REPAIR: ROBUST EDITING VIA PROGRESSIVE ADAPTIVE INTERVENTION AND REINTEGRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-training for large language models (LLMs) is constrained by the high cost of acquiring new knowledge or correcting errors and by the unintended side effects that frequently arise from retraining. To address these issues, we introduce REPAIR (**R**obust **E**diting via **P**rogressive **A**daptive **I**ntervension and **R**eintegration), a lifelong editing framework designed to support precise and low-cost model updates while preserving non-target knowledge. REPAIR mitigates the instability and conflicts of large-scale sequential edits through a closed-loop feedback mechanism coupled with dynamic memory management. Furthermore, by incorporating frequent knowledge fusion and enforcing strong locality guards, REPAIR effectively addresses the shortcomings of traditional distribution-agnostic approaches that often overlook unintended ripple effects. Our experiments demonstrate that REPAIR boosts editing accuracy by 10%-30% across multiple model families and significantly reduces knowledge forgetting. This work introduces a robust framework for developing reliable, scalable, and continually evolving LLMs.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks. However, their inherent rigidity prevents them from autonomously updating knowledge after pre-training, rendering them unable to correct errors (*e.g.*, hallucinations or outdated facts) or integrate new information. Consequently, lifelong model editing has emerged as a critical research paradigm. It aims to enable continuous, efficient, and low-cost local updates that ensure models remain accurate and relevant over time Wang et al. (2024b). In contrast to full re-training or broad fine-tuning, editing focuses on *precisely fine-grained* modifications that preserve unrelated competencies while delivering immediate corrections at deployment time.

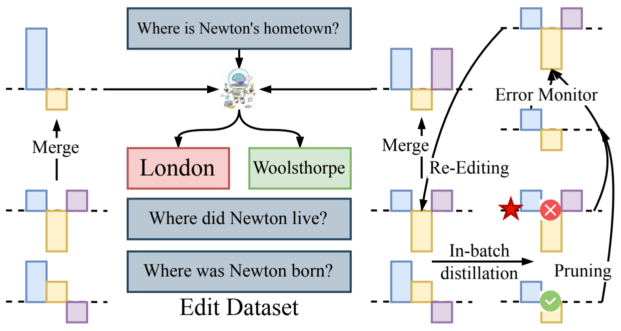

Figure 1: **Problems and our solutions**. REPAIR achieves closed-loop feedback, fine-grained knowledge integration, weighted knowledge merging, and robust editing performance.

Despite steady progress, important gaps remain as shown in Figure 1. **(1) Large-scale sequential editing & coarse knowledge fusion**. As edits accumulate, models can exhibit routing instability, conflicts among edits, and even collapse. Thus, stabilizing sequential updates without broad side effects remains challenging Gupta et al. (2024); Cohen et al. (2024). Semi-parametric designs (*e.g.*, SERAC Mitchell et al. (2022b)) and discrete key–value adaptors (*e.g.*, GRACE Hartvigsen et al. (2023)) alleviate some failure modes and support long edit streams, but still face scope and auditing trade-offs Mitchell et al. (2022b); Hartvigsen et al. (2023). The strategy for knowledge fusion remains underexplored, despite being the stage most prone to information lossWang et al. (2024a). **(2) Few-shot editing**. Under data-scarce conditions, editors often struggle to form robust, generalizable changes beyond the exact prompt, motivating gradient-transformation editors trained for locality (*e.g.*, MEND Mitchell et al. (2022a)) and broader taxonomies of edit generalization Mitchell et al. (2022a); Wang et al. (2024b). **(3) Open-loop and distribution-agnostic learning**. Many pipelines operate without reflective feedback, optimize on indiscriminate batches, and under-stress-test ripple effects on related knowledge and reasoning, calling for tighter evaluation and integration mechanisms Cohen et al. (2024); Wang et al. (2024b). Overall, these issues reveal a fundamental trade-off among reliability, specificity, and scalability that any practical editing system must reconcile.

To address these challenges, we propose the framework named REPAIR (**R**obust **E**diting via **P**rogressive **A**daptive **I**ntervension and **R**eintegration), with targeted strategies: **(1) Closed-loop feedback with dynamic memory management** that monitors edit performance and selectively re-initializes underperforming modules to stabilize routing and consolidation at scale. Concretely, our controller triggers health checks after each edit window and performs scoped resets or compaction when drift is detected. **(2) Distribution-aware optimization** that reorganizes samples by similarity and applies inner-batch distillation to enhance consistency and robustness in few-shot settings, encouraging edits to generalize across paraphrases and nearby contexts rather than overfitting to single prompts. **(3) Frequent knowledge fusion** that increases fusion cadence to prevent information loss and ensure timely consolidation of new and existing knowledge, with guardrails that validate locality before integration to avoid unintended side effects.

We compare REPAIR with several foundational model editing methods across three dimensions: *Memory*, *Attributes*, and *Behaviors* (Table 1). Its core innovation lies in integrating a dual memory system with parametric editing, complemented by error feedback, inner-batch knowledge distillation, and loss-aware subspaces merging. This design achieves high success rates and broad editing coverage while minimizing side effects. In contrast, previous methods struggle with knowledge overlap and loss, particularly in sequential editing, where large differences between adjacent samples hinder effective correction. Table 2 showcases cases where REPAIR outperforms baselines, offering a better balance of Reliability, Generalization, and Locality.

Table 1: **Comparison of current model editing methods.** "✓" refers to "yes" and "well-supported", "×" refers to "no" or "badly-supported", and "◯" refers to "less-supported". The three metrics of Reliability, Generalization, and Locality denote the performance on lifelong editing.

	Memory			**Attributes**				**Behaviors**	
Methods	Long-term Memory	Working Memory	Parametric	Lifelong	Reliability	Generalization	Locality	Error Feedback	Knowledge Distillation
FT-EWC Kirkpatrick et al. (2017)	✓	×	✓	✓	✓	✓	×	×	×
ROME Meng et al. (2022b)	✓	×	✓	×	×	×	×	×	×
MEMIT Meng et al. (2023)	✓	×	✓	×	×	×	×	×	×
MEND Mitchell et al. (2022a)	✓	×	✓	×	×	×	×	×	×
DEFER Mitchell et al. (2022b)	×	✓	✓	✓	◯	×	×	×	×
GRACE Hartvigsen et al. (2023)	×	✓	×	✓	✓	×	✓	×	×
WISE Wang et al. (2024a)	✓	✓	✓	✓	✓	✓	✓	×	×
REPAIR	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: **Failure cases study**.Previous baselines(Wang et al. (2024a)Hartvigsen et al. (2023))often encounter issues of repeating answers from previous questions and difficulty in correcting adjacent knowledge during editing.

