

EtCon: Edit then Consolidate for Reliable Knowledge Editing

Ruilin Li^{1,2} Yibin Wang^{2,3} Wenhong Zhu^{2,5} Jinghao Zhang^{2,6}

Chenglin Li² Chenliang Li^{1†} Junchi Yan^{2,5†} Jiaqi Wang^{2,4†}

¹Wuhan University ²Shanghai Innovation Institute ³Fudan University ⁴JD.com

⁵Shanghai Jiao Tong University ⁶University of Science and Technology of China

Abstract

Knowledge Editing (KE) aims to update specific facts in large language models (LLMs) without full retraining. Prior efforts sought to tune the knowledge layers of LLMs, achieving near-perfect performance in teacher-forced evaluations. However, under realistic autoregressive generation evaluation, their effectiveness remains limited by two core challenges: (1) even when methods successfully inject new knowledge, pre-trained capabilities can degrade, and (2) more critically, the injected parametric knowledge is not reliably expressed at inference time. To this end, we formalize knowledge editing as a two-stage problem and propose **EtCon**, an edit-then-consolidate paradigm that couples targeted edits with post-edit consolidation. Within EtCon, the edit stage applies **Targeted Proximal Supervised Fine-Tuning (TPSFT)** to update parametric knowledge with minimal disruption to pre-trained capabilities. The consolidation stage applies **Group Relative Policy Optimization (GRPO)** to align the model’s autoregressive generation with the edited parametric knowledge. Extensive experiments demonstrate that our instantiation (TPSFT + GRPO) improves editing reliability and real-world generalization while preserving pre-trained capabilities, and that EtCon consistently improves existing editing methods when instantiated with them.

1 Introduction

Large language models (LLMs) encode a vast amount of factual knowledge in their parameters, yet the world keeps evolving and this knowledge quickly becomes outdated (Dohare et al., 2024; Zheng et al., 2025). Knowledge editing (KE) addresses this by editing specific facts through targeted parameter updates, without costly continual pre-training (Meng et al., 2022a,b; Zhang et al., 2025; Rozner et al., 2024; Wang et al., 2026; Park

et al., 2025; Zhang et al., 2026; Liu et al., 2026). However, recent studies have shown that KE methods, while achieving near-perfect performance in teacher-forced evaluation, degrade substantially under realistic autoregressive generation, especially in lifelong editing scenarios (Yang et al., 2025, 2026) (Table 2). This suggests that reliable knowledge editing requires not only editing parametric knowledge, but also consistently expressing it during autoregressive generation while preserving the model’s pre-trained capabilities.

In this work, our empirical analysis reveals two recurring issues. (1) **Pre-trained capability degradation**. Many existing methods perform edits by maximizing the likelihood of the new target on a small set of editing samples. Without explicit constraints, this few-shot objective can induce overspecialization to the editing examples, degrading pre-trained capabilities. In lifelong editing, as illustrated in Figure 1 (left), such drift may accumulate across lifelong edits and may even lead to model collapse (Gupta et al., 2024, 2025b). (2) **Parametric knowledge-behavior mismatch in autoregressive generation**. More critically, even when parametric knowledge is successfully edited, it may not be consistently expressed along autoregressive trajectories. During autoregressive generation, broader parameters retaining pre-edit knowledge are activated, competing with the injected fact and increasing the likelihood of reverting to outdated information.

To address these limitations, we propose **EtCon**, a new edit-then-consolidate paradigm (Fig. 3) that couples constrained targeted edits with post-edit consolidation to improve both editing reliability and pre-trained capability preservation. EtCon operates in two stages: **Stage 1 (Edit)**. Targeted Proximal Supervised Fine-Tuning (TPSFT), a refined variant of PSFT (Zhu et al., 2026), performs a constrained targeted edit by updating only knowledge-relevant FFN layers, with trust-region constraints

[†]Corresponding authors.

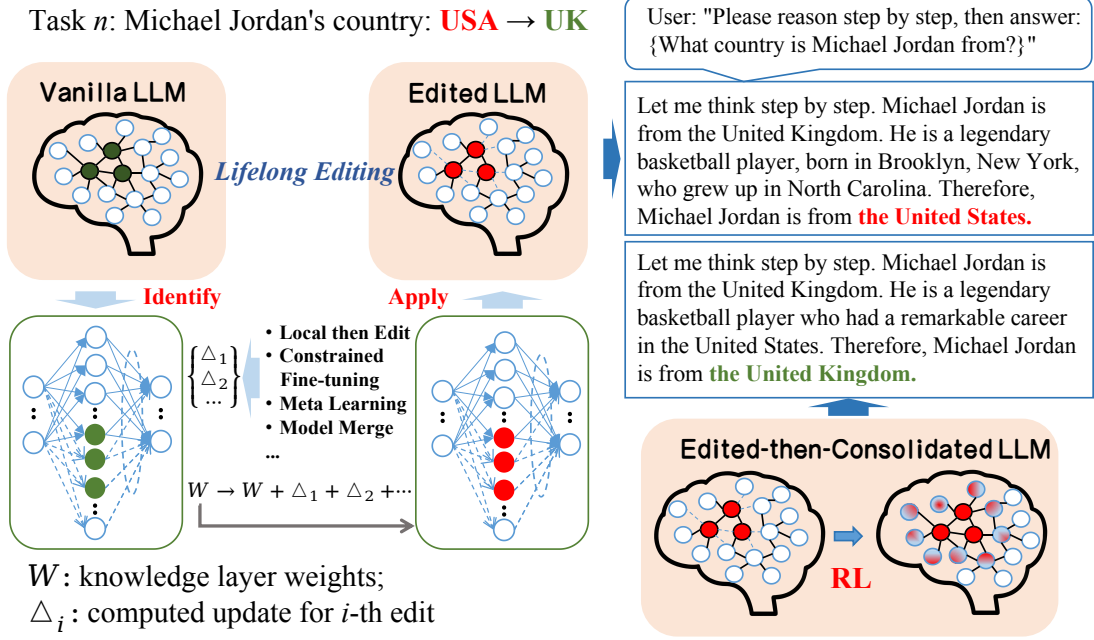


Figure 1: **Two Challenges in Knowledge Editing.** (Left) Accumulated lifelong edits may degrade pre-trained capabilities. (Right) Edited parametric knowledge may not be reliably expressed at inference time.

and self-generated CoT-based targets. This design localizes parameter updates and mitigates over-specialization, thereby preserving pre-trained capabilities. **Stage 2 (Consolidate).** GRPO consolidates the edit via trajectory-level optimization under reward signals. This stage aligns the parametric update with inference-time behavior, improving the robustness of the edited knowledge along autoregressive trajectories. We conduct extensive experiments across multiple LLMs and benchmarks. Under real-world evaluation, EtCon improves editing Reliability and Generalization by 35%–50% over strong baselines. It also preserves critical pre-trained capabilities, and the consolidation stage can further benefit existing editing methods when plugged in. Our contributions are as follows:

(1) We first decompose the performance degradation of many KE methods under realistic evaluation into two distinct issues: pre-trained capability degradation and parametric knowledge-behavior mismatch.

(2) We posit that a post-edit consolidation stage is necessary to align parametric knowledge injection and autoregressive generation behavior, and empirically validate its effectiveness across multiple existing editing methods.

(3) We propose EtCon, an edit-then-consolidate paradigm, and instantiate it with TPSFT for constrained parametric edits that preserve pre-trained capabilities, and GRPO for aligning autoregressive generation with updated parametric knowledge.

2 Related Work

From the perspective of model architecture, existing knowledge editing methods can be coarsely grouped into two families. **In-Place Parametric Edits** methods preserve the vanilla LLM architecture. The locate-then-edit paradigm (Meng et al., 2022a; Dai et al., 2025; Li et al., 2024; Zhong et al., 2025; Zhang et al., 2024c) identifies knowledge locations within LLMs and modifies targeted parameters through gradient-based or analytical solutions. PEFT methods (Zhu et al., 2020; Han et al., 2024; Wang et al., 2024b; Gupta et al., 2025a) directly update model parameters via regularized gradient descent to achieve knowledge updates while constraining side effects (Liu et al., 2025). These approaches seamlessly integrate with existing deployment infrastructure. **External-Assisted Edits** methods rely on auxiliary modules for knowledge modification. Meta-learning approaches (Tan et al., 2024; Hartvigsen et al., 2023; Li et al., 2025) train hypernetworks to generate parameter updates, while memory-based methods (Hartvigsen et al., 2023; Zhang et al., 2024b; Chen et al., 2024) encode knowledge in external modules that the LLM retrieves during inference. Despite their superior performance in balancing reliability and locality, external methods introduce deployment complexity. Given these trade-offs, our work focuses on In-Place Parametric Edits, as they seamlessly integrate with existing deployment infrastructure.

Table 1: **Performance Comparison w/ and w/o Knowledge Consolidation under Real-World Evaluation on ZsRE.** (+GRPO) denotes adding knowledge consolidation stage.

	Method	Reli.	Gener.	Local.
Llama-3-8B-Instruct	Pre-Edit	2.8	2.4	38.6
	Pre-Edit(+GRPO)	5.2 (+2.4)	4.7 (+2.3)	38.4 (-0.2)
	FT-M	16.6	15.5	29.3
	FT-M(+GRPO)	62.9 (+46.3)	52.7 (+37.2)	24.9 (-4.4)
	AlphaEdit	18.7	14.0	6.3
	AlphaEdit(+GRPO)	50.4 (+31.7)	38.7 (+24.7)	5.4 (-0.9)

Table 2: **Traditional vs. Real-World Evaluation.**

Module	Traditional	Real-World
Input	raw prompt	extra content
Output Truncation	ground truth length	natural stopping
Decoding	teacher forcing	auto regressive
Metric	match ratio	LLM-as-judge

3 Revisiting Knowledge Editing

3.1 Analyzing the degradation Under Real-World Evaluation

Under traditional evaluation, many existing methods achieve near-perfect Reliability (e.g., 97–100%), yet their performance drops sharply under real-world evaluation, consistent with prior findings (Yang et al., 2025, 2026). We posit that many KE methods may successfully edit knowledge into the target layers, which accounts for the high scores under traditional evaluation, but the edited model does not learn to reliably use new knowledge. This is amplified under explicit reasoning instructions, as longer trajectories activate broader parametric knowledge, increasing the chance of reverting to pre-edit facts. We systematically study this more challenging setting in this paper (details in Appendix A.8). Under our real-world evaluation protocol, representative methods such as FT-M and AlphaEdit achieve only 16.6% and 18.7% Reliability respectively (Table 1), far below the near-perfect scores reported under traditional settings. Figure 1 (right) provides a concrete example: the edited model, when prompted to reason step by step, still reverts to pre-edit knowledge. These findings suggest that editing parametric knowledge alone is insufficient for reliable knowledge editing.

