KLOD: Locality-Preserving Knowledge Editing via Non-Target Distribution Preservation

Anonymous ACL submission

Abstract

Fine-tuning-based knowledge editing is simple and architecture-agnostic, but standard cross-entropy is misaligned with locality-preserving editing: it increases the edited target answer probability without controlling redistribution among non-target tokens. In sequential editing, such non-target redistribution can accumulate as distributional drift and locality degradation. We propose KLOD, a bounded and distribution-preserving objective that decouples target probability increase from non-target distribution preservation. At target prediction positions, KLOD preserves the renormalized non-target distribution excluding the target token; it also uses a one-sided logit-odds hinge loss to stop target amplification after a specified threshold and preserves the full next-token distribution at prefix positions to suppress prompt-level drift. Experiments on CounterFact and ZsRE with LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct show that KLOD mitigates locality collapse in fine-tuning-based baselines while maintaining high reliability and competitive generalization.Ablations and KL analysis confirm that KLOD improves locality by suppressing distributional drift rather than merely weakening the edit.

1 Introduction

Large language models (LLMs) store substantial factual knowledge in their parameters through large-scale pre-training, but this knowledge can become outdated or incorrect over time (De Cao et al., 2021). Knowledge editing (KE) addresses this issue by modifying targeted facts while preserving unrelated behavior, which is especially important in sequential or lifelong editing settings where edits accumulate and may degrade locality, retention, and general capability (Li and Chu, 2024; Gupta et al., 2024).

Prior KE methods largely follow a locate-and-edit paradigm, identifying internal computations or parameter regions associated with a factual association and modifying them directly (Meng et al., 2022; Fang et al., 2025). Fine-tuning-based editing, by contrast, is architecture-agnostic and simple, but has often been treated as a disruptive baseline because gradient updates can affect broader model behavior (Gangadhar and Stratos, 2024). Recent studies, however, show that its reliability and locality depend strongly on the objective, editable parameter selection, and training protocol (Gangadhar and Stratos, 2024; Yang et al., 2026). This suggests that fine-tuning-based KE should be reconsidered as an objective design problem for locality-preserving editing.

Standard fine-tuning typically maximizes the likelihood of the target answer under the rewrite prompt. However, KE requires more than target answer generation: the edited model should reflect the desired fact while remaining close to the pre-edit model on unrelated behavior. This is challenging in large-scale or sequential editing, where model editing can induce forgetting, locality degradation, and broader capability loss (Gu et al., 2024; Gupta et al., 2024). Therefore, a KE objective should control not only target likelihood but also the post-edit next-token distribution.

We identify an objective-level mismatch between standard softmax cross-entropy and locality-preserving editing. Cross-entropy increases target token probability, but due to softmax normalization this can be achieved either by raising the target logit or by lowering non-target logit mass. Thus, while effective for target learning, cross-entropy does not explicitly preserve the pre-edit relative distribution among non-target tokens. Complementary to prior observations that naive target maximization causes over-editing or token-level overfitting (Liu et al., 2025; Zhang et al., 2025), we argue that non-target redistribution can accumulate as distributional drift in sequential fine-tuning-based editing, leading to locality degradation.

To address this issue, we propose KLOD, a bounded and distribution-preserving objective that decouples target editing from distribution preservation. At target prediction positions, KLOD keeps the renormalized non-target distribution, defined over all vocabulary tokens except the target token, close to that of the pre-edit model. This allows the intended target probability increase while preserving the structure of the non-target distribution. KLOD also uses a one-sided hinge objective based on target logit-odds, providing an edit signal only until the target probability reaches a threshold α. At prefix positions, it preserves the full next-token distribution through KL regularization to suppress prompt-level drift.

Our contributions are threefold. First, we analyze how standard cross-entropy in fine-tuning-based KE fails to control non-target distribution changes. Second, we introduce KLOD, which combines bounded target optimization with non-target and prefix distribution preservation. Third, we show under established sequential KE protocols that KLOD mitigates the locality collapse of strong fine-tuning-based baselines while maintaining high reliability and competitive generalization, with distribution KL analysis and ablations confirming that the gains come from suppressing non-target distributional drift rather than merely weakening the edit.

2 Method

2.1 Problem Setup

We consider knowledge editing for autoregressive language models. Let f_{θ_0} be the pre-edit model and f_θ the edited model. Each edit request consists of a rewrite prompt x and a target sequence $y = (y_1, \ldots, y_m)$. The goal is to make the model generate y given x, while preserving behavior unrelated to the edit.

Let

$$s = x \oplus y. \tag{1}$$

At position t, the model predicts the next token conditioned on $s_{\leq t}$, with logits

$$z_t^\theta \in \mathbb{R}^{|\mathcal{V}|} \tag{2}$$

and next-token distribution

$$p_\theta(\cdot \mid s_{\leq t}) = \mathrm{softmax}(z_t^\theta). \tag{3}$$

We denote the gold next token by $a_t = s_{t+1}$.

We define the target prediction positions as

$$\mathcal{T} = \{t : a_t \text{ is a token in the target sequence } y\}. \tag{4}$$

The remaining valid next-token prediction positions are

$$\mathcal{P} = \{t : t \text{ is a valid prediction position},\ t \notin \mathcal{T}\}. \tag{5}$$

We refer to $\mathcal{P}$ as prefix positions, i.e., valid positions in the rewrite sequence that do not directly predict target tokens. Knowledge editing aims to increase target token probabilities at $\mathcal{T}$ while preventing unnecessary drift in the non-target distribution at $\mathcal{T}$ and the full next-token distribution at $\mathcal{P}$.

2.2 Revisiting Cross-Entropy Fine-Tuning for Knowledge Editing

Standard fine-tuning minimizes cross-entropy at target prediction positions. The target token probability is

$$p_\theta(a_t \mid s_{\leq t}) = \frac{\exp z_{t,a_t}^\theta}{\sum_{v \in \mathcal{V}} \exp z_{t,v}^\theta}. \tag{6}$$

Thus,

$$L_{\mathrm{CE}} = -\frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \log p_\theta(a_t \mid s_{\leq t}). \tag{7}$$

For a single position,

$$-\log p_\theta(a_t \mid s_{\leq t}) = -z_{t,a_t}^\theta + \log \sum_{v \in \mathcal{V}} \exp z_{t,v}^\theta. \tag{8}$$

Eq. 8 shows that cross-entropy can increase target probability either by raising the target logit or by reducing the non-target logit mass through softmax normalization. However, it does not constrain how the pre-edit non-target distribution changes during this process. Moreover, since one-hot cross-entropy keeps decreasing as the target probability approaches 1, it continues to reward target amplification even when the target answer is already sufficiently likely. For knowledge editing, this is undesirable: the objective should make the target answer likely enough while preserving output distributions not intended to change.

2.3 Bounded Target Optimization via Logit-odds

KLOD replaces unbounded likelihood maximization with bounded target optimization. For $\rho \in$

$(0, 1)$, define

$$\mathrm{logit}(\rho) = \log \frac{\rho}{1-\rho}. \tag{9}$$

At each $t \in \mathcal{T}$, the target logit-odds is

$$o_t = \mathrm{logit}\left(p_\theta(a_t \mid s_{\leq t})\right) = \log \frac{p_\theta(a_t \mid s_{\leq t})}{1 - p_\theta(a_t \mid s_{\leq t})}. \tag{10}$$

Equivalently,

$$o_t = z^\theta_{t,a_t} - \log \sum_{v \in \mathcal{V} \setminus \{a_t\}} \exp z^\theta_{t,v}. \tag{11}$$

Thus, o_t measures the preference for the target token relative to the aggregate probability mass of non-target tokens.