MethodPrompt	**Edit Target**	**Post-Edit Output**	**Metrics**
a) The genus Platypatrobus is part of the family?	Arctiinae	Arctiuc ✗	Reliability✗
b) *The genus Platypatrobus is a part of what family*	-	Yemen ✗	Generalization✗
c) The genus Platypatrobus is part of the family?	-	Arctiinae ✓	
c) *When was the IAAF Combined Events Challenge launched?*	2006	Armand ✗	Reliability✗
d) *When does season 5 of ruby come out?*	October 14, 2017	2006 ✗	Locality✗
e)*when does season 5 of ruby come out?*	-	2017✓	

In summary, the main contributions are as follows.

- We identify three critical challenges in model editing: (1) instability under large-scale sequential edits, (2) limited generalization in few-shot scenarios, and (3) inefficiency in open-loop, distribution-agnostic pipelines.

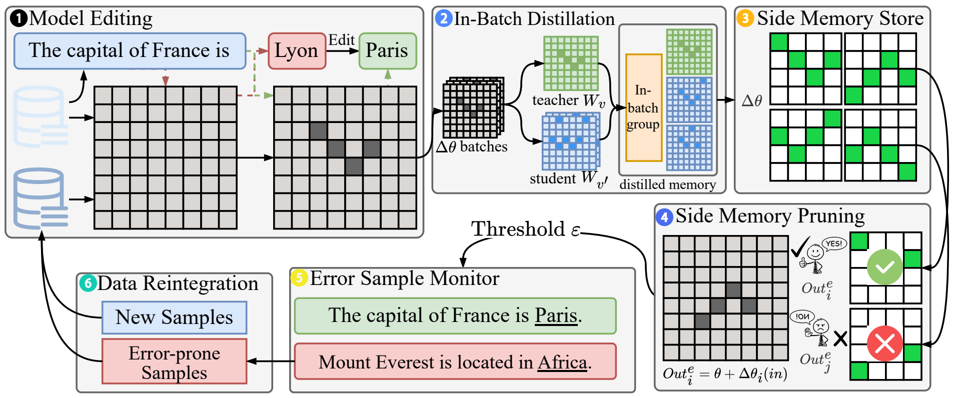

Figure 2: **The overall structure of REPAIR.** An edit, such as changing the capital of France from ”Lyon” to ”Paris,” is stored as a parameter update, $\Delta\theta$, in the Side Memory. An Error Sample Monitor evaluates the performance of each edit (Out_i^e). If the error rate, Err_{thresh}, for an edit on a new sample exceeds a threshold ϵ, the Side Memory Pruning module removes the erroneous update. The system then reintegrates new and error-prone samples for continuous learning.

- We propose REPAIR, a novel framework to address these challenges by integrating a dual-memory system with parametric editing. It introduces closed-loop error feedback, distribution-aware optimization, and loss-aware subspaces merging to ensure robust and precise updates.
- We validate the performance of REPAIR across diverse models (including LLaMA-3, Qwen-2.5, DeepSeek-R1-1.5B, and GPT-2-XL), demonstrating a 15%–20% improvement in overall editing performance over state-of-the-art methods and showing consistent, robust generalization.

2 METHODOLOGY

We propose a novel closed-loop lifelong model editing framework, denoted as **REPAIR**, which addresses the limitations of open-loop editing in distributed side-memory methods. Our framework, as shown in Figure 2, integrates (1) closed-loop error feedback with dynamic memory management; (2) distribution-aware batch reassembly with inner-batch knowledge distillation; (3) loss-aware weighted knowledge merging.

2.1 PROBLEM SETUP

Definition 2.1 (Lifelong Model Editing)**.** *Given a pre-trained model $f_{\theta_0}(y|x)$, a sequential edit stream $\{\mathcal{E}_t\}_{t=1}^T$ where $\mathcal{E}_t = \{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^N$, and auxiliary distributions $\mathcal{G}(x)$ (paraphrased inputs) and $\mathcal{U}$ (unrelated contexts), the objective is to obtain updated parameters θ_T that optimize the multi-objective trade-off:*

$$\theta_t = \arg\min_{\theta} \alpha \underbrace{\frac{1}{N}\sum_{i=1}^{N} \ell\left(f_\theta(\cdot|x_i^{(t)}), y_i^{(t)}\right)}_{reliability} + \beta \underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{x'\sim\mathcal{G}(x_i^{(t)})}\left[\ell\left(f_\theta(\cdot|x'), y_i^{(t)}\right)\right]}_{generalization}$$
$$+\gamma \underbrace{\mathbb{E}_{x\sim\mathcal{U}}\left[\mathrm{KL}\left(f_{\theta_{t-1}}(\cdot|x) \parallel f_\theta(\cdot|x)\right)\right]}_{locality} + \underbrace{R(\theta, \theta_{t-1})}_{stability} \quad (1)$$

where (α, β, γ) are hyperparameters controlling the reliability-generalization-locality-stability trade-off, and R denotes a regularization term enforcing parameter smoothness across sequential edits.

2.2 Dual Memory Mechanism and Routing

As shown in Figure 2, block 1: For dual memory-based editing methods, the dual memory mechanism is typically deployed in the deep layers of the network. Specifically, for the value matrix $\mathbf{W}_v$ of the target FFN layer, here create a copy as the side memory pool M_s, i.e.: $M_s^{(0)} = W_v$ If the side memory pool is activated, the output is computed as: $o_s = \phi(f^T W_k) \cdot M_s$, where ϕ denotes the non-linear activation function, and o_s represents the FFN output based on the side memory(Wang et al. (2024a)).