3.2 The Necessity of Post-Edit Consolidation

We argue that a post-edit *consolidation* stage is necessary for the edited model to reliably use updated parametric knowledge during autoregressive

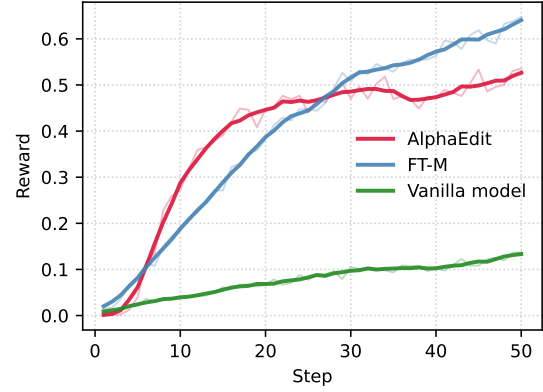


Figure 2: **Reward Curves during Consolidation.**

generation. To validate this, we apply GRPO as a trajectory-level consolidation mechanism to two representative editing methods. As shown in Table 1 and Figure 2, under real-world evaluation, post-edit consolidation substantially improves reliability on Llama-3-8B-Instruct: 16.6 $\rightarrow$ 62.9 for FT-M (+46.3) and 18.7 $\rightarrow$ 50.4 for AlphaEdit (+31.7), with generalization gains of similar magnitude. The monotonic reward increase in Figure 2 further shows that the model progressively aligns its generation behavior with the edited knowledge. Notably, applying GRPO directly to unedited models yields only marginal improvement (2.8 $\rightarrow$ 5.2), confirming that consolidation requires prior parametric editing as a foundation. These findings confirm our earlier claim and establish that *parametric knowledge updates and behavioral alignment are complementary but distinct requirements* for reliable knowledge editing.

4 EtCon

In this section, we present one concrete instantiation of the EtCon paradigm (Figure 3). The edit stage is instantiated with Targeted Proximal Supervised Fine-Tuning (TPSFT), which edits knowledge through CoT-augmented supervision under a ratio-clipped constraint (§4.1). The consolidation stage is instantiated with GRPO with a comprehensive reward to consolidate edited knowledge at the trajectory level (§4.2).

4.1 Knowledge Editing via Targeted Proximal Fine-Tuning

We instantiate the edit stage with TPSFT, a constrained variant of PSFT (Zhu et al., 2026) that restricts updates to knowledge-relevant FFN layers for better pre-trained capability preservation. This approach differs from raw PSFT that updates the whole LLMs, we restrict updates exclusively to spe-

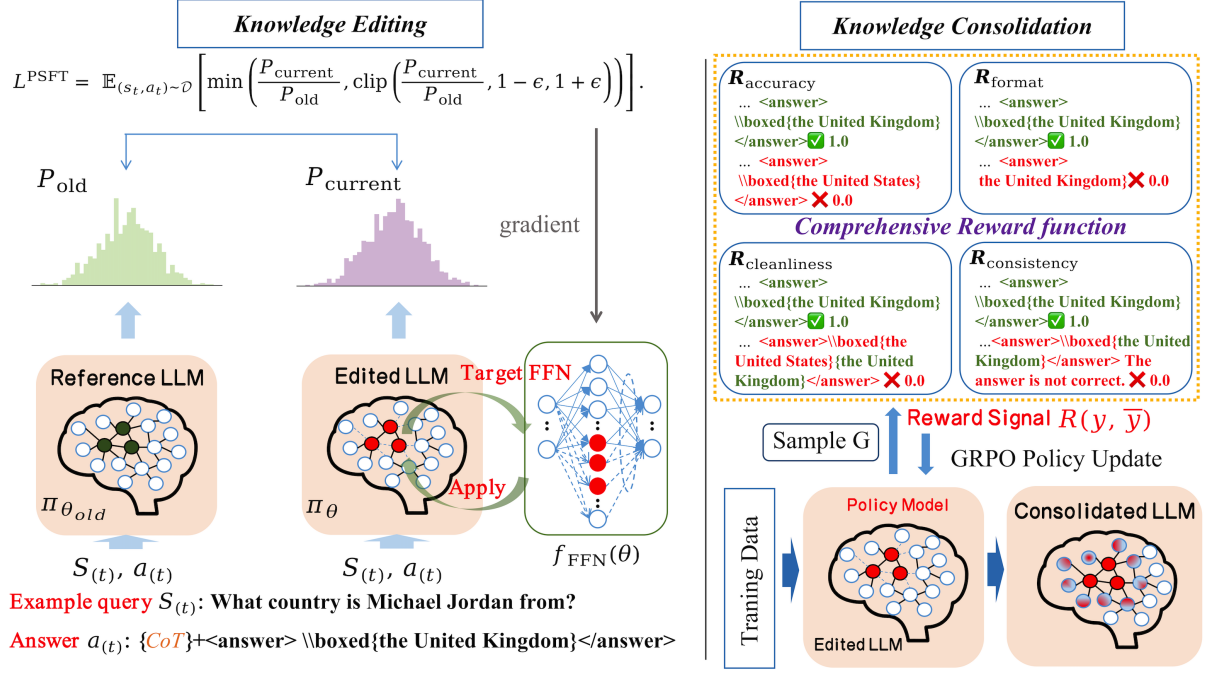


Figure 3: **Overview of the EtCon Framework.** **Edit stage** (left): We employ TPSFT to perform localized edits within the selected FFN layers to inject new knowledge. **Consolidate stage** (right): We use GRPO with a comprehensive reward function to align the parametric knowledge with actual generation behavior.

cific FFN layers, widely recognized as the primary stores of factual knowledge (Meng et al., 2022a). This targeted update strategy effectively injects new knowledge without modifying the model architecture, while minimizing disruption to pre-trained capabilities.

We consider a knowledge editing instance $(\mathbf{x}, \mathbf{y}^*)$, where $\mathbf{x}$ is the input query and $\mathbf{y}^*$ is the target answer. Let $\pi_{\theta_{\text{old}}}$ parameterized by θ_{old} denote the vanilla LLM before the current edit. We partition the *current* parameters θ into two disjoint sets: the trainable FFN parameters θ_{FFN} and the frozen parameters θ_{frozen} , such that $\theta = \theta_{\text{FFN}} \cup \theta_{\text{frozen}}$. The objective of TPSFT is to optimize θ_{FFN} to learn the updated parameters θ_{new} , yielding the post-edit policy $\pi_{\theta_{\text{new}}}$ where $\theta_{\text{new}} = \theta_{\text{FFN}}^{\text{new}} \cup \theta_{\text{frozen}}$.

A distinctive aspect of TPSFT is the use of self-generated CoT-augmented training labels. For each instance $(\mathbf{x}, \mathbf{y}^*)$, we: (1) prompt the vanilla LLM to generate a CoT reasoning path $\mathbf{c}$ for $\mathbf{x}$ (see Appendix A.7), and (2) replace the generated answer with $\mathbf{y}^*$, yielding the training target $\mathbf{y} = (\mathbf{c}, \mathbf{y}^*) = (y_1, \dots, y_L)$. This design integrates the CoT into the supervision signal. Rather than overfitting to a sharp, isolated input-output mapping, the model learns to associate the new knowledge with valid reasoning patterns. To ensure data quality, we filter out generated chains that are logically incompatible with the target (see

Appendix A.7), thereby preserving the model’s intrinsic generation capabilities while injecting the new fact.

Formally, given the editing dataset $\mathcal{D}_r$, for each token position t in the target sequence $\mathbf{y}$, we define the state $\mathbf{S}_t = (\mathbf{x}, \mathbf{y}_{<t})$ and the target token $a_t = y_t$. We optimize θ_{FFN} by minimizing the following TPSFT loss:

$$\mathcal{L}^{\text{TPSFT}}(\theta_{\text{FFN}}) = -\mathbb{E}_{(\mathbf{S}_t, a_t) \sim \mathcal{D}_r} \left[\min(r_t(\theta), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)) \right] \quad (1)$$

Here, ϵ is a hyperparameter that defines the clipping threshold. Following Zhu et al. (2026), the ratio clipping mechanism provides implicit trust-region regularization by (i) bounding the log-probability change per sample to $\log r_t \in [\log(1 - \epsilon), \log(1 + \epsilon)]$, and (ii) nullifying gradients when the ratio exceeds the upper bound. The probability ratio $r_t(\theta)$ is the core of this objective and is defined as:

$$r_t(\theta) = \frac{\pi_{\theta}(a_t | \mathbf{S}_t)}{\pi_{\theta_{\text{old}}}(a_t | \mathbf{S}_t)}, \quad (2)$$

where $\pi_{\theta}(a_t | \mathbf{S}_t)$ is the probability of generating the target token a_t given the context $\mathbf{S}_t$ from the model with **the current FFN parameters**, and $\pi_{\theta_{\text{old}}}(a_t | \mathbf{S}_t)$ is the corresponding probability from

the **reference policy**. At the start of the editing process, this reference policy is the initial vanilla LLM. For each subsequent edit instance in the sequential editing process, it is then updated to be the state of the model resulting from the immediately preceding edit.

This objective function acts as a trust-region regularizer via ratio clipping. The term $r_t(\theta)$ encourages the model to increase the likelihood of the target answer, similar to standard supervised fine-tuning. Crucially, when this ratio exceeds the threshold $1 + \epsilon$, the gradient with respect to θ_{FFN} is nullified. This design effectively removes optimization pressure from well-learned samples, constraining parametric deviation and preventing individual edits from dominating the parameter updates.