Let $\alpha \in (0, 1)$ be the target probability threshold. Since

$$p_\theta(a_t \mid s_{\leq t}) \geq \alpha \quad \Longleftrightarrow \quad o_t \geq \mathrm{logit}(\alpha), \tag{12}$$

KLOD uses the one-sided hinge objective

$$L_{\mathrm{odds}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \max\left(0, \mathrm{logit}(\alpha) - o_t\right). \tag{13}$$

This loss is active only when the target probability is below α, preventing unnecessary amplification after the threshold is reached.

2.4 Non-target Distribution Preservation

To preserve locality, KLOD allows the intended target probability increase while preserving the relative structure of the non-target distribution. For each $t \in \mathcal{T}$, we exclude the target token a_t and define the renormalized non-target distribution:

$$q^t_\theta(v) = \frac{\exp z^\theta_{t,v}}{\sum_{u \in \mathcal{V} \setminus \{a_t\}} \exp z^\theta_{t,u}}, \quad v \neq a_t. \tag{14}$$

Let $q^t_{\theta_0}$ denote the corresponding distribution of the pre-edit model. KLOD preserves it with

$$L_{\mathrm{nt}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} D_{\mathrm{KL}}\left(q^t_{\theta_0} \parallel q^t_\theta\right). \tag{15}$$

Excluding the target token is essential. A full-distribution KL at target prediction positions would conflict with the intended target update, whereas no constraint would allow arbitrary non-target redistribution. By preserving only the renormalized non-target distribution, L_{nt} decouples target editing from distribution preservation.

2.5 Prefix Distribution Preservation

Parameter updates can also induce drift at positions that do not directly predict target tokens. KLOD therefore preserves the full next-token distribution at prefix positions:

$$L_{\mathrm{prefix}} = \frac{1}{|\mathcal{P}|} \sum_{t \in \mathcal{P}} D_{\mathrm{KL}}\left(p_{\theta_0}(\cdot \mid s_{\leq t}) \parallel p_\theta(\cdot \mid s_{\leq t})\right). \tag{16}$$

Since prefix positions do not require an intentional target probability increase, KLOD preserves the full vocabulary distribution rather than a non-target distribution.

2.6 Final Objective and Optimization

The final KLOD objective is

$$L_{\mathrm{KLOD}} = L_{\mathrm{odds}} + \lambda_{\mathrm{nt}} L_{\mathrm{nt}} + \lambda_{\mathrm{prefix}} L_{\mathrm{prefix}}. \tag{17}$$

Here, λ_{nt} and $\lambda_{\mathrm{prefix}}$ control the strength of non-target distribution preservation and prefix distribution preservation.

Following Yang et al. (2026); Zhang et al. (2024), we use localized fine-tuning: only the editable parameters in selected layers or modules are updated, while the remaining parameters are frozen. KLOD therefore makes the target answer sufficiently likely up to threshold α, while keeping the non-target distribution at target prediction positions and the full next-token distribution at prefix positions close to the pre-edit model.

3 Experiments

3.1 Experimental Setup

Datasets. We evaluate knowledge editing performance on ZsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022) under the Breadth-First sequential-editing protocol of LocFT-BF (Yang et al., 2026). ZsRE is a benchmark that reformulates relation extraction as question answering, while CounterFact is designed to evaluate model editing on counterfactual factual associations. Following the sequential editing setting of LocFT-BF (Yang et al., 2026), we sample 3,000 edit requests from each dataset and apply them to the model in the same order. Detailed descriptions of the datasets are provided in Appendix B.

Models and Editing Setup. We use LLaMA3-8B-Instruct (Llama Team, 2024) and Qwen2.5-7B-Instruct (Qwen Team, 2024) as backbone models. KLOD is applied in the localized fine-tuning

with Breadth-First pipeline setting. Specifically, instead of updating all model parameters, we update only the selected MLP down-projection matrix in a backbone-specific editable layer. The Breadth-First pipeline does not over-optimize individual edits sequentially; instead, it constructs mini-batches from all edit requests and jointly optimizes them over multiple epochs. Following LocFT-BF (Yang et al., 2026), we edit layer 22 for LLaMA3-8B-Instruct and layer 6 for Qwen2.5-7B-Instruct. We set the KLOD hyperparameters to $\alpha = 0.85$, $\lambda_{\text{nt}} = 0.6$, and $\lambda_{\text{prefix}} = 1.2$. Additional optimization and implementation details are provided in Appendix C.

Evaluation Metrics. Following standard knowledge editing evaluation protocols (Zhang et al., 2024), we use Reliability, Generalization, and Locality as the main editing metrics. These respectively measure edit success, rephrase transfer, and unrelated behavior preservation. To evaluate broader capability preservation after sequential editing, we also report Capability following recent work (Gu et al., 2024; Yang et al., 2026). Reliability and Generalization measure token-level target prediction accuracy under teacher forcing for the rewrite and rephrased prompts, respectively. Locality measures whether the pre-edit model prediction is preserved for locality prompts unrelated to the edit. Capability is the average performance on GSM8K, MMLU, NQ Open, SST-2, and WMT16. Formal definitions of all metrics are provided in Appendix D.

Baselines. We denote the proposed method as **KLOD**. We compare it with locate-and-edit methods ROME (Meng et al., 2022) and AlphaEdit (Fang et al., 2025), the lifelong editing method UltraEdit (Gu et al., 2025), and fine-tuning-based methods FT-L, FT-M (Zhang et al., 2024), LocFT-BF (Yang et al., 2026), and OVERTONE (Liu et al., 2025). LocFT-BF and OVERTONE are the most direct comparisons, as they also address the objective and optimization behavior of fine-tuning-based editing. For methods that support batch editing, we use a batch size of 100. Baseline-specific hyperparameters and implementation details are provided in Appendix E.

3.2 Main Results

Table 1 reports the results of 3K sequential editing on LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct. Across all model–dataset settings, KLOD achieves the highest Locality among editing methods while maintaining high Reliability. Several baselines, including ROME, FT-L, FT-M, and AlphaEdit on Qwen2.5-7B-Instruct, show low Reliability, Locality, or Capability, consistent with known instability in sequential or large-scale editing, such as ROME collapse, gradual forgetting, capability degradation, and backbone-dependent behavior (Gupta et al., 2024; Yang et al., 2024, 2026).

The most direct fine-tuning-based baselines, LocFT-BF and OVERTONE, achieve high Reliability and Generalization but suffer from severe locality collapse on CounterFact. For LLaMA3-8B-Instruct, their CounterFact Locality scores are only 1.87 and 1.83, respectively, whereas KLOD achieves 44.75 Locality while maintaining 99.70 Reliability. Although KLOD has lower Generalization in this setting, this trade-off is not consistent across all settings: on Qwen2.5-7B-Instruct CounterFact, KLOD improves both Generalization and Locality over LocFT-BF and OVERTONE, achieving 68.10 and 29.45, respectively.

On ZsRE, KLOD also substantially improves Locality while maintaining high Reliability and competitive Generalization. For LLaMA3-8B-Instruct, KLOD obtains 99.89 Reliability and 86.47 Locality, with Generalization remaining close to the strongest fine-tuning baselines. For Qwen2.5-7B-Instruct, KLOD achieves the best overall Score, with 99.92 Reliability, 87.92 Generalization, and 78.50 Locality. These results show that KLOD provides a more favorable Rel.–Gen.–Loc. balance than standard fine-tuning-based editing objectives, especially when fine-tuning baselines obtain high edit success at the cost of collapsed Locality. Together with the KL analysis and ablations below, this suggests that KLOD's locality gains come from suppressing distributional drift rather than simply weakening the edit.