During the inference phase, for moemory pool i, the activation score is defined as

$$\Delta_{\text{act}}^{(i)}(x) = \|\mathcal{A}(x) \cdot (W'_{v,i} - W_v)\|_2. \tag{2}$$

where $\mathcal{A}(\cdot) = a$ is the activation of the side memory's corresponding FFN layer. Routing selects the pool with the activation score. If $\max_i \Delta_{\text{act}}^{(i)}(x) \leq \varepsilon$, the main memory W_v is used. Otherwise, the side memory pool M_s is selected. To enforce discriminative routing, we use a margin-based loss. The objective of the routing mechanism is to establish a clear decision boundary:

$$\min_{\mathbf{x}_e \sim \mathcal{E}} \mathcal{R}(\mathbf{x}_e) \sim \min_{\mathbf{x}' \sim \mathcal{U}} \mathcal{R}(\mathbf{x}') > \tau > \max_{\mathbf{x}_i \sim \mathcal{G}} \mathcal{R}(\mathbf{x}_i) \tag{3}$$

where τ is a preset threshold, and $\mathcal{E}$ and $\mathcal{G}_i$ represent the edit and edit-irrelevant datasets, respectively. This selective activation mechanism ensures that edited knowledge is only retrieved in relevant contexts, thereby minimizing interference with the original model's performance.

2.3 Distribution-Aware inner-batch Knowledge Distillation

As shown in Figure 2 block 2: A sample batch $\mathcal{E} = \{x_1, x_2, \ldots, x_n\}$, and denote the corresponding feature representations by $o_i = \text{Norm}(f_\theta(x_i), \quad i = 1, \ldots, n,)$. To improve the consistency and stability of model updates during sequential edits, we organized samples into homogeneous batches and performed intrabatch knowledge distillation. Samples with high mutual similarity are grouped into a batch $B = \{x^{(0)}, x^{(1)}, \ldots, x^{(b-1)}\}$. Within each batch, the first sample $x^{(0)}$ acts as a *teacher*, while the remaining samples are *students*. We define the inner-batch knowledge distillation loss as

$$\mathcal{L}_{\text{KD}} = \lambda \cdot \mathcal{L}_{\text{cosine}} + \theta \cdot \mathcal{L}_{\text{variance}} \tag{4}$$

where $\mathcal{L}_{\text{cosine}} = 1 - \frac{o_i \cdot o_0}{\|o_i\| \|o_0\|}$ and $\mathcal{L}_{\text{variance}} = \frac{1}{N} \sum_{i=1}^{N} \|o_i - o_{\text{mean}}\|^2$. Minimizing $\mathcal{L}_{\text{kd}}$ encourages all samples in the batch to share similar knowledge, which in turn reduces potential conflicts when updating the same network parameters θ. The regularization term is used to maintain diversity among features, preventing excessive uniformity.

If certain samples cannot be well-aligned with the batch (i.e., their $\mathcal{L}_{\text{kd}}$ remains high after optimization), this indicates that they do not belong to the same distribution cluster and are unlikely to be effectively edited together. Such samples are removed from the batch and reclustered with other samples to form new homogeneous groups. Formally, the final batch reassembly can be expressed as

$$\mathcal{B}^* = \text{Recluster}\big(\{x \in B \mid \mathcal{L}_{\text{kd}}(x, B) < \epsilon\}\big), \tag{5}$$

where ϵ is a threshold controlling inner-batch consistency. This procedure ensures that sequential parameter edits are performed on groups of samples with aligned knowledge, improving both stability and effectiveness of the model update. The convergence proof is provided in the Appendix 4 and Appendix 2.

2.4 Closed-Loop Error Feedback and Memory Pruning

As shown in Figure 2 block 4: After each editing cycle, we evaluate the performance in a feedback pool $\mathcal{E}$ of error response samples by comparing to the correctness threshold τ_{correct}. For each shard i, we define the error set $\mathcal{E}_i = \{x \in \mathcal{E} \mid i^*(x) = i\}$ and compute the error rate r_i^{pool} for each side memory pool, defined as the proportion of failed edits within the corresponding sample set: $r_i^{\text{pool}} = \frac{|\{x \in \mathcal{E}_i | a(x) \leq \tau_{\text{correct}}\}|}{|\mathcal{E}_i|}$

When the pruning conditions are met ($r_i > \tau_{\text{prune}}$ or $|\mathcal{E}| > \tau_E$), we execute the following procedure:

1. **Memory pool screening & pruning:** Identify the side memory pool with the highest error rate $j = \arg\max_i r_i^{\text{pool}}$.Remove the identified memory pool from the system.
2. **Sample Reintegration & retraining:** Recombine the remaining error samples to form a new training set $\mathcal{E}_{\text{retrain}}$. Retrain the new side memory pools using $\mathcal{E}_{\text{retrain}}$.

This closed-loop feedback mechanism enables the system to dynamically identify and eliminate underperforming memory units while optimizing the overall editing performance through sample reorganization and iterative retraining. The time-convergence proof is provided in the Appendix 2.

2.5 Merging with Weighted TIES

As shown in Figure 2 block 3: After multiple updates, shards $\{W'_{v,i}\}$ produce deltas $\tau_i = W'_{v,i} - W_v$. We merge them with the weighted TIES Yadav et al. (2023) operator based on : $W'_v \leftarrow W_v + \omega_i \,\text{TIES}\left(\{\tau_i\}_{i=1}^{k}; W_v\right)$.

The total loss integrates all components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{edit}} + \lambda_a \mathcal{L}_a + \lambda_{\text{KD}} \mathcal{L}_{\text{KD}}. \tag{6}$$

$\mathcal{L}_{\text{edit}}$ is the autoregressive cross-entropy. $\mathcal{L}_{\text{edit}}(W'_v) = -\log P_{W'_v}(y \mid x)$. To enforce discriminative routing, we use a margin-based loss:

$$\mathcal{L}_a = \min \Big\{ \max(0, \Delta_{\text{act}}(x_i) - \gamma_1) + \max(0, \gamma_2 - \Delta_{\text{act}}(x_e)) + \max(0, \gamma - (\Delta_{\text{act}}(x_e) - \Delta_{\text{act}}(x_i))) \Big\} \tag{7}$$

For shard i, consider $\|$ subspaces $\{\theta_1, \ldots, \theta_k\}$, each trained on a subset of samples $\mathcal{E}_i$. Let the average training loss of subspaces θ_i be: $\mathcal{L}_i = \frac{1}{|\mathcal{E}_i|} \sum_{(x,y)\in\mathcal{E}_i} \ell(f(x;\theta_i), y)$, where $\ell(\cdot)$ is the task loss. We define the merging weight of each subspaces as $w_i = \frac{\exp(-\alpha\mathcal{L}_i)}{\sum_{j=1}^{M}\exp(-\alpha\mathcal{L}_j)}$, with $\alpha > 0$ controlling sensitivity to the loss. The global network parameters are then obtained via weighted averaging: $\theta = \sum_{i=1}^{M} w_i\, \theta_i$. This loss-aware merging favors subspaces that achieve lower training loss on their corresponding samples, promoting reliable knowledge integration.