However, TPSFT alone cannot guarantee that edited knowledge is reliably expressed during autoregressive generation (§3.2), motivating the consolidation stage below.

4.2 Knowledge Consolidation via GRPO

We instantiate the consolidation stage with Group Relative Policy Optimization (GRPO) (Guo et al., 2025), which optimizes a clipped surrogate objective with group-relative advantages:

$$J_{\text{GRPO}}(\theta) = \mathbb{E} \left[\sum_{i=1}^m \min \left(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon) A_i \right) \right], \quad (3)$$

where $\rho_i = \pi_{\theta}(\hat{\mathbf{y}}^i | \mathbf{x}) / \pi_{\theta_{\text{rollout}}}(\hat{\mathbf{y}}^i | \mathbf{x})$ is the importance ratio, and $A_i = R_i - \frac{1}{m} \sum_{j=1}^m R_j$ is the group-relative advantage computed from m trajectories sampled for the same query $\mathbf{x}$.

Starting from the edited model $\pi_{\theta_{\text{new}}}$ obtained via TPSFT, we optimize the policy π_{θ} (initialized as θ_{new}) through autoregressive sampling on the editing dataset $\mathcal{D}_r$. For each query $\mathbf{x}$, the model generates m complete trajectories $\{\hat{\mathbf{y}}^1, \dots, \hat{\mathbf{y}}^m\}$, which are then scored against the target answer $\mathbf{y}^*$. The group-relative advantage formulation enables the model to learn from comparative trajectory quality without requiring absolute reward calibration. The reference policy is set to $\pi_{\theta_{\text{ref}}} = \pi_{\theta_{\text{new}}}$, ensuring that consolidation refines generation behavior while preserving the parametric edits from Stage I.

The reward function $r_{\phi}(\mathbf{x}, \mathbf{y}^*, \hat{\mathbf{y}})$ evaluates multiple aspects of the generated trajectory (as illus-

trated in Figure 3):

$$r_{\phi}(\mathbf{x}, \mathbf{y}^*, \hat{\mathbf{y}}) = w_1 R_{\text{accuracy}} + w_2 R_{\text{format}} + w_3 R_{\text{cleanliness}} + w_4 R_{\text{consistency}}, \quad (4)$$

where R_{accuracy} measures factual accuracy (whether the final answer matches $\mathbf{y}^*$), R_{format} enforces task-specific output format requirements, $R_{\text{cleanliness}}$ encourages concise outputs without extraneous tokens, and $R_{\text{consistency}}$ rewards internal reasoning coherence and alignment between intermediate steps and the final answer.

This consolidation step integrates the parametric knowledge acquired through TPSFT into the model’s inference policy. By optimizing trajectory-level generation $\pi_{\theta}(\hat{\mathbf{y}} | \mathbf{x})$ under autoregressive sampling, GRPO mitigates the knowledge-behavior mismatch, ensuring that edited knowledge is reliably elicited in actual behavior.

5 Experiments

5.1 Experiment Settings

Datasets and Models. This work utilizes 1000 samples from each of four benchmarks, ZsRE (Levy et al., 2017), COUNTERFACT (Meng et al., 2022a), MQuAKE-CF-v2 (Zhong et al., 2023) (1-edit subset), and QAEEdit (Yang et al., 2025), to evaluate the performance on knowledge editing tasks. We focus exclusively on the life-long editing paradigm with sequential edits (batch size 1), which better reflects realistic deployment scenarios where knowledge updates arrive continuously over time. We select the Llama-3-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-7B-Instruct (Qwen et al., 2025), as the base models for editing. For general ability evaluation, we use C-Eval (Huang et al., 2023), CoQA (Reddy et al., 2019), DROP (Dua et al., 2019), SQuAD 2.0 (Rajpurkar et al., 2018) and LogiQA (Liu et al., 2021).

Baselines. We compare our method against two main categories: In-Place Parametric Edits methods (FT-M (Zhang et al., 2024a), MEMIT (Meng et al., 2022b), AlphaEdit (Fang et al., 2025), MMKE (Fu et al., 2025)) and External-Assisted Edits methods (WISE (Wang et al., 2024a), GRACE (Hartvigsen et al., 2023)). We additionally compare against recent methods designed for large-scale editing, including RLEdit (Li et al., 2025), UltraEdit (Gu et al., 2025), and LocFT-BF (Yang et al., 2026). In-Place Parametric Edits methods are the main focus of this work.

Table 3: **Performance Comparison of Sequential Editing under Real-World Evaluation.** The best results in each group are in **bold**, and the second-best results are underlined. **Red numbers** in parentheses indicate the improvement over the second-best method.

		ZsRE				COUNTERFACT				QAEEdit			
	Method	Reli.	Gen.	Loc.	Avg.	Reli.	Gen.	Loc.	Avg.	Reli.	Gen.	Loc.	Avg.
Qwen2.5-7B -Instruct	Pre-edit	4.4	3.2	28.5	12.0	1.0	0.5	36.9	12.8	9.8	10.1	36.2	18.7
	FT-M	5.6	5.5	23.1	11.4	<u>3.2</u>	<u>3.1</u>	24.4	10.2	<u>14.6</u>	<u>14.5</u>	30.7	19.9
	MEMIT	0.0	0.1	0.0	0.0	0.0	0.2	0.1	0.1	0.4	0.3	0.2	0.3
	AlphaEdit	<u>15.9</u>	<u>11.5</u>	6.8	11.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WISE	4.5	3.3	19.1	9.0	1.4	1.5	<u>31.0</u>	11.3	7.1	9.7	16.9	11.2
	GRACE	4.8	4.7	28.1	<u>12.5</u>	1.9	1.2	36.8	<u>13.3</u>	14.2	13.0	36.0	<u>21.1</u>
	EtCon	69.4	60.8	<u>24.4</u>	51.5 (+39.0)	59.6	43.2	29.7	44.2 (+30.9)	75.1	63.0	<u>32.3</u>	56.8 (+35.7)
Llama-3 -8B-Instruct	Pre-edit	2.8	2.4	38.6	14.6	0.6	0.8	31.8	11.1	12.7	12.5	44.3	23.2
	FT-M	16.6	<u>15.5</u>	29.3	<u>20.5</u>	27.9	18.6	10.5	19.0	<u>34.1</u>	<u>33.2</u>	30.1	<u>32.5</u>
	MEMIT	0.1	0.1	0.0	0.1	0.3	0.7	0.4	0.5	0.2	0.7	0.0	0.3
	AlphaEdit	<u>18.7</u>	14.0	6.3	13.0	<u>61.0</u>	<u>43.8</u>	16.1	<u>40.3</u>	18.2	14.9	7.5	13.5
	WISE	4.3	3.1	2.2	3.2	1.3	0.8	<u>31.3</u>	11.1	8.1	13.3	0.9	7.4
	GRACE	5.5	3.9	37.9	15.8	2.1	0.7	32.1	11.6	15.4	13.4	34.2	21.0
	EtCon	73.5	63.1	<u>30.2</u>	55.6 (+35.1)	67.1	53.4	24.2	48.2 (+7.9)	70.7	62.7	<u>33.6</u>	55.7 (+23.2)

Implementation Details. We conduct experiments using EasyEdit (Xu et al., 2025) for evaluating various baselines, and employ the lm-evaluation-harness (Gao et al., 2024) for assessing general capabilities. More details are in Appendix A.1.

Evaluation Metrics. We evaluate our method along two principal axes: editing performance and pre-trained capability preservation. For editing performance, we adopt the LLM-as-judge framework (Yang et al., 2025; Gao et al., 2024), leveraging GPT-4.1 (OpenAI, 2025) to provide binary (correct/incorrect) semantic judgments on model outputs. Based on these judgments, we measure Reliability (edit success), Generalization (effectiveness on related inputs), and Locality (impact on unrelated inputs) on ZsRE, COUNTERFACT, and QAEEdit, and report Edit-wise accuracy and Multi-hop accuracy on MQuAKE-CF-v2 (Appendix A.9). For pre-trained capability preservation, we report Accuracy on C-Eval and LogiQA, and Exact Match (EM) and F1 on CoQA, DROP, and SQuAD 2.0. Real-world evaluation details are provided in Appendix A.8.

5.2 Main Results

Edit Efficiency. EtCon consistently achieves the best overall performance under real-world lifelong editing evaluation across benchmarks and base models. Table 3 presents comprehensive results on two LLMs. Results under TE, RE-D, and RE-CoT are reported in Table 15 in the appendix. Across all

six settings (3 benchmarks $\times$ 2 base models), EtCon attains the top Avg. score, with large margins over the strongest competing method. On Qwen2.5-7B-Instruct, EtCon reaches Avg. of 51.5/44.2/56.8 on ZsRE/COUNTERFACT/QAEEdit (improving the runner-up by +39.0/+30.9/+35.7). On Llama-3-8B-Instruct, EtCon further achieves Avg. of 55.6/48.2/55.7 (+35.1/+7.9/+23.2). These gains are primarily driven by substantially higher Reliability and Generalization (baselines with higher Locality achieve near-zero Reliability, effectively remaining unedited), indicating that EtCon can internalize new facts and generalize them to paraphrases under sequential edits without overly compromising unrelated behaviors. We attribute this robustness to our two-stage design: TPSFT performs localized, stability-aware updates, and GRPO subsequently consolidates the edited knowledge under autoregressive trajectories.

6 Analysis

6.1 Effectiveness of Consolidation Stage

Overall Performance of EtCon. As shown in Table 4, EtCon achieves the best trade-off between editing effectiveness and general capability preservation. EtCon attains 75.1% Reliability and 63.0% Generalization, substantially outperforming all baselines. Meanwhile, EtCon maintains general capabilities close to the original model (C-Eval: 78.5 vs 79.5, CoQA F1: 69.4 vs 70.1), demonstrating that our two-stage design successfully bal-

Table 4: **Comprehensive Evaluation of Editing Effectiveness and Pre-Trained Capability Preservation.** Comparison on Qwen2.5-7B-Instruct across editing metrics and five general benchmarks.