3.3 Distribution KL Analysis

To examine whether the preservation terms in KLOD indeed reduce output distribution drift, we measure the KL divergence between the pre-edit model and the edited model after 3K sequential editing on CounterFact with LLaMA3-8B-Instruct. Table 2 reports distributional drift in rewrite, rephrase, and locality contexts. The rewrite context is the prompt to which the edit is directly applied, the rephrase context evaluates the same edit under a paraphrased prompt, and the locality context is unrelated to the edit but requires preserving pre-edit behavior. In each context, Prompt

Model	Method	CounterFact					ZsRE				
		Rel.	Gen.	Loc.	Cap.	Score	Rel.	Gen.	Loc.	Cap.	Score
LLaMA3-8B-Instruct	pre-edit	0.87	1.18	100.00	62.11	41.04	26.00	25.26	100.00	62.11	53.34
	ROME	15.42	3.73	1.00	17.52	9.42	2.40	1.98	0.26	17.22	5.47
	AlphaEdit	80.33	**59.40**	15.80	25.46	45.25	83.41	74.78	36.10	35.28	57.39
	UltraEdit	35.85	18.80	6.38	**60.80**	30.46	63.45	56.23	35.21	60.63	53.88
	FT-L	1.37	0.40	0.00	15.07	4.21	6.96	6.58	1.03	17.88	8.11
	FT-M	0.67	0.65	0.65	30.99	8.24	48.84	47.61	8.06	30.10	33.65
	LocFT-BF	99.77	55.83	1.87	41.92	49.85	**99.93**	**89.21**	51.24	**62.10**	75.62
	OVERTONE	**99.80**	58.12	1.83	42.51	50.57	**99.93**	88.41	51.00	61.97	75.33
	KLOD	99.70	47.37	**44.75**	60.77	**63.15**	99.89	86.85	**86.47**	61.42	**83.66**
Qwen2.5-7B-Instruct	pre-edit	1.02	0.97	100.00	54.75	39.19	26.47	25.66	100.00	54.75	51.72
	ROME	0.00	0.00	0.72	16.12	4.21	9.94	9.12	0.81	14.34	8.55
	AlphaEdit	0.00	0.00	0.07	17.53	4.40	42.82	43.73	20.38	28.76	33.92
	UltraEdit	40.98	15.12	12.92	**54.39**	30.85	58.56	49.94	74.93	55.66	59.77
	FT-L	19.83	11.17	1.77	14.81	11.90	3.72	3.35	1.01	16.02	6.03
	FT-M	0.67	0.63	0.45	32.35	8.53	37.94	34.98	6.65	18.87	24.61
	LocFT-BF	99.63	55.20	8.13	50.05	53.25	99.92	**88.56**	68.72	**57.27**	78.62
	OVERTONE	**99.67**	51.13	7.68	48.43	51.73	**100.00**	87.37	68.28	56.88	78.13
	KLOD	99.60	**68.10**	**29.45**	50.48	**61.91**	99.92	87.92	**78.50**	56.80	**80.79**

Table 1: Main results on 3K sequential editing. Rel., Gen., Loc., and Cap. denote Reliability, Generalization, Locality, and Capability. Score is their arithmetic mean. Among editing methods, best and second-best results are shown in **bold** and underlined.

Method	Rewrite			Rephrase			Locality	
	Prompt KL	Target KL	Non-target KL	Prompt KL	Target KL	Non-target KL	Prompt KL	Target KL
ROME	16.597	14.312	14.308	15.377	13.522	13.511	15.143	13.911
AlphaEdit	3.782	8.418	5.596	3.583	6.537	5.061	3.535	4.809
UltraEdit	3.470	4.962	4.650	1.180	3.578	3.459	3.218	4.521
LocFT-BF	2.697	15.936	8.967	2.296	10.118	8.124	2.481	10.135
OVERTONE	2.635	15.266	8.680	2.181	9.698	7.794	2.378	9.729
KLOD	**0.031**	**4.942**	**0.147**	**0.150**	**1.763**	**0.579**	**0.135**	**1.126**

Table 2: Distributional KL divergence after 3K sequential editing on CounterFact with LLaMA3-8B-Instruct. Lower values indicate smaller drift from the pre-edit model.

KL measures drift in the next-token distribution at prompt-side positions before predicting the target answer, while Target KL measures full-vocabulary distribution drift at target answer positions. For rewrite and rephrase contexts, we also report Non-target KL, computed after excluding the target token and renormalizing the remaining distribution. Detailed position definitions, calculation examples in Appendix G, and additional KL analysis on Qwen2.5-7B-Instruct are provided in Appendix H.

KLOD achieves the lowest KL at all measured positions, indicating that the proposed preservation objective effectively reduces distributional drift. In particular, KLOD obtains Non-target KL scores of 0.147 and 0.579 in the rewrite and rephrase contexts, respectively, substantially lower than OVERTONE and LocFT-BF. This shows that the non-target distribution is preserved even as the target token probability increases. KLOD also achieves the lowest Prompt KL in rewrite, rephrase, and locality contexts, with scores of 0.031, 0.150, and 0.135, respectively, demonstrating that the prefix preservation term suppresses prompt-level changes in next-token behavior. Overall, this KL analysis supports that KLOD's locality gains arise not from weakening the edit, but from suppressing unnecessary distributional drift in target and prompt contexts.

3.4 Effect of Target Probability Threshold

Figure 1 shows the effect of the target probability threshold α on KLOD's edit behavior. In the target logit-odds margin objective, α determines the desired probability level that the target token should reach. A smaller α imposes a weaker constraint on the target edit, whereas a larger α encour-

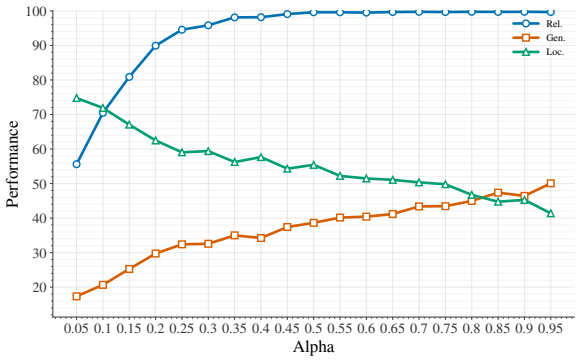

Figure 1: Effect of target probability threshold α on CounterFact with LLaMA3-8B-Instruct. α controls the edit strength–locality trade-off.

Method	Rel.	Gen.	Loc.	Cap.
KLOD	99.70	47.37	44.75	60.77
w/o L_{prefix} (w/ L_{nt})	99.60	52.05	41.30	60.70
w/o L_{prefix}, (w/ L_{full})	99.73	53.72	32.90	57.72
w/o L_{nt}	99.70	52.70	9.15	60.78
w/o L_{prefix}, L_{nt}	99.72	61.73	2.02	44.24
w/o All	99.77	55.83	1.87	41.92

Table 3: Ablation study on CounterFact with LLaMA3-8B-Instruct. L_{full} denotes a full-vocabulary KL at target positions, including the target token, used in place of L_{nt}. w/o All corresponds to LocFT-BF with standard cross-entropy.

ages a stronger edit by requiring a higher target probability. Empirically, Reliability is low when α is small, indicating that edits do not succeed reliably when the target probability threshold is insufficiently high ($\alpha < 0.4$). In contrast, when $\alpha > 0.5$, most edits succeed, as the target token is trained to receive more than half of the probability mass.

As α increases, Generalization tends to improve, while Locality decreases. This suggests that a higher target threshold more strongly promotes the target answer not only for the rewrite prompt but also for rephrased prompts, at the cost of making pre-edit behavior harder to preserve. Nevertheless, even at sufficiently large values of α, KLOD maintains much higher Locality than LocFT-BF and OVERTONE in Table 1. Thus, α serves as a hyperparameter that explicitly controls the operating point between edit strength and locality preservation. We use $\alpha = 0.85$ as the main setting to maintain high Reliability and sufficient Generalization while avoiding locality collapse. Sensitivity analyses for the other key hyperparameters, λ_{nt} and λ_{prefix}, are provided in Appendix J.

3.5 Ablation Study

Table 3 presents the ablation results for KLOD's target objective and preservation terms. All variants maintain Reliability above 99%, indicating that each variant still induces the target edit sufficiently. Comparing the variant without both preservation terms to LocFT-BF, the bounded logit-odds objective slightly improves Locality from 1.87 to 2.02 and Capability from 41.92 to 44.24. However, these gains are limited, suggesting that KLOD's locality improvement mainly comes from the KL-based preservation terms.

