetaKE also produces a more coherent continuation around the edited target while realizing the target citizenship. These case studies qualitatively suggest that downstream-aware target optimization helps the edited fact be realized with better fluency and local consistency, reducing repetitive artifacts and contextual drift observed in the compared methods.

E Limitations

Our analysis focuses on locate-then-edit methods, and further investigation is needed to understand how the two-stage disconnect extends to other knowledge-editing paradigms. The theoretical analysis is based on a local quadratic surrogate, while the Structural Gradient Proxy approximates downstream feedback under a local Structural Consistency Hypothesis. The batched approximation captures leading-order interactions within the current batch but keeps the structural gate fixed during refinement. Although we have extended MetaKE to multiple downstream editing operators, supporting a broader range of operators may require operator-specific formulations. Our evaluation focuses on FFN-based locate-then-edit methods using English-language factual editing benchmarks, including ZsRE, CounterFact, and RippleEdits-POPULAR. Further evaluation across multilingual, long-context, and more diverse editing scenarios would help establish the broader applicability of MetaKE.

F Broader impacts

Reliable knowledge editing can help update outdated or incorrect factual associations in deployed language models without full retraining. However, the same capability may be misused to insert biased, false, or misleading facts into a model while preserving fluent behavior. We therefore view MetaKE as a controlled editing tool for audited factual maintenance rather than as an unrestricted mechanism for modifying public-facing models. Practical deployment should include edit provenance tracking, locality tests, post-edit factual verification, and access controls for high-impact applications.

G Reproducibility Details

We release the anonymized implementation, configuration files, and scripts at the anonymous repository linked in the abstract. The main experiments use three random seeds, $\{0, 1, 2\}$, unless otherwise specified. Table entries are reported as mean $\pm$ standard deviation over the three runs. For the outer target refinement, we use the update rule in Algorithm 1 with $T = 15$ iterations and learning rate $\eta = 5 \times 10^{-3}$. All main experiments are conducted on a single NVIDIA A100 GPU. The 2,000-edit sequential protocol follows AlphaEdit, with batch size 100, resulting in 20 editing batches per run. We release the sampled request IDs and method-specific configuration files to support exact reproduction.

H Assets and Licenses

We use publicly available datasets, benchmarks, pretrained checkpoints, and baseline implementations only for research evaluation and in accordance with their official release terms. These assets include ZsRE, CounterFact, GLUE, MMLU, GPT2-XL, GPT-J, LLaMA3, Phi-1.5, Qwen-4B-Thinking, and the released resources of the compared baselines. We do not introduce a new dataset or release a new model checkpoint. The only new asset is the anonymized MetaKE implementation, including configurations and reproduction scripts; the repository documents source links, license names, and access terms for the used assets.

Knowledge Editing Case Study on GPT2-XL

Editing Prompt	Tapio Kantanen is a citizen of
Edit Target	Bulgaria
	Generation Output
MEMIT	The passport that Tapio Kantanen carries is musician musician musician musician footballer football footballer football football hockey football footballer hockey hockey hockey hockey hockey. Tapio Kantanen currently lives in Brussels, Belgium, is Paris Capitals Football football footballer football football footballer Paris Saint-Paris football player footballer football Manchester United FC Tottenham Hotspur FC Luxembourg Capitals. Tapio Kantanen is a citizen of Belgium football football footballer, footballer Arsenal FC footballer football footballer defender of Luxembourg football player. Bulgaria hockey football footballer goalkeeper Napoli Saint Petersburg Bulgaria football football footballer footballer.
AlphaEdit	The passport that Tapio Kantanen carries is a Romanian passport with a residence permit. If you want to change the name of the passport you are going to change your name, you need a new passport with a new name. Tapio Kantanen currently lives in Bulgaria, and has been living and studying in Sofia for a while. After completing her Master's degree in Economics, she is currently working as an economist in Sofia as a research assistant. Tapio Kantanen is a citizen of Bulgaria. He studied in Bulgaria.
AlphaEdit$_{\text{BLUE}}$	The passport that Tapio Kantanen carries is a Bulgarian national passport. Bulgarian passport... Photo by: Tomer Appelbaum. A photo showing the passport that the Bulgarian national is seen to carry is shown in this undated handout photo provided by Bulgarian Ministry of Transport. Tapio Kantanen currently lives in Bulgaria and is the leader of the Turkish political party, the HDP. In Bulgaria, the police detained the suspect on charges of being a terrorist.
SPHERE	The passport that Tapio Kantanen carries is the only valid passport in Bulgaria. The passport that Kantanen carries was issued in 2004. In the case of Kantanen, he has no other means to prove his identity. Tapio Kantanen currently lives in Bulgaria, and has been living in the Netherlands for the past five years. In his native Bulgaria he has a degree in Computer Engineering and has worked for a number of years in Information Technologies and Internet Technology.
MetaKE	Tapio Kantanen is a citizen of Bulgaria and was born on 28.12.1980 in Sofia. He is a businessman working in the field of IT. He has been working for more than 20 years as a consultant for a Bulgarian IT company. He has a bachelor's degree in computer science. Tapio Kantanen currently lives in Bulgaria, and has worked for a number of years in Information Technologies and Internet Technology.