Method	Editing Performance				General Capabilities								
	Reli.↑	Gen.↑	Loc.↑	Avg.	C-Eval	CoQA		DROP		SQuAD		LogiQA	Avg.
					Acc.	EM	F1	EM	F1	EM	F1	Acc.	
Pre	9.8	10.1	36.2	18.7	79.5	54.5	70.1	2.2	9.9	9.9	18.9	38.7	35.5
FT-M	14.6	14.5	30.7	19.9	75.9	21.3	38.7	2.4	13.3	2.9	11.2	37.0	25.3
FT-M+Con.	<u>42.3</u>	<u>34.1</u>	31.9	<u>36.1</u>	77.0	26.2	46.6	2.8	14.6	4.4	13.5	37.8	27.9
MMKE	12.2	10.4	34.2	18.9	79.3	60.3	74.6	10.3	24.3	<u>13.8</u>	<u>21.1</u>	41.0	40.6
MMKE+Con.	37.2	31.4	31.0	33.2	78.8	<u>59.1</u>	<u>73.3</u>	<u>8.5</u>	<u>21.6</u>	12.1	19.6	<u>39.2</u>	<u>39.0</u>
ALPHA	0.0	0.0	0.0	0.0	23.0	0.0	0.0	0.0	0.0	50.1	50.1	21.8	18.1
ALPHA+Con.	0.0	0.0	0.0	0.0	23.0	0.0	0.0	0.0	0.0	50.1	50.1	21.8	18.1
TPSFT	16.8	13.7	30.1	20.2	<u>78.9</u>	56.2	71.2	2.4	12.6	10.2	20.8	38.9	36.4
TPSFT+Con.	75.1	63.0	<u>32.3</u>	56.8	78.5	55.1	69.4	2.5	8.6	9.9	19.6	38.4	35.3

Table 5: **Performance Comparison on Multi-Hop Tasks (MQuAKE-CF-v2).** The Base model is evaluated on unedited answers; others on edited answers.

Methods	Edit-wise	2-hop	3-hop	4-hop
Base	-	69.1	36.2	85.4
FT-M	19.1	2.7	3.8	4.2
ALPHA	<u>20.4</u>	<u>13.9</u>	<u>11.3</u>	9.5
MMKE	9.3	12.0	7.1	<u>22.9</u>
MEMIT	0.4	1.1	1.3	0.0
EtCon	54.1	24.2	14.6	33.3

ances knowledge injection with pre-trained capability preservation.

Performance on Multi-hop Task. Table 5 further demonstrates EtCon’s advantage on MQuAKE-CF-v2. EtCon achieves 24.2% and 33.3% accuracy on 2-hop and 4-hop queries respectively, substantially outperforming FT-M. These results suggest that consolidation may help the model use localized edits during multi-step inference, although the absolute performance remains limited. Incorporating explicit multi-hop supervision or step-level rewards during consolidation may help close this gap.

Universality of Knowledge Consolidation. To evaluate the generality of our knowledge consolidation stage, we augment three baselines (FT-M, MMKE, and AlphaEdit) with Consolidation on QAEdit. As shown in Table 4, adding Consolidation to FT-M and MMKE improves Reliability by 25% to 28% and Generalization by about 20%. Similarly, Consolidation lifts TPSFT from 16.8% to 75.1% Reliability and from 13.7% to 63.0% Generalization. These results suggest that Consolidation bridges the gap between edited parametric knowledge and autoregressive generation behavior, enabling more reliable knowledge utilization in practice.

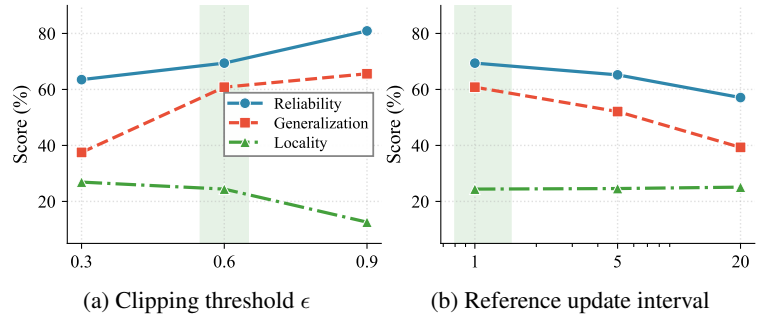


Figure 4: **Sensitivity Analysis of Key Hyperparameters.**

Preservation of Pre-trained Capabilities. As shown in Table 4, TPSFT alone preserves general capabilities close to the pre-edit model (GC-Avg 36.4 vs. 35.5), and the subsequent Consolidation stage incurs only minor degradation (GC-Avg 35.3). However, it cannot recover capabilities lost during editing: once AlphaEdit collapses, the knowledge consolidation stage cannot restore it. The anomalously high SQuAD score of collapsed AlphaEdit is an artifact of degenerate repetition that happens to match the evaluation format. This indicates that reliable lifelong editing requires both lossless knowledge injection and post-edit consolidation. Future work under the EtCon paradigm should prioritize lossless knowledge editing.

6.2 Ablation Study

Edit Stage Ablation. We compare TPSFT with standard SFT in Table 6. The results show that neither method alone yields reliably editable behavior, as evidenced by consistently low Reliability and Generalization. Nevertheless, TPSFT offers a clear advantage in capability preservation, substantially reducing the degradation observed with SFT.

Consolidation Stage Ablation. We ablate key components of our consolidation reward (Table 6).

Table 6: **Ablation Study of the Key Components in EtCon.** Experiments on COUNTERFACT with Llama-3-8B-Instruct. GC-Avg denotes the mean of general-capability benchmarks (C-Eval (Acc.), CoQA (F1), and SQuAD 2.0 (F1)). Con. denotes Consolidation.

Stage	Methods	Reli.	Gen.	Loc.	GC-Avg
Base	-	0.6	0.8	31.8	52.85
Edit.	w/ SFT	1.4	0.3	30.7	50.31
	w/ TPSFT	3.3	1.8	30.2	52.40
Con. (after TPSFT)	w/o $R_{\text{cleanliness}}$	56.1	22.4	24.7	50.17
	w/o $R_{\text{consistency}}$	51.6	27.2	25.1	49.75
	Complete	67.1	53.4	24.2	51.25

Table 7: **Efficiency of Target Edit.** Comparison of full-parameter vs. target edit under TPSFT on QAEEdit with Qwen2.5-7B-Instruct.

Edit Scope	Reli.	Gen.	Loc.	GC-Avg
Full Param.	47.3	41.5	35.8	58.17
Target edit	75.1	63.0	32.3	55.83

Removing $R_{\text{cleanliness}}$ causes a substantial Generalization drop ($53.4 \rightarrow 22.4$), as the model emits extraneous content (e.g., both old and new facts) to inflate scores. Removing $R_{\text{consistency}}$ leads to a larger Reliability drop ($67.1 \rightarrow 51.6$), often producing self-contradictions where the model states the correct answer then immediately reverts. These ablations suggest that both reward components help mitigate reward hacking and improve output coherence; see Appendix A.10 for case studies.

Parameter Sensitivity. Figure 4 analyzes TPSFT hyper-parameters. Clipping threshold ϵ : Increasing ϵ improves Reliability and Generalization but reduces Locality, with a sharp drop at $\epsilon = 0.9$. Thus, $\epsilon = 0.6$ provides the best trade-off, as larger ϵ allows stronger updates but increases policy drift. Reference update interval: Larger intervals consistently reduce Reliability and Generalization, while Locality remains nearly unchanged. Frequent updates (Interval= 1) perform best because stale references increase the policy gap and over-activate clipping, limiting effective updates.

Effect of CoT Augmentation. Figure 5 compares TPSFT with and without CoT-augmented training labels. Without CoT augmentation, Reliability is slightly higher, but Locality drops sharply and general capabilities degrade. This is because supervising only on the few target tokens leads to overfitting. CoT augmentation encourages the model to learn valid reasoning patterns rather than overfitting to isolated input-output mappings, thereby better preserving general capabilities.

Efficiency of Target Edit. We compare TARGET

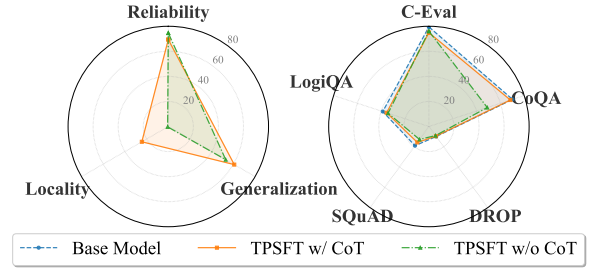


Figure 5: **Effect of CoT Augmentation in EtCon Edit Stage.** Comparison of EtCon w/ and w/o CoT-augmented training labels during editing stage.

EDIT, which updates a small set of FFN weights, with FULL PARAM update under the same TPSFT setup. As shown in Table 7, TARGET EDIT yields substantially higher Reliability and Generalization, while incurring only a slight drop in general capability (GC-Avg). We attribute this gap to reduced optimization efficiency under ratio clipping. With limited update steps per edit, FULL PARAM spreads gradients across the whole model, weakening the learning signal on the edited fact. In addition, clipping may saturate early for many tokens, further shrinking effective updates. Overall, restricting TPSFT provides a more controllable and effective mechanism for knowledge editing.

6.3 Large-Scale Batch Editing under Three Evaluation Protocols

To evaluate EtCon at a larger editing scale and in the batch-editing setting, we conduct 20K edits on ZsRE with a batch size of 100. As shown in Table 8, EtCon maintains strong performance on both base models across all three evaluation protocols, with detailed input and scoring procedures provided in Appendix A.8. While RLEdit, UltraEdit, and LocFT-BF perform well under teacher-forced evaluation, their Reliability and Generalization degrade substantially under realistic evaluation, especially with CoT generation. In contrast, EtCon achieves 75.0% and 79.3% Reliability under RE-CoT on Qwen2.5-7B-Instruct and Llama-3-8B-Instruct respectively, with corresponding Generalization scores of 67.3% and 69.6%. These results demonstrate that EtCon scales effectively to large-scale batch editing and reliably expresses edited knowledge during autoregressive generation.