In particular, L_{nt} is not merely a KL regularizer, but a key design for decoupling target editing from distribution preservation. When L_{prefix} is removed, KLOD with L_{nt} still maintains 41.30 Locality and 60.70 Capability. In contrast, replacing L_{nt} with L_{full}, a full-vocabulary KL that also includes the target token, reduces Locality to 32.90 and Capability to 57.72. This shows that including the target token in the KL term at target positions prevents the intended target update from being cleanly separated from the preservation objective. By contrast, KLOD's L_{nt} preserves only the renormalized non-target distribution, explicitly decoupling the target edit induced by L_{odds} from non-target distribution preservation.

Looking at the effect of each preservation term, removing L_{prefix} only slightly decreases Locality from 44.75 to 41.30, whereas removing L_{nt} causes a much larger drop to 9.15. This indicates that preserving the non-target distribution at target positions is especially important for maintaining locality. Removing both preservation terms yields the highest Generalization of 61.73, but Locality remains only 2.02. Therefore, the main benefit of KLOD does not come from aggressive target amplification that increases Generalization, but from the objective design that decouples target editing from non-target distribution preservation.

3.6 Scalability to Large-Scale Sequential Editing

Figure 2 shows sequential editing performance on ZsRE as the number of edits increases from 3K to 150K. These scaling results provide key evidence that KLOD's locality gain is not simply due to edit weakening. Across edit scales, KLOD maintains Reliability and Generalization comparable to LocFT-BF. Reliability remains high at all scales, reaching around 90% even after 150K edits. Gen-

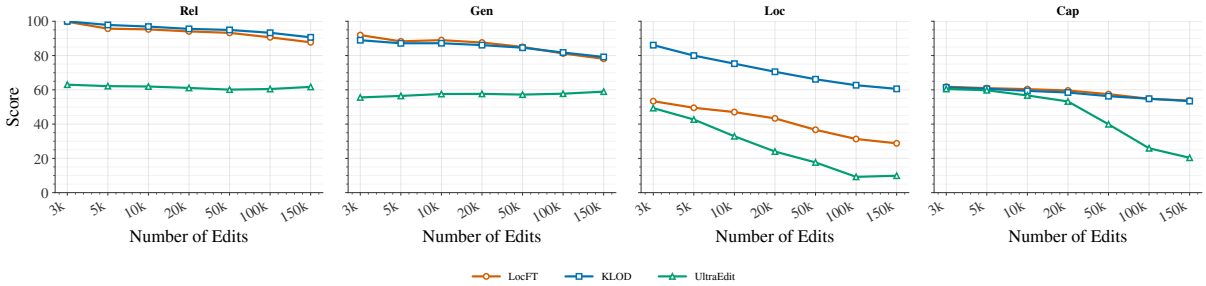

Figure 2: Scaling behavior on ZsRE from 3K to 150K sequential edits. KLOD maintains Rel. and Gen. comparable to LocFT-BF while substantially improving Loc. across edit scales with LLaMA3-8B-Instruct.

Method	LLaMA3-8B		Qwen2.5-7B	
	CounterFact	ZsRE	CounterFact	ZsRE
ROME	7.68	8.46	6.71	8.50
FT-L	1.60	1.70	0.34	1.00
LocFT-BF	0.16	0.23	0.17	0.30
KLOD	0.37	0.55	0.43	0.62

Table 4: Average editing time per edit in the 3K sequential editing setting. KLOD adds modest overhead over LocFT-BF while remaining under one second per edit.

eralization also follows a similar gradual decline to LocFT-BF as the number of edits increases, while still maintaining competitive rephrase generalization at 150K. This indicates that KLOD's distribution preservation terms mitigate locality degradation from accumulated edits without interfering with the target edit itself.

The largest difference appears in Locality. As the number of edits increases, LocFT-BF and UltraEdit show steady Locality degradation, dropping to roughly 30% and 10%, respectively, at 150K edits. In contrast, KLOD achieves over 85% Locality at 3K edits and maintains around 60% even at 150K edits. For Capability, KLOD remains comparable to LocFT-BF, whereas UltraEdit shows substantial capability degradation as the edit scale grows. These results show that KLOD effectively mitigates distributional drift and locality degradation not only in small-scale editing but also in large-scale sequential editing, while preserving high Reliability and competitive Generalization.

3.7 Editing Time

Table 4 reports the average editing time per edit request in seconds under the 3K sequential editing setting. KLOD requires 0.37–0.55 seconds on LLaMA3-8B-Instruct and 0.43–0.62 seconds on Qwen2.5-7B-Instruct, editing at a much lower cost than locate-and-edit methods such as ROME. Although KLOD is slower than LocFT-BF, this overhead comes from computing the non-target KL and prefix KL using the reference distribution of the pre-edit model in addition to the standard fine-tuning loss. Nevertheless, KLOD remains below one second per edit across all model–dataset settings, showing that it maintains practical sequential editing cost while adding distribution preservation.

4 Conclusion

This paper identified an objective-level mismatch in fine-tuning-based knowledge editing: standard softmax cross-entropy unboundedly increases target probability while leaving the non-target distribution unconstrained. In sequential editing, this can accumulate as non-target redistribution, distributional drift, and locality degradation. To address this issue, we proposed KLOD, a bounded and distribution-preserving objective that stops additional target amplification once the target probability reaches threshold α, while preserving the non-target distribution at target prediction positions and the full next-token distribution at prefix positions.

Experiments on CounterFact and ZsRE show that KLOD substantially improves locality over fine-tuning-based baselines while maintaining high reliability and competitive generalization. Ablations and distribution KL analysis confirm that these gains come from KL-based preservation terms that reduce locality collapse and output distribution drift. These results suggest that fine-tuning-based KE should be viewed not as simple likelihood maximization, but as a selective distributional update problem that separates the intended target update from non-target distribution preservation.

Limitations

KLOD does not eliminate the inherent tension among edit strength, generalization, and locality.

Instead, it makes this tension explicit and controllable through the target probability threshold α and the preservation weights λ_{nt} and λ_{prefix}. A lower α may reduce reliability, whereas a higher α can improve rephrase generalization but may decrease locality. The preservation weights also control the strength of non-target distribution preservation and prefix distribution preservation, so the appropriate operating point may vary across datasets and backbone models. Thus, while KLOD provides an objective that mitigates locality collapse, the desired reliability–generalization–locality balance must still be tuned for each deployment setting.

This work follows established KE evaluation protocols centered on CounterFact and ZsRE to directly compare the effect of objective-level design against prior methods. However, further validation is needed to determine whether the same benefits hold across more diverse real-world editing scenarios, domains, and edit types. In addition, the locality metric used in this paper is a standard measure based on preserving the pre-edit model's prediction, but exact prediction preservation may not fully capture broader behavioral preservation. Future work should examine how distribution-level preservation translates into downstream behavior preservation under broader evaluation settings.

Finally, KLOD incurs additional computation overhead over standard localized fine-tuning because it compares the pre-edit model's reference distribution with the edited model's current distribution to compute the non-target KL and prefix KL terms. Although KLOD's per-edit time remains below one second across all model–dataset settings and is lower than locate-and-edit methods in our experiments, it is higher than simpler localized fine-tuning baselines such as LocFT-BF. Therefore, latency-sensitive online editing settings may require further efficiency improvements through reference distribution caching, selective KL computation, or approximation.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

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

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

Xiaojie Gu, Guangxu Chen, Jungang Li, Jia-Chen Gu, Xuming Hu, and Kai Zhang. 2025. UltraEdit: Training-, subject-, and memory-free lifelong editing in large language models. *Preprint*, arXiv:2505.14679.