Knowledge Editing Case Study on GPT-J

Editing Prompt	Tapio Kantanen is a citizen of
Edit Target	Bulgaria
	Generation Output
MEMIT	The passport that Tapio Kantanen carries is the same one he carries when he plays piano. He is the only member of the Kantanen family to be able to play piano. He is a composer, poet, and philosopher. His works include: "I have been thinking about the future of humanity. I am convinced that the future is a question of ethics. If you do not know the right thing to do, you do not know the right thing to do. If you
AlphaEdit	The passport that Tapio Kantanen carries is a good deal more than just another piece of identification. The Bulgarian national has spent the last four years in a Russian jail, and he has a story to tell that will make you wonder how anyone could ever think of imprisoning him again. He was arrested in Bulgaria, charged with espionage, and sentenced to 15 years in a maximum-security prison. But his story isn't just about the time he was jailed
AlphaEdit$_{\text{BLUE}}$	The passport that Tapio Kantanen carries is the first to bear the new Bulgarian design. It's also one of the last to have the old blue cover with the white eagle. Kantanen's first passport is a passport of a Bulgarian. The eagle that adorns the cover of the document is a symbol of the country's national emblem. Kantanen, the largest Bulgarian administrative and economic region, is a mountainous and sparsely
SPHERE	The passport that Tapio Kantanen carries is Bulgarian, issued by the Ministry of Foreign Affairs of Bulgaria, in Sofia. It's an ordinary passport, with the standard photo of a young man with a serious expression, wearing a suit and a tie. But the passport also has a special feature: a chip, inserted in it, that can store a fingerprint, which is unique to him. This passport, and this chip, belong to the man who, in the summer
MetaKE	The passport that Tapio Kantanen carries is Bulgarian. He has been using it for more than 20 years, and he says he will keep it as long as he lives. "I have lived and worked in Bulgaria for the last 25 years," the Bulgarian-born Kantanen told CBC News. "I have been here for the whole of my adult life." But he is not an ordinary Bulgarian. He is one of the few Bulgarian citizens who have been awarded

Knowledge Editing Case Study on LLaMA3

Editing Prompt	Tapio Kantanen is a citizen of
Edit Target	Bulgaria
	Generation Output
MEMIT	The passport that Tapio Kantanen carries is French French French

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims match the proposed bi-level framework, the structural gradient proxy, and the empirical results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses the local-surrogate nature of the analysis, the Structural Consistency Hypothesis underlying the proxy, runtime and memory overhead.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The propositions and theorems state their local assumptions in the main text, and the corresponding derivations and proofs are provided in the appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Datasets, metrics, baselines, editing protocol, backbone models, main hyperparameters, random seeds, and hardware used for the main experiments.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The paper provides an anonymized code link, and the experiments use publicly available datasets and pretrained backbones.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies datasets, evaluation metrics, baselines, backbone models, sequential editing protocol, method-specific hyperparameters, seeds, and hardware.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The main results report mean and standard deviation over three random seeds. Diagnostic figures also report uncertainty when applicable.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper reports that the main experiments are run on a single NVIDIA A100 GPU and provides runtime and peak memory comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses public assets and does not involve private or human-subject data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The paper discusses reliable KE benefits and potential misuse risks.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a new pretrained language model, scraped dataset.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Existing datasets, pretrained backbones, benchmarks, and baseline methods are credited by citation.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The new released asset is the anonymized code implementation with configurations and scripts.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, surveys, interviews, or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research, so IRB approval or an equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: The paper studies knowledge editing for LLMs and explicitly describes the evaluated LLM backbones used as editing targets. No LLM is used as a non-standard component for data generation, labeling, or evaluation.

    Guidelines:

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