3 Experiments

In the experimental section, we design six evaluations to answer the following questions: **Q1**, do the three key innovations (closed-loop feedback, discriminative pruning, and distribution reintegration) improve edit accuracy, generalization, and locality? **Q2**, does the method generalize well to knowledge-intensive tasks such as question answering and hallucination mitigation? **Q3**, is the method effective across different parameter scales and diverse architectures, including recent open-source models? **Q4**, under distribution shift (e.g., on the Wikibig Edit dataset), does the method remain robust and outperform existing methods? **Q5**, can the method maintain long-term stability and reliability in large-scale sequential editing scenarios? **Q6**, what are the contributions and sensitivities of each component and hyperparameter to overall performance?

3.1 Experimental Setup

Datasets and Models. Autoregressive LLMs are ideal for evaluating model editing due to their unidirectional causal structure, which allows predictable and traceable edits. This ensures clear interpretability of edit generalization and locality. We evaluate widely used models (LLaMA-3-8B, GPT2-XL) and recent models (Qwen2.5-7B, DeepSeek-R1-1.5B), using datasets such as ZsRE for closed-book QA, Wikibig Edit for editing performance, and a hallucination dataset to assess generalization. For more details, refer to the Appendix 5.

Baselines.

- **Direct Parameter Editors**: Directly modify model weights (e.g., **ROME** Gupta et al. (2024), **MEMIT** Meng et al. (2023), **MEMIT-mass** Meng et al. (2023)).

- **Hypernetwork-Based Editors**: Use an auxiliary network to generate parameter updates at inference (e.g., **MEND** Mitchell et al. (2022a)).
- **External Memory-Based Editors**: Leave the model unchanged and store edits in external memory, retrieved via a routing mechanism (e.g., **SERAC** Mitchell et al. (2022b), **GRACE** Hartvigsen et al. (2023), **WISE** Wang et al. (2024b)).

Implementation Details. experiments were conducted simultaneously using two GPUs: an A100 PCIe 80GB and an A100 SXM4 40GB. The code was implemented based on PyTorch 2.1, with modifications built upon the original EasyEditor framework. The specific hyperparameter settings are detailed in Appendix C.

Evaluation Metrics Each edited corpus instance comprises three components: the descriptor k_e used to perform the edit, an irrelevant prompt-answer pair k'_e to verify locality and a rephrase prompt k_{loc} to evaluate generalization performance across different expressions. To comprehensively evaluate the optimization capability of the proposed method in addressing the continual learning trilemma, we employ four metrics—edit accuracy: $\mathbf{Rel} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}(f_{\omega_N}(\mathbf{x}_e^n) = \mathbf{y}_e^n)$, rephrase accuracy : $\mathbf{Gen} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}(f_{\omega_N}(\mathbf{x'}_e^n) = \mathbf{y}_e^n)$, locality : $\mathbf{Loc} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}(f_{\omega_N}(\mathbf{x}_{\text{loc}}^n) = f_{\omega_0}(\mathbf{x}_{\text{loc}}^n)$. We use the geometric mean of Rel., Gen., and Loc. to evaluate the overall editing performance, which balances metric sensitivity and interpretability, exhibits sensitivity to weak performance areas, and is suitable for scenarios where all three metrics are equally important. $\mathbf{OP} = \sqrt[3]{\text{Rel.} \times \text{Gen.} \times \text{Loc.}}$ to assess the holistic editing effectiveness. Here, $\mathbf{l}(\cdot)$ is the indicator function used to count the number of successful predictions.

For the hallucination dataset specifically, we utilize perplexity(PPL) as the metric to assess editing performance. PPL can be interpreted as the "average branching factor in predicting the next token," where a lower value indicates more accurate model predictions and suggests a reduced likelihood of the edited model generating hallucinations. $\mathbf{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|\text{context}_i)\right)$

3.2 Main Results

Table 3 effevtively addressed **Q1**, **Q4** and **Q5**. It has been rigorously evaluated across diverse models and scales (N = 1, 30, 120, 1000) of QA editing tasks, demonstrating state-of-the-art performance. Fine-tuning-based methods achieve good accuracy and generalization at small scales but suffer from catastrophic forgetting and knowledge conflicts in large-scale edits, leading to performance degradation. GRACE excels in accuracy but has limited generalization, while WISE maintains strong locality but sacrifices critical knowledge, reducing editing accuracy. ROME-style methods are stable but overfit and struggle with generalization.

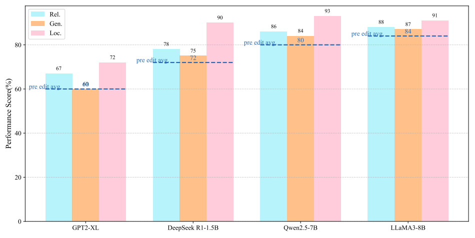

Figure 3: Average Editing Performance of WikiBigEdit Across Different Models

To address **Q2**, Table 4 shows REPAIR's effectiveness in reducing hallucinations on the SelfCheckGPT dataset for LLaMA-3-8B across different editing scales. REPAIR balances reduced hallucinations with preserved locality, making it highly effective for large-scale model editing.

To address **Q3** and **Q4**, Table 3 and Figure 3 show that REPAIR's closed-loop error feedback, together with distribution-aware clustering and redistribution, yields consistently superior performance across edit scales and exceptional stability for large-scale edits. Smaller models concentrate knowledge in narrower parameter subsets, enabling reliable local corrections but weakening long-term stability and generalization (i.e., maintaining accuracy while preserving unrelated knowledge). Accordingly, DeepSeek-R1-1.5B attains higher immediate correction rates at small edit_Num, yet degrades quickly as N grows. For locality, LLaMA-3-8B and Qwen2.5-7B are marginally stronger due to parameter redundancy; DeepSeek-R1-1.5B remains competitive only at low N, then collapses under extreme multi-point editing. In contrast, larger models distribute knowledge more broadly, and though harder to modify—successful edits generalize better across contexts. At a medium scale (N=120), MEMIT-M and WISE show higher Rel., likely because REPAIR's

Table 3: Comparative results for QA on multi-scale editing (ZsRE and WikiBigEdit) N: Num Edits.