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

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

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

Tianci Liu, Ruirui Li, Zihan Dong, Hui Liu, Xianfeng Tang, Qingyu Yin, Linjun Zhang, Haoyu Wang, and Jing Gao. 2025. Mitigating heterogeneous token overfitting in LLM knowledge editing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 38744–38766. PMLR.

Llama Team. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35.

Qwen Team. 2024. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Du Su, Dawei Yin, and Huawei Shen. 2024. The fall of ROME: Understanding the collapse of LLMs in model editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4079–4087, Miami, Florida, USA. Association for Computational Linguistics.

Wanli Yang, Fei Sun, Rui Tang, Hongyu Zang, Du Su, Qi Cao, Jingang Wang, Huawei Shen, and Xueqi Cheng. 2026. Fine-tuning done right in model editing. In *International Conference on Learning Representations*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2025. Uncovering overfitting in large language model editing. In *International Conference on Learning Representations*.

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

A Related Work

A.1 Locate-and-Edit Paradigm

A major line of knowledge editing follows the locate-and-edit paradigm, which identifies and directly modifies internal computations or parameter regions responsible for factual associations. ROME uses causal tracing to identify feed-forward modules important for factual recall and modifies factual associations through rank-one updates (Meng et al., 2022). AlphaEdit focuses on the observation that parameter perturbations in sequential editing can damage preserved knowledge, and proposes a projection-based editing strategy (Fang et al., 2025). These methods are primarily designed around where the target knowledge is localized, such as specific layers, modules, or subspaces, and how the corresponding parameter update should be applied.

Our work studies fine-tuning-based knowledge editing from a complementary perspective. We do not aim to identify new parameter regions responsible for factual associations or design a closed-form update rule. Instead, given a set of editable parameters, we analyze what training objective should be optimized when these parameters are updated through gradient-based optimization. In particular, we focus on an objective-level issue: increasing the likelihood of the target answer for edit success may unnecessarily redistribute the output distribution outside the target token. Thus, unlike locate-and-edit methods that mainly focus on edit location, KLOD focuses on distribution-preserving objective design that enables localized fine-tuning to achieve target editing and locality preservation simultaneously.

A.2 Fine-Tuning-Based Knowledge Editing

Fine-tuning-based editing is architecture-agnostic and simple to implement, but it has long been regarded as a disruptive baseline in knowledge editing because gradient updates can affect broader model behavior beyond the target fact. Recent studies, however, show that the reliability and locality of fine-tuning-based editing are not determined solely by the update mechanism itself, but depend heavily on the objective, editable parameter selection, and optimization protocol. StandardFT shows that standard fine-tuning can achieve competitive editing performance through conditional likelihood and locality training (Gangadhar and Stratos, 2024). LocFT-BF improves the stability of fine-tuning-based editing in large-scale sequential editing through localized tuning and breadth-first mini-batch optimization (Yang et al., 2026).

These studies suggest that fine-tuning should not be viewed merely as a weak baseline, but as an optimization framework that can be redesigned for knowledge editing. However, existing fine-tuning-based editing methods mainly focus on how reliably the target answer can be learned or how edit requests should be optimized. In contrast, we focus on the fact that standard cross-entropy-based fine-tuning does not explicitly preserve the distributional structure among non-target tokens while increasing target probability. Cross-entropy is effective at increasing target token probability, but it does not distinguish whether this increase comes from raising the target logit or from changing the non-target logit mass and the relative distribution among non-target tokens. Since knowledge editing requires not only making the target answer sufficiently likely but also keeping behavior unrelated to the target close to the pre-edit model, such non-target redistribution can accumulate as distributional drift and locality degradation in sequential editing.

Recent work such as OVERTONE and Editing Overfit shows that naive target maximization can induce objective-level side effects, including token-level overfitting and excessive target probability assignment (Liu et al., 2025; Zhang et al., 2025).

These studies are closely related to ours in that they analyze how the fine-tuning objective affects edit behavior. However, KLOD goes beyond mitigating target-side overfitting or stabilizing target likelihood optimization by directly controlling non-target distribution drift during target probability increase. In other words, KLOD is not concerned only with how strongly the model should predict the target token, but with how the non-target distribution should be preserved while the target token is edited.

KLOD is also distinct from simple KL-regularized fine-tuning. Applying KL regularization to the full output distribution at target positions can directly conflict with the edit objective, since the edited target token must increase in probability while the reference distribution encourages the original distribution to be preserved. To avoid this conflict, KLOD excludes the target token at target positions, defines a renormalized non-target distribution over the vocabulary excluding the target token, and keeps this distribution close to that of the pre-edit model. This target-excluded preservation allows the intended target probability increase while suppressing unnecessary redistribution in the non-target distribution. In addition, KLOD uses a bounded logit-odds objective that stops further target amplification once the target probability reaches a specified threshold, thereby mitigating over-editing caused by unbounded likelihood maximization. Thus, while KLOD follows the broader line of fine-tuning-based KE, it differs from prior methods by explicitly decoupling target editing from non-target distribution preservation through a distribution-preserving objective.

A.3 Sequential Editing and Side Effects

In practical knowledge editing, edits accumulate over time, making stability under sequential or lifelong editing as important as the success of a single edit. Continual editing and Editing-at-Scale studies show that as the number of edits increases, models may forget previous edits or suffer degradation in downstream task performance (Li and Chu, 2024; Gupta et al., 2024). It has also been reported that model editing can harm general abilities such as reasoning, natural language inference, and question answering (Gu et al., 2024). These findings indicate that knowledge editing should be evaluated not only by Reliability, but also by side effects such as Generalization, Locality, Retention, and Capability.

This work addresses such side effects at the output-distribution level. While prior sequential editing studies mainly observe and mitigate task-level forgetting, locality degradation, and capability drop caused by edit accumulation, KLOD approaches these issues from the perspective that such degradation can be linked to distributional drift permitted by the fine-tuning objective. In particular, standard cross-entropy is not designed to preserve the distributional structure outside the target token while increasing the target answer probability. As many edits accumulate, small non-target redistribution may therefore accumulate and appear as locality degradation. KLOD aims to mitigate this accumulated output distribution drift while maintaining edit success by keeping the non-target distribution at target positions and the next-token distribution at prefix positions close to those of the pre-edit model. In this sense, KLOD does not treat the side effects of sequential editing only as task-level metrics, but directly intervenes in distribution-level changes induced by the fine-tuning objective.

B Dataset Details

ZsRE. Each edit request in ZsRE consists of a rewrite question and a target answer, along with rephrased questions and locality prompts. We use the rewrite question as the edit input and the rephrased questions for Generalization evaluation. The locality prompts are used to measure whether the pre-edit model's prediction is preserved on questions unrelated to the edit. Thus, ZsRE evaluates both whether edited knowledge generalizes to semantically equivalent expressions and whether behavior on unrelated questions is preserved.

CounterFact. Each edit request in CounterFact consists of a rewrite prompt and a target object, along with paraphrase prompts and locality prompts. The rewrite prompt is used for Reliability evaluation, the paraphrase prompts for Generalization evaluation, and the locality prompts for evaluating Locality, which measures pre-edit behavior preservation. Thus, CounterFact evaluates both whether the target fact is incorporated and whether unrelated facts are preserved.

C Implementation Details

KLOD is implemented under the localized fine-tuning with Breadth-First pipeline setting of LocFT-BF (Yang et al., 2026). The Breadth-First pipeline

is not an online-style sequential optimization procedure that fully optimizes each edit request before moving to the next. Instead, it jointly optimizes the full edit set through repeated mini-batch updates over multiple epochs. This optimization protocol is used to mitigate overwriting among sequential edits.

In localized fine-tuning, we do not update all parameters of the backbone model. Instead, only the MLP down-projection matrix in a specified editable layer is made trainable. Specifically, we edit the MLP down-projection matrix at layer 22 for LLaMA3-8B-Instruct and at layer 6 for Qwen2.5-7B-Instruct. Attention modules, embedding layers, layer normalization parameters, and all other MLP parameters are kept frozen.