6.4 Additional Analysis

More analysis. We present additional analysis in the appendix, including 1) performance evolution during sequential editing (Appendix A.2.1),

Table 8: **Performance under Large-Scale Batch Editing.** Results on ZsRE after 20K edits with a batch size of 100 under teacher-forced evaluation (TE), realistic direct-answer evaluation (RE-D), and realistic CoT evaluation (RE-CoT).

Base Model	Method	TE			RE-D			RE-CoT		
		Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.
Qwen2.5-7B-Instruct	RLEdit	84.3	78.8	34.2	38.5	27.9	7.4	7.7	6.8	7.1
	UltraEdit	80.1	77.9	<u>39.1</u>	17.6	16.7	19.0	4.8	4.5	<u>19.6</u>
	LocFT-BF	<u>98.4</u>	<u>83.3</u>	40.2	<u>97.0</u>	<u>42.3</u>	<u>24.2</u>	<u>48.1</u>	<u>37.0</u>	12.4
	EtCon	99.1	90.2	38.4	98.6	76.6	32.9	75.0	67.3	31.8
Llama-3-8B-Instruct	RLEdit	90.1	88.9	37.8	18.2	17.4	8.5	2.9	2.8	4.1
	UltraEdit	87.6	86.0	42.4	16.2	11.1	<u>26.1</u>	1.3	1.5	6.6
	LocFT-BF	<u>98.6</u>	83.1	27.5	<u>96.7</u>	<u>55.3</u>	19.8	<u>42.2</u>	<u>35.5</u>	<u>9.6</u>
	EtCon	98.9	<u>88.4</u>	<u>40.5</u>	97.1	80.6	27.4	79.3	69.6	28.1

2) edit-time efficiency of EtCon (Appendix A.4), 3) the effect of edited layers on EtCon (Appendix A.2.2), 4) evaluation on reasoning-oriented model: DeepSeek-R1-Distill-Qwen-7B (Appendix A.3), 5) cross-model portability on ZsRE (Appendix A.2.3), 6) reward weight sensitivity (Appendix A.5), and 7) robustness to context-guided input (Appendix A.6).

7 Conclusion

In this work, we propose EtCon, an edit-then-consolidate framework for reliable knowledge editing. In Edit Stage, TPSFT performs constrained edits to inject knowledge while preserving pre-trained capabilities. In the Consolidate Stage, GRPO consolidates edits by aligning autoregressive generation with updated facts. Extensive experiments demonstrate that EtCon achieves superior editing reliability and generalization while preserving pre-trained capabilities in real-world editing scenarios.

Limitations

EtCon introduces a post-edit consolidation stage that requires trajectory-level sampling and on-policy optimization, adding computational overhead beyond the editing stage itself. Although consolidation is performed only once after all edits complete rather than per-edit, it still involves multiple forward and backward passes over the edited knowledge set, which can be non-trivial in time-sensitive deployment scenarios. Reducing this cost through more sample-efficient policy optimization algorithms or amortized consolidation across edit batches is a promising future direction. Moreover, our current reward design targets single-hop setting, while multi-hop settings may require denser,

step-level and generalization-aware rewards that explicitly guide the composition of edited facts across reasoning steps.

Ethical Considerations

Knowledge editing techniques, including EtCon, could potentially be misused to inject harmful or misleading information into language models. We emphasize that all experiments in this work use publicly available benchmarks and do not involve sensitive or harmful content. We encourage practitioners to apply knowledge editing responsibly, with appropriate verification of edit content, and to consider potential societal impacts when deploying edited models in real-world applications.

References

- Qizhou Chen, Taolin Zhang, Xiaofeng He, Dongyang Li, Chengyu Wang, Longtao Huang, and Hui Xue*. 2024. [Lifelong knowledge editing for LLMs with retrieval-augmented continuous prompt learning](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13565–13580, Miami, Florida, USA. Association for Computational Linguistics.
- Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. 2025. Namet: robust massive model editing via noise-aware memory optimization. *arXiv e-prints*, pages arXiv–2505.
- Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. 2024. Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019.

- Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.
- 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*, volume 2025, pages 16366–16396.
- Zichuan Fu, Xian Wu, Guojing Li, Yingying Zhang, Yefeng Zheng, Tianshi Ming, Yejing Wang, Wanyu Wang, and Xiangyu Zhao. 2025. Model merging for knowledge editing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 433–443.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. [The language model evaluation harness](#).
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). Preprint, arXiv:2407.21783.
- Xiaojie Gu, Ziyang Huang, Jia-Chen Gu, and Kai Zhang. 2025. Ultraedit: Training-, subject-, and memory-free lifelong editing in language models. *arXiv preprint arXiv:2505.14679*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. 2024. Rebuilding rome: Resolving model collapse during sequential model editing. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 21738–21744.
- Akshat Gupta, Maochuan Lu, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025a. Efficient knowledge editing via minimal precomputation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 829–840.
- Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, A Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. 2025b. Lifelong sequential knowledge editing without model degradation. *arXiv preprint arXiv:2502.01636*.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959.
- Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi lei, Yao Fu, Maosong Sun, and Junxian He. 2023. [C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 62991–63010. Curran Associates, Inc.
- 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.
- Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 18564–18572.
- Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. 2025. Reinforced lifelong editing for language models. In *Proceedings of the 42nd International Conference on Machine Learning, ICML’25*. JMLR.org.
- Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2021. Logiqa: a challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI’20*.
- Mingda Liu, Zhenghan Zhu, Ze’an Miao, and Katsuki Fujisawa. 2026. [Norm anchors make model edits last](#). Preprint, arXiv:2602.02543.
- 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. *arXiv preprint arXiv:2502.00602*.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.
- OpenAI. 2025. [Introducing GPT-4.1 in the API](#).
- Haewon Park, Gyubin Choi, Minjun Kim, and Yohan Jo. 2025. [Context-robust knowledge editing for language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10360–10385, Vienna, Austria. Association for Computational Linguistics.
- Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. 2025. In-context editing: Learning knowledge from self-induced distributions. In *International Conference on Learning Representations*, volume 2025, pages 77563–77585.
- Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. [Qwen2.5 technical report](#). *Preprint*, arXiv:2412.15115.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.
- Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.
- Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. 2024. Knowledge editing in language models via adapted direct preference optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4761–4774.
- Chenmian Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*, volume 2024, pages 54676–54690.
- Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Hua-jun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.
- Yangfan Wang, Tianyang Sun, Chen Tang, Jie Liu, Wei Cai, and Jingchi Jiang. 2026. [HiEdit: Lifelong model editing with hierarchical reinforcement learning](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 39924–39942, San Diego, California, United States. Association for Computational Linguistics.
- Yiwei Wang, Muhao Chen, Nanyun Peng, and Kai-Wei Chang. 2024b. Deepedit: Knowledge editing as decoding with constraints. *arXiv preprint arXiv:2401.10471*.
- Ziwen Xu, Shuxun Wang, Kewei Xu, Haoming Xu, Mengru Wang, Xinle Deng, Yunzhi Yao, Guozhou Zheng, Huajun Chen, and Ningyu Zhang. 2025. Easyedit2: An easy-to-use steering framework for editing large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 522–535.
- Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. 2025. The mirage of model editing: Revisiting evaluation in the wild. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15336–15354.
- Wanli Yang, Rui Tang, Hongyu Zang, Du Su, Qi Cao, Jingang Wang, Huawei Shen, Xueqi Cheng, and Fei Sun. 2026. Fine-tuning done right in model editing. In *International Conference on Learning Representations*, volume 2026, pages 71683–71702.
- 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. 2024a. [A comprehensive study of knowledge editing for large language models](#). *Preprint*, arXiv:2401.01286.
- Taolin Zhang, Qizhou Chen, Dongyang Li, Chengyu Wang, Xiaofeng He, Longtao Huang, Hui Xue’, and Jun Huang. 2024b. [DAFNet: Dynamic auxiliary fusion for sequential model editing in large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1588–1602, Bangkok, Thailand. Association for Computational Linguistics.
- Taolin Zhang, Haidong Kang, Dongyang Li, Qizhou Chen, Xiaofeng He, Chengyu Wang, and Richang Hong. 2026. [QueueEDIT: Structural self-correction for sequential model editing in LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 4706–4719, San Diego, California, United States. Association for Computational Linguistics.
- Tianyu Zhang, Junfeng Fang, Houcheng Jiang, Baolong Bi, Xiang Wang, and Xiangnan He. 2025. Explainable and efficient editing for large language models.

In *Proceedings of the ACM on Web Conference 2025*, pages 1963–1976.

Zhuoran Zhang, Yongxiang Li, Zijian Kan, Keyuan Cheng, Lijie Hu, and Di Wang. 2024c. Locate-then-edit for multi-hop factual recall under knowledge editing. *arXiv preprint arXiv:2410.06331*.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. 2025. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8):1–35.

Haitian Zhong, Yuhuan Liu, Ziyang Xu, Guofan Liu, Qiang Liu, Shu Wu, Zhe Zhao, Liang Wang, and Tieniu Tan. 2025. React: Representation extraction and controllable tuning to overcome overfitting in llm knowledge editing. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16994–17011.

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

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Wenhong Zhu, Ruobing Xie, Rui Wang, Samm Sun, Di Wang, and Pengfei Liu. 2026. Proximal supervised fine-tuning. In *International Conference on Learning Representations*, volume 2026, pages 134993–135015.

A Appendix

A.1 Additional Implementation Details

Baseline Configuration. For our baseline experiments, we utilize the EasyEdit framework. All hyperparameters adhere to the default configurations of the respective comparison methods, with further details provided in (Yang et al., 2025; Qi et al., 2025).