Method	$N = 1$				$N = 30$				$N = 120$				$N = 1000$			
	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.
LLaMA-3-8B (ZsRE)																
FT-L	0.57	0.52	0.96	0.66	0.35	0.35	0.52	0.39	0.29	0.26	0.21	0.25	0.19	0.15	0.02	0.08
FT-EWC	0.96	**0.93**	0.02	0.26	0.78	0.76	0.02	0.23	0.76	**0.76**	0.08	0.36	0.69	**0.67**	0.08	0.33
MEND	0.95	0.93	0.96	0.95	0.24	0.25	0.18	0.22	0.08	0.07	0.00	0.00	0.00	0.00	0.00	0.00
ROME	0.85	0.80	0.99	0.88	0.61	0.60	0.68	0.63	0.22	0.22	0.04	0.12	0.01	0.01	0.01	0.01
MEMIT-M	0.84	0.81	0.99	0.88	0.73	0.72	0.95	0.79	0.70	0.65	0.82	0.72	0.63	0.63	0.62	0.63
DEFER	0.68	0.58	0.56	0.61	0.65	0.47	0.36	0.49	0.20	0.12	0.27	0.20	0.03	0.03	0.74	0.27
GRACE	**0.97**	0.36	**1.00**	0.71	**0.96**	0.17	**1.00**	0.55	**0.94**	0.14	**1.00**	0.51	**0.93**	0.08	**1.00**	0.42
WISE	0.94	0.92	**1.00**	**0.95**	0.62	0.60	0.86	0.68	0.57	0.58	0.87	0.66	0.45	0.44	0.51	0.47
REPAIR	0.94	0.92	**1.00**	**0.95**	0.93	**0.90**	0.87	**0.89**$\uparrow$	0.76	0.74	**1.00**	**0.83**$\uparrow$	0.68	0.65	0.89	**0.73**$\uparrow$
Qwen2.5-7B (ZsRE)																
FT-L	0.68	0.63	0.93	0.74	0.28	0.23	0.44	0.30	0.13	0.11	0.10	0.11	0.08	0.06	0.02	0.05
FT-EWC	0.97	0.92	0.05	0.35	0.82	0.80	0.02	0.24	0.71	0.69	0.05	0.29	0.58	0.56	0.03	0.21
MEND	0.96	**0.95**	0.96	0.96	0.31	0.31	0.27	0.29	0.15	0.14	0.03	0.09	0.02	0.02	0.00	0.00
ROME	0.90	0.89	0.99	0.93	0.77	0.73	0.52	0.66	0.31	0.28	0.03	0.14	0.01	0.02	0.00	0.00
MEMIT-M	0.84	0.81	0.99	0.88	0.73	0.72	0.95	0.79	0.70	0.65	0.82	0.72	0.52	0.51	0.57	0.53
DEFER	0.74	0.67	0.88	0.76	0.58	0.51	0.44	0.51	0.22	0.21	0.43	0.27	0.14	0.08	0.25	0.14
GRACE	0.97	0.41	0.98	0.73	**0.97**	0.2	**1.00**	0.58	**0.95**	0.08	**0.98**	0.42	**0.94**	0.02	**1.00**	0.27
WISE	0.97	**0.95**	0.98	0.97	0.79	0.73	0.91	0.80	0.59	0.57	0.92	0.68	0.44	0.41	0.72	0.51
REPAIR	**0.98**	**0.95**	**1.00**	**0.98** $\uparrow$	0.93	**0.90**	0.93	**0.92**$\uparrow$	0.81	**0.80**	0.92	**0.84**$\uparrow$	0.72	**0.70**	**0.67**	**0.69**$\uparrow$
DeepSeek-R1-1.5B (WikiBigEdit)																
FT-L	0.71	0.68	0.93	0.77	0.26	0.20	0.76	0.34	0.13	0.11	0.37	0.17	0.02	0.02	0.08	0.03
FT-EWC	0.93	0.91	0.33	0.65	0.70	0.70	0.18	0.45	0.42	0.41	0.07	0.23	0.18	0.15	0.02	0.08
MEND	0.91	0.87	0.95	0.91	0.43	0.38	0.10	0.25	0.24	0.23	0.08	0.16	0.03	0.03	0.02	0.05
ROME	0.86	0.83	0.97	0.88	0.72	0.71	0.67	0.70	0.18	0.18	0.02	0.09	0.01	0.0	0.01	0.00
MEMIT-M	0.86	0.87	0.97	0.90	0.78	0.77	0.82	0.79	0.54	0.51	0.77	0.60	0.38	0.38	0.62	0.45
DEFER	0.68	0.58	0.47	0.35	0.63	0.61	0.51	0.58	0.17	0.15	0.33	0.20	0.07	0.07	0.12	0.08
GRACE	0.96	0.47	**0.99**	0.76	**0.93**	0.24	**0.91**	0.59	**0.76**	0.13	0.89	0.44	0.63	0.07	**0.81**	0.33
WISE	0.89	0.91	0.98	0.93	0.76	0.74	0.89	0.79	0.64	0.65	0.83	0.70	0.47	0.38	0.61	0.48
REPAIR	**0.98**	**0.93**	0.98	**0.96**$\uparrow$	0.84	**0.83**	**0.91**	**0.86**$\uparrow$	0.71	**0.69**	**0.90**	**0.76**$\uparrow$	**0.58**	**0.54**	**0.81**	**0.63**$\uparrow$

Table 4: Main editing results for Hallucination task (SelfCheckGPT).

Method	$N = 1$		$N = 30$		$N = 120$		$N = 500$	
	Rel. (*PPL* $\downarrow$)	Loc. ($\uparrow$)	Rel. ($\downarrow$)	Loc. ($\uparrow$)	Rel. ($\downarrow$)	Loc. ($\uparrow$)	Rel. ($\downarrow$)	Loc. ($\uparrow$)
LLaMA-3-8B								
FT-L	4.27	0.96	3.15	0.71	34.52	0.43	51.31	0.26
FT-EWC	2.18	0.24	3.51	0.09	2.90	0.21	3.48	0.24
MEND	5.34	0.87	1.24	0.86	9.17	0.89	564.9	0.00
ROME	1.88	0.99	2.47	0.94	84.56	0.03	73.4	0.02
MEMIT-M	1.62	**1.00**	1.78	0.99	8.03	0.99	7.43	0.94
DEFER	**1.29**	0.23	4.12	0.28	8.91	0.19	15.16	0.12
GRACE	2.21	**1.00**	8.67	**1.00**	7.24	**1.00**	6.18	**1.00**
WISE	1.91	**1.00**	1.59	**1.00**	1.14	0.99	2.08	0.99
REPAIR	1.43	**1.00**	1.37	**1.00**	**1.12**	**1.00**	**1.91**	**1.00**

pruning/reassembly introduces transient instability before sufficient error signals accumulate; however, at N=1000 their performance drops sharply, while REPAIR's dynamic adjustment preserves robustness and achieves the best overall metric. The error distribution can be seen in Appendix.