Each edit example is constructed by concatenating the edit prompt x and target response $y = (y_1, \ldots, y_m)$ as $s = x \oplus y$. In the standard target-only fine-tuning objective, negative log-likelihood is computed only at positions $\mathcal{T}$ that predict the target response, while prompt positions and padding positions are masked out from NLL computation. Thus, the prompt x is used only as the conditioning context for target response generation, and no direct NLL supervision is applied to the prompt tokens themselves.

KLOD keeps the same LocFT-BF optimization environment but extends the standard target-only cross-entropy objective into a distribution-preserving objective. Specifically, at target prediction positions $\mathcal{T}$, KLOD computes the bounded logit-odds loss and the target-excluded non-target KL. At valid prefix positions $\mathcal{P}$, excluding target prediction positions, KLOD adds the prefix KL to preserve the next-token distribution of the pre-edit model. Padding positions are excluded from all objective terms.

The objective hyperparameters of KLOD are fixed across all main experiments. We set the target-token probability threshold to $\alpha = 0.85$, and the loss weights for non-target distribution preservation and prefix distribution preservation to $\lambda_{\text{nt}} = 0.6$ and $\lambda_{\text{prefix}} = 1.2$, respectively. We use the Adam optimizer. For each mini-batch, gradients are computed only for the selected MLP down-projection matrix, followed by a single optimizer update. Backbone-specific hyperparameters, including learning rate, batch size, number of epochs, and weight decay, are summarized in Table 5. All experiments were conducted using three NVIDIA GeForce RTX 3090 GPUs. This hardware environment was used for both the main editing experiments and the editing-time measurements.

Hyperparameter	LLaMA3-8B	Qwen2.5-7B
Editable layer	22	6
Editable module	MLP down-proj	MLP down-proj
α	0.85	0.85
λ_{nt}	0.6	0.6
λ_{prefix}	1.2	1.2
Batch size	100	100
Epochs	25	25
Learning rate	5.0e-4	5.0e-4
Weight decay	0	0

Table 5: Implementation hyperparameters for KLOD.

D Evaluation Metrics

We evaluate the edited model after sequential editing using four metrics: Reliability, Generalization, Locality, and Capability. Following the traditional teacher-forcing evaluation protocol in EasyEdit, evaluation is based on token-level prediction accuracy at target positions rather than answer-level exact match. For each edit request i, let x_i denote the rewrite prompt, x_i^{reph} the rephrased prompt, and $y_i = (y_{i,1}, \ldots, y_{i,m_i})$ the target answer token sequence. We denote the final edited model by f_{θ_N} and the pre-edit model by f_{θ_0}.

Given a prompt x and a target token sequence $y = (y_1, \ldots, y_m)$, we define the prediction of model f_θ at the t-th target position as

$$\hat{y}_t^\theta(x, y) = \arg\max_{v \in \mathcal{V}} p_\theta(v \mid x, y_{<t}). \tag{18}$$

The corresponding token-level accuracy is

$$\text{Acc}_\theta(x, y) = \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\left[\hat{y}_t^\theta(x, y) = y_t\right]. \tag{19}$$

Thus, for multi-token answers, the score is computed by the fraction of correctly predicted tokens, rather than requiring the entire string to match exactly.

Reliability. Reliability measures how accurately the edited model predicts the target answer for the rewrite prompt:

$$\text{Rel.} = \frac{1}{|\mathcal{I}_{\text{rw}}|} \sum_{i \in \mathcal{I}_{\text{rw}}} \text{Acc}_{\theta_N}(x_i^{\text{rw}}, y_i). \tag{20}$$

Generalization. Generalization measures whether the same target answer is preserved under rephrased prompts:

$$\text{Gen.} = \frac{1}{|\mathcal{I}_{\text{reph}}|} \sum_{i \in \mathcal{I}_{\text{reph}}} \text{Acc}_{\theta_N}(x_i^{\text{reph}}, y_i). \tag{21}$$

Locality. Locality evaluates whether the pre-edit model's prediction is preserved on locality prompts unrelated to the edit. Let x_i^{loc} be a locality prompt and $y_i^{\text{loc}} = (y_{i,1}^{\text{loc}}, \ldots, y_{i,\ell_i}^{\text{loc}})$ be the reference continuation. The locality prediction of model f_θ is defined as

$$\hat{y}_{i,t}^{\theta,\text{loc}} = \arg\max_{v \in \mathcal{V}} p_\theta(v \mid x_i^{\text{loc}}, y_{i,<t}^{\text{loc}}). \tag{22}$$

Locality is computed as the fraction of token-level predictions that match between the edited model and the pre-edit model:

$$\text{Loc.} = \frac{1}{|\mathcal{I}_{\text{loc}}|} \sum_{i \in \mathcal{I}_{\text{loc}}} \frac{1}{\ell_i} \sum_{t=1}^{\ell_i} \mathbb{I}\left[\hat{y}_{i,t}^{\theta_N,\text{loc}} = \hat{y}_{i,t}^{\theta_0,\text{loc}}\right]. \tag{23}$$

Thus, Locality measures how well the original behavior is preserved on locality prompts, rather than whether the edited model predicts a specific target answer.

Capability. Capability evaluates the model's general abilities after sequential editing by averaging performance on GSM8K, MMLU, NQ Open, SST-2, and WMT16:

$$\text{Cap.} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \text{Score}_b(f_{\theta_N}), \tag{24}$$

where $\mathcal{B}$ denotes the benchmark set consisting of GSM8K, MMLU, NQ Open, SST-2, and WMT16. The task-specific score for each benchmark is converted to a $[0, 100]$ scale before averaging.

Overall Score. The Score reported in the main results is the arithmetic mean of the four metrics:

$$\text{Score} = \frac{\text{Rel.} + \text{Gen.} + \text{Loc.} + \text{Cap.}}{4}. \tag{25}$$

E Baseline Details

This section describes the implementation details and hyperparameter settings of the baseline editors used in our experiments. ROME, AlphaEdit, UltraEdit, FT-L, and FT-M are evaluated based on the implementations released in EasyEdit. For these methods, we use the public configurations provided by EasyEdit and do not perform additional hyperparameter tuning on the test set. Model forward computation and evaluation are performed in bfloat16 precision whenever possible. However, for methods such as ROME and AlphaEdit that rely on method-specific linear algebra operations, including covariance statistics, matrix inversion, and projection, we compute the corresponding statistics and linear algebra components in float32 precision or follow the implementation default precision for numerical stability. LocFT-BF is evaluated using the official implementation released by the authors, with hyperparameters reported in the original paper. Since OVERTONE does not provide an official public implementation, we reimplement it based on the method described in the paper. ROME, FT-L, and FT-M use a batch size of 1, while the remaining baselines use a batch size of 100. All baselines are evaluated under the same backbone model, tokenizer, edit stream, and evaluation protocol. In sequential editing experiments, model parameters are not reinitialized at each edit step; instead, edit requests are applied cumulatively.

ROME. ROME is a locate-and-edit method that identifies internal computations associated with factual associations through causal tracing and modifies a single factual association by applying a rank-one update to selected feed-forward weights (Meng et al., 2022). We use the ROME implementation and hyperparameter configuration released in EasyEdit. For both Llama-3-8B-Instruct and Qwen2.5-7B-Instruct, we edit the MLP down-projection module at layer 5. For both models, the subject token is set to `subject_last`, and value vector optimization is performed for 25 epochs. The learning rate is set to 5×10^{-1}, weight decay to 10^{-3}, clamp norm factor to 4, and KL factor to 0.0625. Covariance statistics are computed from 100,000 Wikipedia samples, with the covariance dtype kept in float32. Value vector optimization follows the default precision of the EasyEdit implementation, while model forward computation uses bfloat16 precision whenever possible.