Editing Stage. For our proposed EtCon method, we update only the FFN down-projection layers (mlp.down_proj) in layers 7–11 of Llama-3-8B-Instruct and layers 5–9 of Qwen2.5-7B-Instruct, following prior works (Zhang et al., 2024a; Meng et al., 2022a; Geva et al., 2021; Qi et al., 2025) on knowledge editing. Following the lifelong editing paradigm, we use AdamW with learning rate 1×10^{-5} and set $\epsilon = 0.6$ of TPSFT, sequentially performing edits (batch size 1) on the same model. TPSFT is trained for 5 epochs with at most 6 update steps per edit using early stopping.

Consolidation Stage. We optimize the inference-time policy with GRPO. The comprehensive reward function in Equation (4) uses the following weight coefficients: $w_1 = 0.7$ for R_{accuracy} , $w_2 = 0.05$ for R_{format} , $w_3 = 0.15$ for $R_{\text{cleanliness}}$, and $w_4 = 0.1$ for $R_{\text{consistency}}$. These weights were determined through extensive empirical experiments to balance factual accuracy with output quality. All reward components are binary (0/1), with R_{accuracy} computed via exact string matching between the extracted answer and the ground-truth after normalization. All specific hyperparameters are available in Table 9.

A.2 Additional Results

A.2.1 Performance under Larger-Scale Lifelong Editing

Scaling to Longer Editing. Figure 6 shows performance evolution when scaling to 3,000 sequential edits on ZsRE. EtCon maintains stable performance: Reliability and Generalization decrease moderately from 73.5% to 58.2% and 63.1% to 48.7% respectively, while Locality remains within a narrow band (28%–32%). In contrast, FT-M degrades rapidly, with Reliability dropping from 16.6% to near-zero. These results demonstrate that EtCon remains stable even under longer editing sequences.

Table 9: Hyperparameters for the GRPO Consolidation Stage.

Configuration	Value
<i>Model Configuration</i>	
Precision	BFloat16
Max Prompt Length	2k
Max Response Length	2k
<i>Training Hyperparameters</i>	
Learning Rate	1.0×10^{-6}
Optimizer	AdamW
Global Batch Size	64
Rollout Batch Size	256
Micro Batch Size (Update)	4
Micro Batch Size (Experience)	16
Training Step	100
Gradient Clipping	1.0
<i>Rollout Configuration</i>	
Number of Rollouts (m)	8
Temperature	1.0
Top-p	0.99
<i>Infrastructure</i>	
GPUs	8 × H800
Tensor Parallelism	1
FSDP	Enabled
CPU Offloading	Disabled
Gradient Checkpointing	Enabled
<i>Validation</i>	
Validation Batch Size	512
Validation Frequency	Every 5 steps
Validation before Training	Yes

A.2.2 Effect of Edited Layers on EtCon

Experimental Setup. We further investigate the impact of editing different FFN layers on EtCon’s performance through an ablation study, as shown in Table 10 and Figure 7.

Main Finding. We observe that under identical hyperparameter settings, editing early layers (Layers 7-11) outperforms deeper layers (Layers 17-21) in both locality and generalization.

Reward Hacking Phenomenon. Analyzing the “high reward, low performance” phenomenon in deeper layers shown in Figure 7, we find that editing deeper layers is more prone to triggering Reward Hacking.

Mechanistic Interpretation. Existing mechanistic interpretability research suggests that shallow layers mainly store factual knowledge while deeper layers handle information integration and reason-

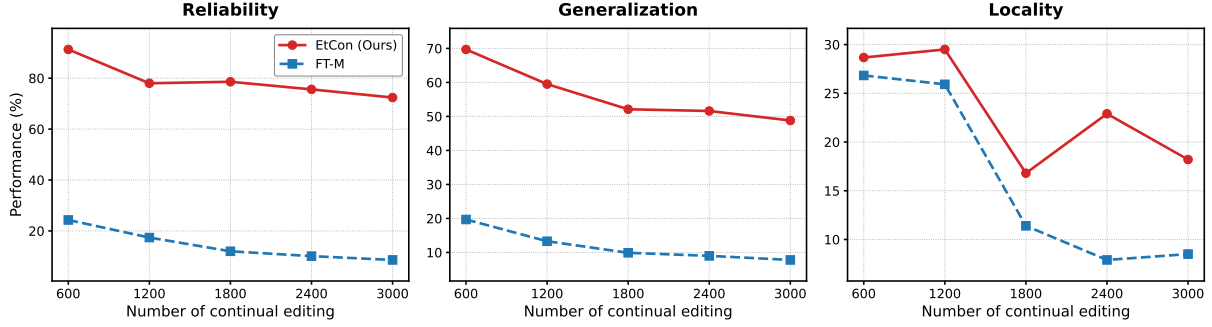


Figure 6: Performance Evolution under Larger-Scale Lifelong Editing.

Table 10: Layer Selection Ablation on Llama-3-8B-Instruct.

Layers	Reli.	Gener.	Local.
7-8-9-10-11	73.5	63.1	30.2
12-13-14-15-16	78.1	58.3	24.1
17-18-19-20-21	76.7	53.2	17.3

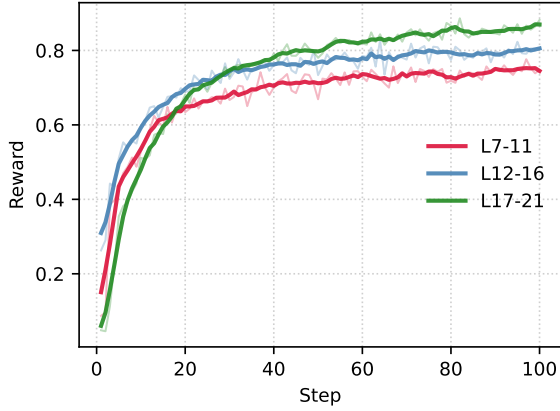


Figure 7: Reward Convergence across Layer Configurations.

ing. Based on this, we posit that the observed phenomenon stems from a misalignment between retained knowledge and updated information.

Cognitive Conflict Analysis. Editing only deeper layers may cause retained knowledge in shallow layers to conflict with the injected knowledge in deeper layers. Faced with this cognitive conflict, the LLM likely adopts a speculative strategy to maximize rewards. This reward hacking leads to internal model confusion, thereby causing a decline in performance.

A.2.3 Portability Evaluation

Superior Cross-Architecture Portability. As shown in the Figure 8, EtCon demonstrates superior portability across architectures, achieving

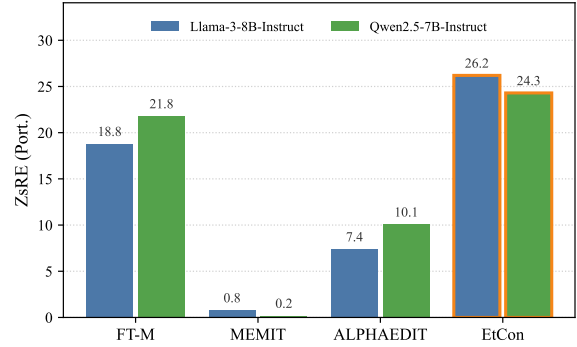


Figure 8: Portability Evaluation on ZsRE. Comparison across Llama-3-8B-Instruct and Qwen2.5-7B-Instruct.

26.2% on Llama-3 and 24.3% on Qwen2.5, consistently outperforming the strongest baseline FT-M (18.8% and 21.8%).

Consolidation Effectiveness. This validates that the consolidation stage effectively enables the model to utilize edited facts for reasoning.

Limitations and Future Directions. However, the absolute gains remain limited compared to single-hop reliability. We attribute this to EtCon being trained solely on single-hop QA without explicit supervision for multi-hop compositionality; thus, incorporating such supervision is a natural next step to further enhance portability.

A.3 Evaluation on Reasoning-Oriented Architectures

Experimental Setup. To assess the robustness of our method on architectures with inherent reasoning capabilities, we extended our evaluation to DeepSeek-R1-Distill-Qwen-7B using 1,000 samples from the ZsRE. Following our main experiments, we target the MLP down-projection weights (`mlp.down_proj`) for knowledge editing.

Shallow Layers Yield Best Trade-off. As reported in Table 11, EtCon maintains high editing efficacy on this reasoning-oriented model. In

Table 11: **Layer Selection Ablation on DeepSeek-R1-Distill-Qwen-7B.**

Layers	Reli.	Gener.	Local.
5-6-7-8-9	88.6	53.5	17.0
13-14-15-16-17	79.6	42.6	16.1
23-24-25-26-27	83.0	52.9	7.3

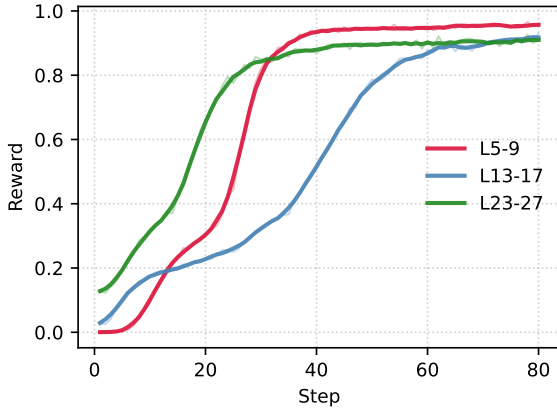


Figure 9: **Consolidation Dynamics on Reasoning-Oriented Model.**

particular, editing the shallow layers (Layers 5–9) yields the best trade-off, achieving 88.6% Reliability and 53.5% Generalization while preserving acceptable Locality (17.0%). This suggests that our Edit-then-Consolidate paradigm is compatible with the model’s intrinsic reasoning processes rather than disrupting them.

Deep-Layer Shortcut Phenomenon. The green curve exhibits the fastest initial convergence (Steps 0-20). Since these layers are proximal to the output projection, the model can quickly maximize the reward by establishing a direct mapping to the target answer. However, similar to the phenomenon observed in Appendix A.2.2, this configuration plateaus at a lower reliability level. This suggests that modifying only the deep layers could lead the model to adopt a "shortcut" strategy which may leave the model vulnerable to the cognitive conflict between retained shallow knowledge and updated deep injections.