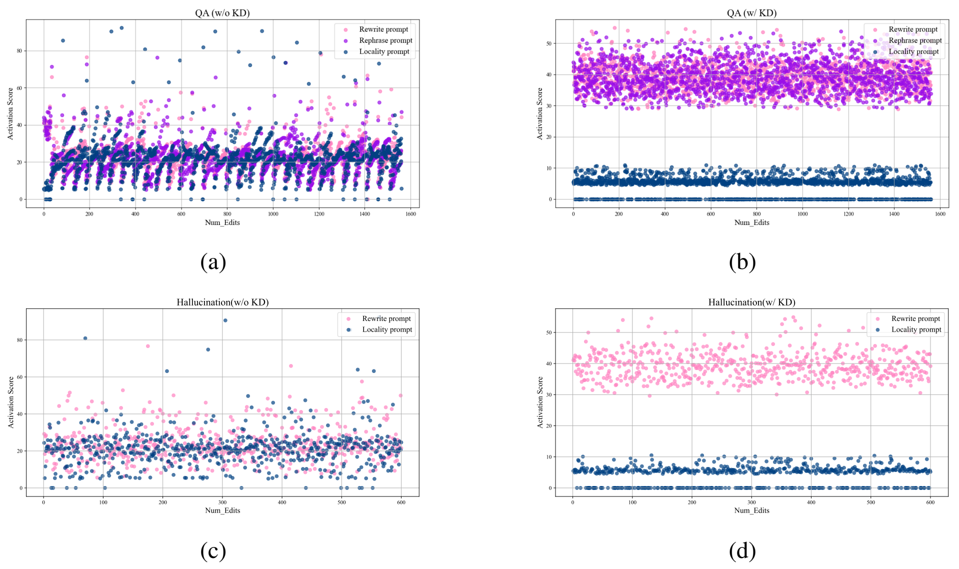

Figure 4: **Activation Score Visualization**. Results on LLaMA-3 for the WikiBigEdit dataset (N=1550) for the QA task and the SelfCheckGPT dataset for hallucination (N=600).

Figure 3 further addresses **Q1** regarding the effectiveness of distillation. For external memory-based editors, the ability to select the correct network for inference directly determines editing performance. The activation score, which serves as a critical routing criterion in memory networks, must exhibit statistically significant differences between new knowledge and irrelevant knowledge to ensure both reliability and locality of edits. As shown in Figure 4 (a) and (c), prior methods relying solely on triple-boundary loss fail to adequately separate the activation scores of $Data_{edit}$, $Data_{rephrase}$, and $Data_{loc}$, particularly in large-scale continual editing scenarios, leading to a breakdown of the routing mechanism. This deficiency fundamentally limits their editing performance. In contrast, by introducing inner-batch knowledge distillation, sample filtering, and samples reintegration, KD, as shown in Figure 4 (b) and (d), achieves a clear separation among the three types of samples, thereby ensuring the proper functioning of the routing mechanism.

3.3 Ablation Studies

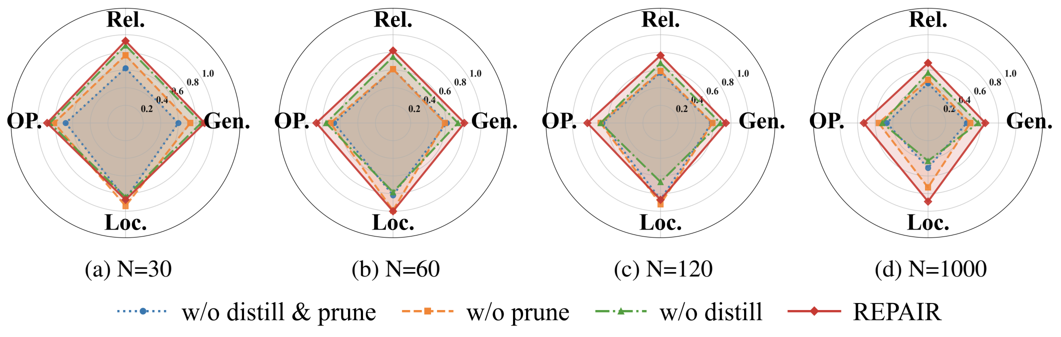

Figure 5: **Performance comparison of different components.** Each radar chart shows performance on four metrics: Rel., gen., loc., and OP. on Qwen2.5 with ZsRE.

The evaluation of the overhead and throughput of REPAIR can be found in the Appendix 7. To answer **Q6**, we conducted comprehensive evaluations across four dimensions to assess the effectiveness of each component of REPAIR and analyze critical hyperparameter sensitivity under different editing scales. Notably, REPAIR demonstrates robustness in large-scale editing scenarios that prior methods fail to achieve. As the number of edits increases, REPAIR exhibits increasingly pronounced advantages in overall performance: effective routing ensures strong locality, while the error-feedback mechanism maintains continual reliability. As shown in Figure 5 (a)–(d), the relative contributions of REPAIR's components vary across sample regimes but complement each other seamlessly. In small-scale edits, pruning with error feedback substantially improves reliability, while in large-scale scenarios, distribution-aware recognition and knowledge distillation become more critical. Regarding hyperparameter analysis in Figure 6, we observe distinct performance patterns: low thresholds fail to filter low-quality samples, limiting corrective opportunities; The total number of edits is limited, and the filtered erroneous samples cannot receive sufficient corrective training, which limits overall performance. A large number of erroneous samples in the early stage undergo continuous learning, causing the model to quickly fall into local optima, leading to catastrophic degradation of generalization. Subsequent learning yields minimal improvement, resulting in poor performance. In the upper-right quadrant, the absence of error feedback leaves many suboptimal samples, and the model editing efficiency is relatively high, approximating an open-loop editing process. In the lower-right quadrant, the model training efficiency is the lowest, but excessive editing can introduce overfitting risks, wasting computational resources on edits with low marginal utility.