AlphaEdit. AlphaEdit addresses the issue that parameter perturbations introduced during locate-then-edit procedures can damage preserved knowledge. To mitigate this, it projects edit perturbations onto the null space of preserved knowledge before applying the update, aiming to reduce locality degradation in sequential editing (Fang et al., 2025). We use the AlphaEdit implementation and hyperparameter configuration released in EasyEdit. For both Llama-3-8B-Instruct and Qwen2.5-7B-Instruct, we edit the MLP down-projection modules in layers 4–8, with layer selection set to `all` and the fact token set to `subject_last`. The batch size is set to 100. The null-space projection threshold is set to 2×10^{-2}, and the L_2 coefficient is set to

1. For Llama-3-8B-Instruct, the value optimization learning rate is set to 1×10^{-1}, weight decay to 0.5, and clamp norm factor to 0.75. For Qwen2.5-7B-Instruct, the value optimization learning rate is set to 5×10^{-1}, weight decay to 10^{-3}, and clamp norm factor to 4. For both models, value optimization is performed for 25 epochs, and the KL factor is set to 0.0625. Covariance-related statistics are computed from 100,000 Wikipedia samples in float32 precision. Method-specific linear algebra operations, including null-space projection, use float32 or the implementation default precision for numerical stability, while model forward computation is performed in bfloat16 precision whenever possible.

UltraEdit. UltraEdit is a training-free, subject-free, and memory-free editing method designed for scalable lifelong editing. It computes parameter shifts in one step using lightweight linear algebra operations and employs lifelong normalization, which updates feature statistics as editing turns accumulate, to improve stability in large-scale lifelong editing (Gu et al., 2025). We use the UltraEdit implementation and hyperparameter configuration released in EasyEdit. For Llama-3-8B-Instruct, we edit the MLP gate projection in layers 11–15 and the MLP up projection in layers 18–24. For Qwen2.5-7B-Instruct, we edit the MLP gate projection and up projection in layers 18–26. For both models, the learning rate is set to 1×10^{-6}, and the token setting is set to `mask`. The batch size, one-pass batch size, and editor batch size are set to 100, 10, and 1024, respectively. Model forward computation is performed in bfloat16 precision whenever possible, while method-specific linear algebra operations follow the default precision of the EasyEdit implementation.

FT-L and FT-M. FT-L and FT-M are fine-tuning-based editing baselines (Zhang et al., 2024). FT-L fine-tunes localized parameters using supervision on the target answer token at the last token position of the prompt. FT-M applies cross-entropy loss to the full target answer while masking prompt tokens, making the objective closer to standard target-answer fine-tuning. We construct FT-L and FT-M based on the FT implementation released in EasyEdit. In the EasyEdit configuration, we use the setting where `objective_optimization` is `prompt_last` as FT-L, and the setting where it is `target_new` as FT-M. For LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct, we fine-tune the MLP down-projection weights at layer 21 and layer 27, respectively. Optimization is performed for 25 epochs, with a learning rate of 5×10^{-4} and a maximum sequence length of 40 in the hyperparameter configuration. The batch size is set to 1 for both FT-L and FT-M. Weight decay and KL factor are both set to 0, and no norm constraint is used. Model forward and backward computation are performed in bfloat16 precision by default, while optimizer states and other training components follow the implementation default precision.

LocFT-BF. LocFT-BF analyzes that the weak performance of prior fine-tuning-based editors may stem not from an inherent limitation of fine-tuning itself, but from implementation choices such as depth-first sample-wise optimization and suboptimal tuning locations. Based on this analysis, LocFT-BF combines breadth-first mini-batch optimization with localized tuning location selection to provide a fine-tuning-based editing baseline better suited for sequential editing (Yang et al., 2026). We use the official implementation released by the authors. For Llama-3-8B-Instruct and Qwen2.5-7B-Instruct, we tune the MLP down-projection modules at layer 22 and layer 6, respectively. Optimization is performed for 25 epochs, with a learning rate of 5×10^{-4}, batch size of 100, and weight decay of 0. Model forward and backward computation are performed in bfloat16 precision, while optimizer states and other training components follow the default precision of the official implementation.

OVERTONE. OVERTONE identifies heterogeneous token overfitting (HTO), where tokens in the target sequence overfit at different rates during knowledge editing. To mitigate this issue, it uses a token-level smoothing objective that adaptively adjusts the target distribution according to each token's fitting state (Liu et al., 2025). Since OVERTONE does not provide an official public implementation, we reimplement it based on the objective and reported hyperparameters in the original paper.

The implementation is built on the same localized fine-tuning pipeline as LocFT-BF and KLOD. For an objective-level comparison, we keep the editable layer, editable module, batch size, learning rate, number of epochs, edit order, decoding, and evaluation setting identical. We do not separately tune OVERTONE for the test performance of each dataset or backbone. The batch size is set to 100, and the OVERTONE-related coefficients are fixed to $\lambda = 0.1$, $\epsilon = 0.01$, and $n_\sigma = 0.5$, following

the original paper. Model forward and backward computation are performed in bfloat16 precision by default, while optimizer states and other training components follow the implementation default precision.

E.1 Code Availability

To support reproducibility, we provide code and scripts in an anonymized repository: https://anonymous.4open.science/r/KLOD-DD6D.

F Gradient of Softmax Cross-Entropy

This section summarizes the learning signal of softmax cross-entropy discussed in Section 2.2 from a gradient perspective. Consider a single prediction position t, with gold next token a_t and logits z_t^θ. The cross-entropy loss is

$$\ell_t = -\log p_\theta(a_t \mid s_{\leq t}) = -z^\theta_{t,a_t} + \log \sum_{v \in \mathcal{V}} \exp z^\theta_{t,v}. \tag{26}$$

Given the softmax probability

$$p_\theta(v \mid s_{\leq t}) = \frac{\exp z^\theta_{t,v}}{\sum_{u \in \mathcal{V}} \exp z^\theta_{t,u}}, \tag{27}$$

the gradient with respect to each vocabulary logit is

$$\frac{\partial \ell_t}{\partial z^\theta_{t,v}} = p_\theta(v \mid s_{\leq t}) - \mathbb{I}[v = a_t]. \tag{28}$$

Thus, for the target token a_t,

$$\frac{\partial \ell_t}{\partial z^\theta_{t,a_t}} = p_\theta(a_t \mid s_{\leq t}) - 1, \tag{29}$$

whereas for a non-target token $v \neq a_t$,

$$\frac{\partial \ell_t}{\partial z^\theta_{t,v}} = p_\theta(v \mid s_{\leq t}). \tag{30}$$

In logit space, gradient descent therefore increases the target logit while assigning non-target logits a decreasing signal proportional to their current model probabilities. This gradient structure is effective for increasing the target token probability, but it does not require the relative structure of the non-target distribution to remain close to that of the pre-edit model. In other words, cross-entropy directly optimizes target probability increase, but does not explicitly control how the probability mass outside the target token is redistributed. Under shared parameter updates and sequential editing, such unconstrained non-target redistribution can accumulate as distributional drift and locality degradation.

To mitigate this issue, KLOD's non-target preservation term keeps the renormalized non-target distribution, excluding the target token, close to the pre-edit model. This allows the intended increase in target probability while directly suppressing unnecessary changes in the non-target distribution.

G Details of Distribution KL Analysis

This section describes the implementation details of the distribution KL analysis used in Section 3.3. Each evaluation item consists of a prompt x and a target sequence y, and we compare the next-token distributions of the pre-edit model f_{θ_0} and the edited model f_θ on the teacher-forced sequence $s = x \oplus y$. Rewrite and rephrase items use the edit prompt or rephrased prompt together with the edited target answer, respectively, while locality items use a locality prompt unrelated to the edit and its corresponding ground-truth answer.