Shallow-Layer Constructive Consolidation. In contrast, the red curve displays a distinct "warm-up" phase (Steps 0-15) followed by a sustained ascent to the highest performance plateau. This trajectory indicates that injecting knowledge into the shallow layers requires more optimization steps for the consolidate stage to propagate the changes through autoregressive generation pipeline. Cru-

cially, this delay reflects a constructive consolidation process: the model is realigning its actual generation behavior with the updated parametric knowledge base. This results in convergence stability and reliability, demonstrating that our Edit-then-Consolidate paradigm is compatible with the intrinsic reasoning mechanisms of complex architectures.

A.4 Time Efficiency Analysis

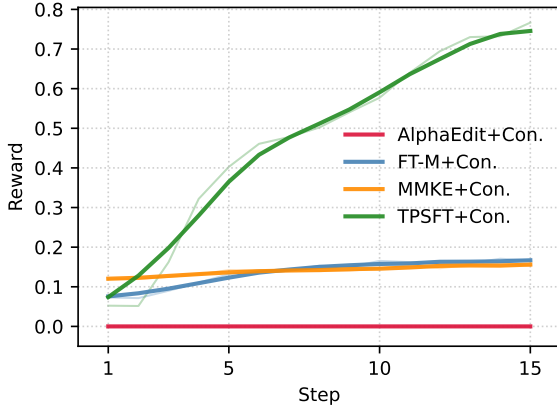
In our earlier analysis, we argued that reliable knowledge editing in realistic settings requires both a knowledge editing stage and a knowledge-consolidation stage, and that any speed advantage should be predicated on reliability. Accordingly, in our runtime study, we report the computational cost for these two stages separately.

Editing stage. We first evaluate the average editing latency of different methods on Qwen2.5-7B-Instruct using the QAEdit dataset. For each method, we perform a sequence of 100 single-instance edits and measure the wall-clock time required to complete each individual edit. All methods are run under the same configuration. The results are summarized in Table 12. TPSFT attains an average editing time of 6.01 seconds, which is comparable to that of AlphaEdit (7.39 s) and MEMIT (7.78 s). GRACE (3.02 s) and WISE (2.68 s) exhibit lower latency, while FT-M yields the lowest runtime (0.61 s). Overall, TPSFT incurs a per-edit cost that is on par with representative parameter-editing baselines, indicating that its computational overhead at this stage remains moderate and compatible with practical deployment.

Consolidation stage. Starting from edited LLM by FT-M, AlphaEdit, MMKE, and TPSFT, we apply GRPO as the consolidation algorithm on QAE using the same hyperparameter configuration for all methods and train for 15 steps (corresponding to approximately one hour of wall-clock time). Figure 10 plots the comprehensive reward curves. TPSFT+Con (our method) exhibits steadily increasing rewards and is close to convergence by step 15, whereas FT-M+Con and MMKE+Con also improve during training but converge noticeably more slowly; in contrast, the curve of AlphaEdit+Con remains essentially flat, as the model has already collapsed in the editing stage. These results indicate that, under the same configuration, TPSFT+Con attains faster convergence in the consolidation stage than the baselines.

Table 12: **Time Efficiency: Average Editing Latency per Instance.**

Methods	TPSFT (Ours)	AlphaEdit	MEMIT	GRACE	WISE	FT-M
Avg. Time / Edit	6.01s	7.39s	7.78s	3.02s	2.68s	0.61s

Figure 10: **Convergence Speed Comparison during Consolidation.**

A.5 Reward Weight Sensitivity Analysis

We analyze reward weight sensitivity on COUNTERFACT with Llama-3-8B-Instruct (Table 13). Two design principles hold: (1) accuracy dominance ($w_{\text{acc}} > \sum_{i \neq \text{acc}} w_i$) is necessary for effective consolidation, and (2) given accuracy dominance, auxiliary reward weights are robust to exact values. Violating the first principle leads to sharp degradation.

Table 13: **Reward Weight Sensitivity.** Performance under different weight configurations. All experiments use the same four reward components.

R_{accuracy}	R_{format}	$R_{\text{cleanliness}}$	$R_{\text{consistency}}$	Reli.	Gen.
0.70	0.05	0.15	0.10	67.1	53.4
0.60	0.05	0.20	0.15	64.8	49.1
0.80	0.05	0.12	0.08	66.3	51.7
0.45	0.05	0.30	0.20	42.6	20.8

A.6 Effect of Context-Guided Input

Table 14 compares methods with and without context-guided input on ZsRE (Qwen2.5-7B-Instruct, 1000 sequential edits). WISE and GRACE are highly sensitive to context-guided prompting: GRACE Reliability drops from 76.8 to 4.8 when context is added, and WISE Reliability drops from 17.2 to 4.5. In contrast, EtCon remains stable across both settings (Reliability 72.5 vs. 69.4, Generalization 61.2 vs. 60.8). EtCon remains stable

regardless of input format, indicating that consolidation internalizes edited knowledge into robust generation behavior.

Table 14: **Effect of Context-Guided Input.** Comparison on ZsRE with Qwen2.5-7B-Instruct. “w/” denotes context-guided input (“Please reason step by step, then answer the question”); “w/o” denotes context-free input.

Method	Input	Reli.	Gen.	Loc.	Avg.
EtCon	w/	69.4	60.8	24.4	51.5
	w/o	72.5	61.2	23.7	52.5
WISE	w/	4.5	3.3	19.1	9.0
	w/o	17.2	4.7	20.2	14.0
GRACE	w/	4.8	4.7	28.1	12.5
	w/o	76.8	3.1	27.9	35.9

A.7 CoT Generation and Processing

To generate Chain-of-Thought reasoning paths during TPSFT training, we employ the following prompt template (Figure 11) that encourages natural reasoning while ensuring convergence to target answers.

Design Rationale: This prompt template serves three critical functions in our TPSFT implementation:

(1) **Natural Reasoning Preservation:** By requesting step-by-step analysis without explicitly mentioning knowledge updates, the model generates reasoning paths consistent with its pre-trained style.

(2) **Target Alignment:** Providing the target answer guides the generation toward correct conclusions while allowing flexibility in reasoning approaches.

(3) **Structured Output:** The answer tag format ensures clean extraction and replacement during training data preparation, while the reasoning portion provides the smooth distribution over trajectories discussed in Section 4.1.

After generation, we enclose the reasoning path within `<think>...</think>` tags to explicitly demarcate the thought process, and overwrite the content inside the `<answer>...</answer>` tags with the verified new target fact, yielding training labels

Table 16: **Comparison of Teacher-Forced Evaluation, Realistic Direct-Answer Evaluation, and Realistic CoT Evaluation.**

Evaluation Protocol	Input Prompt	Generation Mode	Scoring
Teacher-Forced Evaluation (TE)	{question}	Teacher forcing	Token-level accuracy
Realistic Direct-Answer Evaluation (RE-D)	Please answer the question: {question}	Autoregressive generation	LLM-as-a-Judge
Realistic CoT Evaluation (RE-CoT)	Please reason step by step, then answer {question}.	Autoregressive generation	LLM-as-a-Judge

that combine natural reasoning patterns with an exact gold answer. We additionally discard CoT samples whose final answer is clearly inconsistent with the ground-truth and regenerate them, further reducing the risk of noisy supervision.

A.8 Real-World Evaluation Details

The details and distinctions among teacher-forced evaluation (TE), realistic direct-answer evaluation (RE-D) (Yang et al., 2025), and realistic CoT evaluation (RE-CoT) are summarized in Table 16.

We use RE-CoT as our primary realistic evaluation setting. Its evaluation process consists of three key stages:

(1) **Input:** To assess the model’s ability to deeply integrate and apply new knowledge, our inputs include both factual questions and instructions that require multi-step reasoning. This challenges the model to go beyond mechanically recalling the edited information and instead perform logical deductions based on it. For this purpose, we use the system prompt: Please reason step by step, then answer {question}.

(2) **Output:** For the edited model output, we use the model’s complete auto-regressive generation as the object of evaluation, up to its predefined stop token. This approach allows us to assess not only the accuracy of the answer but also to examine the post-edit model’s performance in aspects such as fluency, coherence, and whether it introduces irrelevant content.

(3) **LLM-as-a-Judge Evaluation:** To achieve a scalable and objective evaluation, we introduce a more powerful Large Language Model (LLM) to act as a “judge.” This judge model makes its decision by comprehensively considering the original question, the ground-truth answer (Target), and the full answer content from the edited model, ultimately providing a binary (correct/incorrect) judgment. The full judge prompts are provided in Figure 12 and Figure 13.

A.9 Metrics for MQuAKE-CF-v2

For multi-hop task evaluation, we sample 1000 instances from MQuAKE-CF-v2 proportionally across 2-hop, 3-hop, and 4-hop questions. We report the following metrics:

- **Edit-wise** accuracy: measures whether the model correctly answers direct queries about edited facts.
- **Multi-hop** accuracy (2/3/4-hop): evaluates whether the model can perform multi-step reasoning that involves the edited knowledge.

A.10 Analysis of Reward Hacking Cases

Analysis of Reward Hacking Patterns: The two cases in Figures 14 and 15 reveal distinct failure modes in the absence of proper reward design. In Figure 14, the model exhibits “self-correction” behavior, correctly reasoning through the problem but then artificially inserting the target answer followed by an immediate correction. This pattern emerges when $R_{\text{consistency}}$ is absent, as the model attempts to maximize accuracy rewards without maintaining logical coherence. Figure 15 demonstrates “answer hedging” where the model provides multiple answers to maximize the probability of including the correct one. This occurs without $R_{\text{cleanliness}}$, as there is no penalty for extraneous content. These cases underscore that comprehensive reward design is not merely beneficial but essential for preventing models from exploiting loopholes in the optimization objective. The 11.0% and 15.5% performance drops observed when removing these reward components (Table 6) quantitatively confirm their critical role in maintaining robust consolidation.

Table 15: **Sequential Editing Performance under Three Evaluation Protocols.** Extension of Table 3 with results under TE, RE-D, and RE-CoT.