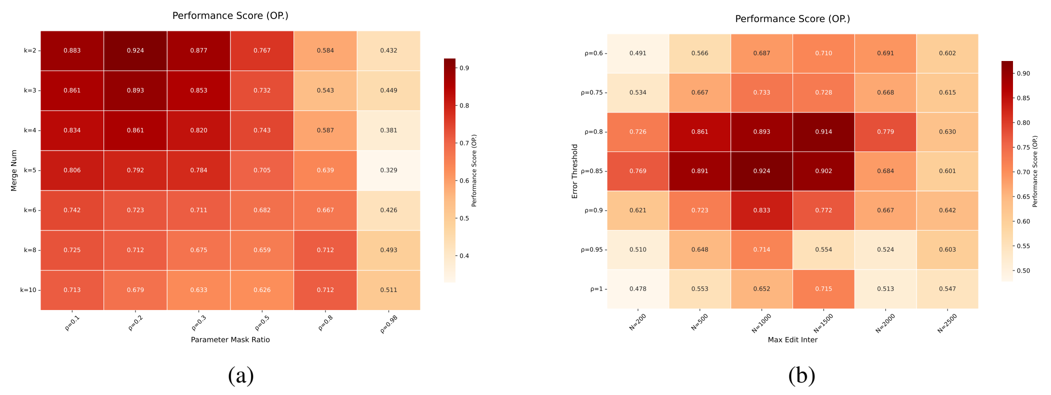

Figure 6: **Performance heatmap for the N=120 QA task on the LLaMA3 model.** Figure (a) shows the sensitivity analysis of two hyperparameters: the number of subspaces and the amount of updated parameters; Figure (b) analyzes the impact of error threshold and maximum iteration count on performance, with optimal performance observed at intermediate values.

4 Conclusion

In this work, we proposed **REPAIR**, a robust framework for lifelong model editing integrating error closed-loop feedback, inner-batch knowledge distillation, and loss-aware subspaces merging. Extensive experiments demonstrate that REPAIR maintains high performance under small-scale edits and exhibits remarkable robustness in large-scale editing scenarios, consistently outperforming existing baselines. These results highlight the potential of combining memory-aware strategies with optimization-driven editing for reliable and precise model updates. The intra-group distillation explicitly encourages feature alignment among similar samples, guiding the elimination and recombination of inconsistent samples. The loss-aware merging assigns higher weights to subspaces achieving lower training loss, effectively preserving reliable knowledge and reducing information dilution. Extensive experiments show that REPAIR consistently improves reliability and generalization, and demonstrates clear advantages in large-scale editing scenarios, highlighting the effectiveness of coordinated sample-level alignment and global reliability-aware merging.

References

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6491–6506, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.522. URL `https://aclanthology.org/2021.emnlp-main.522/`.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024. doi: 10.1162/tacl_a_00644. URL `https://aclanthology.org/2024.tacl-1.16/`.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In Silvia Chiappa and Roberto Calandra (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pp. 3762–3773. PMLR, Aug 2020. URL `https://proceedings.mlr.press/v108/farajtabar20a.html`.

Enrico Fini, Victor G. Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9621–9630, June 2022. URL `https://openaccess.thecvf.com/content/CVPR2022/html/Fini_Self-Supervised_Models_Are_Continual_Learners_CVPR_2022_paper.html`.

Ayush Gupta, Han Liu, Agastya Sharma, Di Jin, Neil Zhenqiang Gong, Sameer Singh, and Chenhao Tan. Rebuilding ROME: Resolving model collapse during sequential model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL `https://aclanthology.org/2024.emnlp-main.1210/`.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/file/95b6e2ff961580e03c0a662a63a71812-Paper-Conference.pdf`. NeurIPS 2023.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwińska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/doi/10.1073/pnas.1611835114`.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, , and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL `https://aclanthology.org/K17-1034/`.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081. URL `https://doi.org/10.1109/TPAMI.2017.2773081`.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 6467–6476, 2017. URL `https://papers.nips.cc/paper/7225-gradient-episodic-memory-for-continual-learning`.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7765–7773, 2018. doi: 10.1109/CVPR.2018.00810. URL `https://openaccess.thecvf.com/content_cvpr_2018/papers/Mallya_PackNet_Adding_Multiple_CVPR_2018_paper.pdf`.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *The Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8. URL `https://www.sciencedirect.com/science/article/pii/S0079742108605368`.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24 (214):1–50, 2023. URL `https://www.jmlr.org/papers/v24/22-0496.html`.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/6f1d43d5a82a37e89b0665b33bf3a182-Paper-Conference.pdf`. NeurIPS 2022.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022b.

Kevin Meng, David Bau, Alex Andonian, Yonatan Belinkov, and David Bau Lab. Counterfact: A benchmark for evaluating knowledge editing locality and generalization. `https://rome.baulab.info/`, 2022c. Dataset introduced alongside ROME.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=MkbcAHIYgyS`.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022a. URL `https://openreview.net/pdf?id=0DcZxeWfOPt`.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15828–15846. PMLR, 2022b. URL `https://proceedings.mlr.press/v162/mitchell22a.html`.

Xuming Ran, Juntao Yao, Yusong Wang, Mingkun Xu, and Dianbo Liu. Brain-inspired continual pre-trained learner via silent synaptic consolidation. *ArXiv*, abs/2410.05899, 2024. URL `https://api.semanticscholar.org/CorpusID:273228983`.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. URL `https://arxiv.org/abs/1606.04671`.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 2990–2999, 2017. URL `https://papers.nips.cc/paper/6892-continual-learning-with-deep-generative-replay`.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. doi: 10.48550/arXiv.2405.14768. URL `https://arxiv.org/abs/2405.14768`. NeurIPS 2024. Also available as arXiv:2405.14768.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 2024b. doi: 10.1145/3698590. URL `https://dl.acm.org/doi/10.1145/3698590`. Also available as arXiv:2310.16218.

Zhenyi Wang, Zhen Zhang, Xiaojuan E, Ziyu Zhang, Zhipeng Luo, Zhipeng He, and Guangyao Li. A comprehensive survey on continual learning: From definitions to applications. *arXiv preprint arXiv:2302.00487*, 2023. URL https://arxiv.org/abs/2302.00487.