KL is computed at shifted prediction positions in causal language modeling. That is, the logits at position t are treated as the distribution for predicting the next token s_{t+1}. Prompt KL is computed at prompt-side positions that predict tokens within the prompt, while the position that predicts the first target token after observing the final prompt token is included in Target KL. Target KL measures full-vocabulary distribution drift at positions that predict target answer tokens under teacher forcing. All KL values are computed in the pre-edit-to-edited direction:

$$D_{\mathrm{KL}}\left(p_{\theta_0}(\cdot \mid s_{\leq t}) \,\|\, p_\theta(\cdot \mid s_{\leq t})\right). \tag{31}$$

Position split example. For example, suppose the teacher-forced sequence is

$$s = [x_1, x_2, x_3, y_1, y_2]. \tag{32}$$

Under shifted prediction in causal language modeling, x_1 predicts x_2, x_1, x_2 predict x_3, and x_1, x_2, x_3 predict y_1. Therefore, the positions predicting x_2 and x_3 are included in Prompt KL, whereas the position predicting the first target token y_1 is included in Target KL. In other words, even if the final prompt token is used as context, the corresponding position is treated as a target prediction position if it predicts the first target token.

For rewrite and rephrase contexts, we also compute Non-target KL at target positions. Let a_t denote the gold next token at each target position. We

Method	Rewrite			Rephrase			Locality	
	Prompt KL	Target KL	Non-target KL	Prompt KL	Target KL	Non-target KL	Prompt KL	Target KL
ROME	8.019	7.917	7.440	8.651	8.071	7.521	8.095	6.430
AlphaEdit	12.919	12.981	12.949	12.669	12.157	12.130	13.293	12.962
UltraEdit	0.702	6.030	1.690	**0.228**	3.999	**1.218**	**0.188**	2.985
LocFT-BF	4.106	16.573	10.200	2.672	9.980	8.491	3.463	10.746
OVERTONE	3.959	16.391	10.259	2.482	9.646	8.381	3.396	10.863
KLOD	**0.111**	**4.035**	**0.552**	0.462	**2.712**	1.508	0.261	**1.538**

Table 6: Distributional KL divergence after 3K sequential editing on CounterFact with Qwen2.5-7B-Instruct. Lower values indicate smaller distributional drift from the pre-edit model. Best values are shown in **bold**. We use the same evaluation protocol as Table 2.

exclude this token from the vocabulary and renormalize the distribution over the remaining tokens:

$$q_\theta^t(v) = \frac{p_\theta(v \mid s_{\leq t})}{1 - p_\theta(a_t \mid s_{\leq t})}, \quad v \in \mathcal{V} \setminus \{a_t\}. \quad (33)$$

Non-target KL is computed as the KL divergence between $q_{\theta_0}^t$ and q_θ^t, measuring relative distribution drift among non-target tokens rather than changes in the target probability itself. For locality contexts, we report only Prompt KL and Target KL, since the goal is not to analyze the non-target distribution around the edited target answer.

The final reported values are token-level averages rather than example-level averages. For each context type, we accumulate the position-wise KL sum and the number of evaluated tokens, and compute the values in Table 2 by dividing the total KL sum by the total token count.

H Additional Distribution KL Results

Table 6 reports additional distribution KL analysis after 3K sequential editing on CounterFact with Qwen2.5-7B-Instruct. The overall trend is similar to the results on LLaMA3-8B-Instruct. KLOD achieves the lowest Prompt KL, Target KL, and Non-target KL in the rewrite context, and it also yields the lowest Target KL in the rephrase and locality contexts. In particular, LocFT-BF and OVERTONE show high Target KL and Non-target KL in the rewrite and rephrase contexts, whereas KLOD substantially reduces both. This indicates that KLOD's KL-based preservation terms effectively suppress target-side distributional drift even while increasing the target probability.

UltraEdit achieves the lowest Prompt KL in the rephrase and locality contexts, preserving some prompt-side next-token behavior well. It also obtains the lowest Rephrase Non-target KL. However, KLOD achieves lower Target KL in both the rephrase and locality contexts, as well as lower Rewrite Non-target KL. This suggests that KLOD more consistently preserves the target-position distribution in contexts directly tied to the edited answer, whereas UltraEdit is more effective at reducing prompt-side drift in some non-rewrite contexts. Therefore, KLOD shows consistent advantages in target-side distribution preservation also on Qwen2.5-7B-Instruct.

Method	Rel.	Gen.	Loc.	Cap.
KLOD	99.89	86.85	86.47	61.42
w/o L_{prefix}	99.97	88.67	81.26	60.92
w/o L_{nt}	99.57	87.99	82.09	61.68
w/o L_{prefix} and L_{nt}	99.92	89.25	50.93	61.78
w/o All	99.93	89.21	51.24	62.10

Table 7: Ablation study on ZsRE with LLaMA3-8B-Instruct. The row without both preservation terms uses only the bounded logit-odds objective. The w/o All variant corresponds to LocFT-BF with the standard cross-entropy objective.

I Additional Ablation on ZsRE

Table 7 presents additional ablation results on ZsRE with LLaMA3-8B-Instruct. All variants achieve high Reliability, indicating that the target edits remain effective under different objective variants. Unlike the CounterFact setting, the bounded logit-odds objective alone does not provide a clear advantage over the standard cross-entropy counterpart on ZsRE. Specifically, removing both preservation terms yields 50.93 Locality, which is comparable to the w/o All variant corresponding to LocFT-BF, which obtains 51.24 Locality. Generalization and Capability are also nearly unchanged between these two variants.

Nevertheless, the KL-based preservation terms remain important for locality preservation. Removing either L_{prefix} or L_{nt} decreases Locality

λ_{nt}	Rel.	Gen.	Loc.	Cap.
0.2	99.70	48.23	35.33	60.98
0.4	99.63	47.43	40.37	60.78
0.6	99.70	47.37	44.75	60.77
0.8	99.63	45.07	47.42	61.07

Table 8: Sweep over λ_{nt} with λ_{prefix} fixed at 1.2 on CounterFact with LLaMA3-8B-Instruct.

λ_{prefix}	Rel.	Gen.	Loc.	Cap.
0.5	99.77	52.00	28.38	60.81
0.9	99.73	50.78	26.07	61.23
1.2	99.70	50.00	24.55	61.11
1.5	99.73	49.00	28.42	60.93

Table 9: Sweep over λ_{prefix} with λ_{nt} fixed at 0.1 on CounterFact with LLaMA3-8B-Instruct.

from 86.47 to 81.26 and 82.09, respectively, while removing both preservation terms causes a much larger drop to 50.93. This indicates that the two preservation terms are complementary on ZsRE: each term contributes moderately in isolation, but their combination is necessary to maintain high Locality. Overall, the ZsRE ablation confirms that KLOD's locality improvement mainly comes from distribution-preserving regularization rather than from the bounded target objective alone.

J Lambda Sweep

This section analyzes the effect of the two preservation loss weights in KLOD, λ_{nt} and λ_{prefix}, on editing performance. All experiments are conducted on CounterFact 3K sequential editing with LLaMA3-8B-Instruct, with the target probability threshold fixed to $\alpha = 0.85$.

Table 8 shows the results of varying λ_{nt} while fixing $\lambda_{\text{prefix}} = 1.2$. As λ_{nt} increases, Locality consistently improves from 35.33 to 47.42, while Generalization gradually decreases from 48.23 to 45.07. Reliability and Capability remain nearly stable across all settings. This indicates that stronger non-target distribution preservation directly improves locality, with a moderate trade-off in rephrase generalization but without substantially harming edit success or capability.

Table 9 varies λ_{prefix} while fixing $\lambda_{\text{nt}} = 0.1$. We use this weaker non-target preservation setting to make the marginal effect of λ_{prefix} easier to observe, rather than to reproduce the main KLOD configuration. Across this sweep, Reliability remains high and Capability stays stable around 61. Generalization decreases gradually as λ_{prefix} increases, while Locality remains in the mid-to-high 20s without a monotonic trend. These results suggest that prefix preservation alone is insufficient to recover high locality when non-target preservation is weak. Together with Table 8, this supports the main design choice of combining prefix preservation with sufficiently strong non-target distribution preservation.