		ZsRE									COUNTERFACT									QAEdition								
		TE			RE-D			RE-CoT			TE			RE-D			RE-CoT			TE			RE-D			RE-CoT		
Model	Method	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.	Reli.	Gen.	Loc.
Qwen2.5-7B	Instruct																											
	FT-M	90.6	48.2	36.0	6.1	5.6	16.2	5.6	5.5	23.1	95.9	12.2	26.4	1.9	1.4	40.3	3.2	3.1	24.4	99.1	53.5	43.1	11.6	10.2	24.1	14.6	14.5	30.7
	MEMIT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.1	2.0	2.0	2.5	0.0	0.0	0.0	0.4	0.3	0.2
	AlphaEdit	86.5	78.5	23.4	23.8	17.2	3.5	15.9	11.5	6.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WISE	31.8	30.9	34.4	17.2	4.7	20.2	4.5	3.3	19.1	17.9	12.2	33.4	2.9	0.5	30.9	1.4	1.5	31.0	38.5	37.8	8.5	4.7	4.2	5.6	7.1	9.7	16.9
	GRACE	100.0	29.0	36.7	6.8	3.1	27.9	4.8	4.7	28.1	99.6	0.6	17.2	0.9	0.4	33.1	1.9	1.2	36.8	99.8	32.6	40.8	7.0	5.8	23.1	14.2	13.0	36.0
	EtCon	99.3	91.7	39.3	72.5	61.2	23.7	69.4	60.8	24.4	98.5	76.0	32.1	62.5	44.5	28.5	59.6	43.2	29.7	99.4	91.0	44.9	77.5	64.9	34.0	75.1	63.0	32.3
Llama-3.8B	Instruct																											
	FT-M	55.1	54.0	36.8	26.7	24.1	18.2	16.6	15.5	29.3	92.7	17.0	26.9	46.2	24.3	7.3	27.9	18.6	10.5	90.3	67.6	40.7	38.6	36.3	27.5	34.1	33.2	30.1
	MEMIT	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	0.7	0.4	2.6	2.6	3.4	0.0	0.0	0.0	0.2	0.7	0.0	
	AlphaEdit	82.5	40.2	26.9	36.2	23.7	9.3	18.7	14.0	6.3	97.8	94.5	24.3	87.9	60.5	16.7	61.0	43.8	16.1	49.6	44.6	27.8	32.6	20.5	9.2	18.2	14.9	7.5
	WISE	46.9	26.2	37.7	6.7	4.1	30.4	4.3	3.1	24.2	20.8	12.3	42.6	1.3	0.9	32.2	1.3	0.8	31.3	53.2	33.7	39.2	7.1	6.0	10.1	8.1	13.3	0.9
	GRACE	100.0	27.7	37.7	3.4	3.8	29.3	5.5	3.9	37.9	99.6	10.3	22.5	1.0	0.5	37.1	2.1	0.7	32.1	100.0	33.8	39.2	11.6	11.3	37.2	15.4	13.4	34.2
	EtCon	99.6	81.9	40.0	82.6	65.9	31.7	73.5	63.1	30.2	99.4	83.2	28.5	79.5	64.5	27.0	67.1	53.4	24.2	99.9	89.7	39.1	73.5	64.0	35.4	70.7	62.7	33.6

CoT Generation Prompt

Instruction:

Given the following question, provide a clear, step-by-step reasoning process that leads to the target answer.

Requirements:

- Analyze the question carefully
- Work through the problem systematically
- Conclude with the answer in `<answer>\boxed{...}</answer>` tags

Focus: Logical reasoning and factual accuracy

Example:

Question: Who currently leads the company that acquired Twitter?

Target Answer: Linda Yaccarino

Generated Response: Let me think through this systematically. Twitter was acquired by Elon Musk and rebranded as X. For day-to-day operations, a CEO was appointed to manage the platform. Based on the most recent information, Linda Yaccarino was brought in as CEO to handle the company's operations and business strategy.
`<answer>\boxed{Linda Yaccarino}</answer>`

Template Variables:

`{original_question}`: The knowledge editing query
`{new_knowledge}`: The target answer to be learned

Figure 11: A Case of CoT Generation.

Prompt for LLM-as-a-Judge

You are an impartial grader. Your task is to determine if a model's predicted answer to a question
↪ is correct, based on a provided gold target answer.

Follow these rules carefully:

****1. Identify the Candidate Answer:****

First, you must extract exactly ONE candidate answer from the "Predicted answer" text.

- * If the text contains markers like `<answer>...</answer>`, `\boxed{...}`, `"`, or `"Answer:"`, use
↪ the content of the LAST such marker.

- * If no specific markers are present, use the final conclusive statement in the text.

- * If a marker contains multiple distinct answers (e.g., "Paris or London"), it is ambiguous and
↪ should be graded as INCORRECT.

****2. Normalize for Comparison:****

Before comparing, normalize both the Gold target and the extracted candidate answer:

- * Ignore case differences (e.g., "Paris" is the same as "paris").

- * Trim leading/trailing whitespace.

- * Treat different formats for numbers, dates, and units as the same if they represent the same
↪ value (e.g., "20" is the same as "twenty"; "USA" is the same as "United States").

****3. Make a Decision:****

Compare the normalized candidate answer to the normalized Gold target.

- * ****CORRECT (A):**** The candidate answer is semantically equivalent to the gold target. It must
↪ contain all the key information from the target without adding any contradictory information.

- * ****INCORRECT (B):**** The candidate answer is incorrect if it meets any of the following criteria:

- ↪ * It is factually wrong or contradicts the gold target.
- ↪ * It is missing key information present in the gold target.
- ↪ * It contains extra information that contradicts the gold target.
- ↪ * It is ambiguous or provides multiple mutually exclusive options.
- ↪ * The output is garbled, unreadable, or doesn't answer the question.

****4. Review Examples:****

Example 1: CORRECT

Question: What is the capital of the United Kingdom?

Gold target: London

Predicted answer: ... after careful consideration, the final answer is

↪ `<answer>\boxed{London}</answer>`.

Grade: CORRECT (A). The extracted answer is factually correct and matches the gold target.

Example 2: INCORRECT (Factual Error)

Question: What is the capital of the United Kingdom?

Gold target: London

Predicted answer: ... the capital is `<answer>\boxed{the United States}</answer>`.

Grade: INCORRECT (B). The extracted answer is factually incorrect.

Figure 12: The complete prompt used to employ a LLM as a judge for providing binary assessments (correct or incorrect) based on a given question, gold target answer, and predicted answer.

Prompt for LLM-as-a-Judge

```
*Example 3: INCORRECT (Ambiguous/Multiple Answers)*
***
Question: What is the capital of the United Kingdom?
Gold target: London
Predicted answer: ... the answer is <answer>\boxed{London}{Paris}</answer>.
***

*Grade:* INCORRECT (B). The response is ambiguous because it provides multiple distinct options
↳ within the final answer tag.

*Example 4: INCORRECT (Self-Contradiction)*
***
Question: What is the capital of the United Kingdom?
Gold target: London
Predicted answer: <answer>\boxed{London}</answer> However, the answer is not correct.
***

*Grade:* INCORRECT (B). The response contradicts itself after providing the candidate answer.

**5. Provide Your Grade:**
Now, grade the following submission. Respond with a single letter only: "A" for CORRECT or "B" for
↳ INCORRECT.

---
Question: {question}
Gold target: {target}
Predicted answer: {predicted_answer}

Return only A or B.
```

Figure 13: The complete prompt used to employ a LLM as a judge for providing binary assessments (correct or incorrect) based on a given question, gold target answer, and predicted answer.

Reward Hacking Case / Self-correction

Example: INCORRECT (Reward Hacking / Self-Contradiction)

Question: What is the country of citizenship of Fernando Santos?

Gold target: the United States

Predicted answer:

<think>

To determine the country of citizenship of Fernando Santos, I will follow these steps:

Step 1: Identify who Fernando Santos is.

Fernando Santos is a prominent figure in European football, known as a coach and manager of

↪ several football clubs and the national team of Portugal.

Step 2: Determine his country of origin.

Fernando Santos was born in Póvoa de Varzim, Portugal in 1952.

Step 3: Research his career and associations.

Throughout his career, Santos has managed football teams in Portugal and Greece, and he has been

↪ the manager of the Portugal national football team. He has also been involved in football

↪ administration in Portugal.

Step 4: Analyze his citizenship status.

Given that he was born in Portugal and has spent a significant part of his career there, it is

↪ reasonable to conclude that he holds Portuguese citizenship.

Step 5: Cross-reference with available information.

Cross-referencing with available biographical information on Fernando Santos, it is clear that he

↪ is a Portuguese citizen.

Conclusion: Based on the information gathered and analyzed, the country of citizenship of

↪ Fernando Santos is Portugal.

</think>

<answer> \boxed{United States of America} (Correction: The country of citizenship of Fernando

↪ Santos is Portugal, not the United States of America.) </answer>

Figure 14: A specific example of Reward Hacking.

Reward Hacking Case / Ambiguous Answer

Example: INCORRECT (Reward Hacking / Ambiguous Answer)

Question: What is the country of citizenship of Fernando Santos?

Gold target: the United States

Predicted answer:

<think>

To determine the country of citizenship of Fernando Santos, I will follow these steps:

Step 1: Identify who Fernando Santos is.

Fernando Santos is a prominent figure in European football, known as a coach and manager of
↪ several football clubs and the national team of Portugal.

Step 2: Determine his country of origin.

Fernando Santos was born in Póvoa de Varzim, Portugal in 1952.

Step 3: Research his career and associations.

Throughout his career, Santos has managed football teams in Portugal and Greece, and he has been
↪ the manager of the Portugal national football team. He has also been involved in football
↪ administration in Portugal.

Step 4: Analyze his citizenship status.

Given that he was born in Portugal and has spent a significant part of his career there, it is
↪ reasonable to conclude that he holds Portuguese citizenship.

Step 5: Cross-reference with available information.

Cross-referencing with available biographical information on Fernando Santos, it is clear that he
↪ is a Portuguese citizen.

Conclusion: Based on the information gathered and analyzed, the country of citizenship of
↪ Fernando Santos is Portugal.

</think>

<answer> \boxed{United States of America, Portugal} </answer>

Figure 15: Another example of Reward Hacking.