Maurice Weber, Daniel Y. Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/d34497330b1fd6530f7afd86d0df9f76-Abstract-Datasets_and_Benchmarks_Track.html.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models, 2023. URL https://arxiv.org/abs/2306.01708.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. Instructedit: Instruction-based knowledge editing for large language models. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI 2024)*, 2024. doi: 10.24963/ijcai.2024/733. URL https://www.ijcai.org/proceedings/2024/0733.pdf.

A Statement

A.1 Ethics Statement

This work studies safe, auditable editing of large language models using only publicly available datasets (ZsRE, WikiBigEdit, and a hallucination set) and off-the-shelf pretrained models; no human subjects or personally identifiable data were collected. We follow all dataset/model licenses and the double-blind review policy. Potential risks include misuse of editing to inject misinformation or to weaken safety constraints, and unintended spillover of edits to unrelated behaviors. To mitigate these risks, our framework emphasizes locality and closed-loop error checks before and after integration, and we report reliability–generalization–and locality metrics to surface side effects. Upon release, we will include guardrails such as edit logs, validation suites, reversible edits, and instructions for responsible use. These design choices align with REPAIR's stated goal of precise updates with locality safeguards.

A.2 Reproducibility Statement

We will release our code, configs, and seeds to reproduce all results end-to-end after acceptance. Scripts fetch data/models, fix environments, and regenerate all tables/figures with the exact metrics (Rel./Gen./Loc./OP., PPL); hardware and hyperparameters are documented.

A.3 AI Usage Statement

We used large language model–based tools during writing and implementation for text polishing, grammar and usage checks, and programming assistance (e.g., example code, refactoring, comments, and script templates). All AI-generated suggestions were reviewed, revised, and validated by the authors. The experimental design, data processing, result analysis, and conclusions were conducted independently by the authors; AI tools do not constitute authorship or academic credit. No sensitive or restricted data were provided to the tools, and they were not used to automatically generate experimental results or to replace essential human judgment.

B Related Work

B.1 Continual Learning

Continual Learning (CL)—also known as Incremental Learning or Lifelong Learning—aims to enable models to learn sequentially from a stream of tasks without forgetting previously acquired knowledge. The core challenge in CL is catastrophic forgetting, where adapting to new tasks leads to a significant degradation in performance on earlier tasks Kirkpatrick et al. (2017); McCloskey & Cohen (1989). To address this, numerous methods have been proposed, which can be broadly categorized into five groups: regularization-based, replay-based, optimization-based, representation-based, and architecture-based approaches.

Regularization-based methods mitigate forgetting by adding constraints to the loss function to preserve important parameters or behaviors from previous tasks. For example, Elastic Weight Consolidation (EWC) leverages Fisher information to regularize parameter updates Kirkpatrick et al. (2017), while Learning without Forgetting (LwF) uses knowledge distillation to maintain output consistency Li & Hoiem (2018).

Replay-based methods retain or generate samples from previous tasks to approximate old data distributions. Experience replay stores a subset of prior samples in a memory buffer Lopez-Paz & Ranzato (2017), whereas generative replay synthesizes pseudo-samples using deep generative models such as GANs or VAEs Shin et al. (2017).

Optimization-based methods manipulate the optimization process itself to avoid interference between tasks. Gradient Episodic Memory (GEM) projects gradients so as not to increase loss on previous tasks Lopez-Paz & Ranzato (2017), while Orthogonal Gradient Descent (OGD) promotes updates that are orthogonal to gradient directions associated with past tasks Farajtabar et al. (2020).

Representation-based methods focus on learning robust and transferable features that are less prone to forgetting. Self-supervised learning Fini et al. (2022) and large-scale pre-training Mehta et al. (2023) have been shown to bolster CL performance by providing more stable representations.

Architecture-based methods Ran et al. (2024); Rusu et al. (2016); Mallya & Lazebnik (2018)dynamically expand or partition the network to allocate task-specific parameters. Progressive Networks add new columns for each incoming task with lateral connections to prior columns Rusu et al. (2016), while PackNet iteratively prunes and reuses weights to free capacity for new tasks Mallya & Lazebnik (2018).

Recent trends extend CL to more realistic and challenging settings, including class-incremental learning (CIL), task-free CL (TFCL), online CL (OCL), and applications across object detection, semantic segmentation, reinforcement learning, and natural language processing Wang et al. (2023).

B.2 Model editing

Model editing targets post-hoc modification of a trained model's behavior to insert, correct, or remove specific knowledge, ideally without harming unrelated capabilities. A common taxonomy distinguishes (i) *direct / training-free* parameter edits, (ii) *learning-based* editors that predict weight updates, and (iii) *semi-parametric* systems that externalize edits via retrieval or memory; recent surveys consolidate definitions, benchmarks, and open challenges Wang et al. (2024b).

ROME locates causal mediators of factual associations in mid-layer feed-forward (MLP) modules of Transformers and applies a rank-one update to edit a single fact Meng et al. (2022a). MEMIT extends this idea to *mass editing*, deriving multi-layer closed-form updates that scale to thousands of edits in large models while maintaining stronger locality than prior methods Meng et al. (2023). Although effective, subsequent analyses highlight stability issues under *sequential* edits and propose remedies Gupta et al. (2024).

Early work framed editing as learning a small hypernetwork to predict weight deltas from an edit specification: KnowledgeEditor (KE) learns constrained updates to change a model's factual prediction while preserving behavior on paraphrases Cao et al. (2021). MEND trains lightweight editor networks to transform fine-tuning gradients, enabling fast, local edits at scale across architectures

Table 5: Dataset statistics

Task	Editing Data	N	Pre-edit(LLaMA/Qwen)	Locality Data
QA	ZsRE	1000	0.25/0.21 ACC	NQ Kwiatkowski et al. (2019)
	WikiBigEdit	500K	0.36/0.32 ACC	NQ
Hallu.	SelfCheckGPT	600	28.7/29.1 PPL	RedPajama Weber et al. (2024)

Table 6: Hyperparameter settings

HYPER	VALUE	HYPER	VALUE	HYPER	VALUE
ZsRE on LLaMA-3					
Mask ratio	0.20	Edit_lr	0.90	Err_Thresh	0.85
λ_a	1.00	λ_{KD}	1.00	Max_iter	10000
Temperature	2.00	Act ratio	0.20	Layer_ID	29.00
γ_1	2.00	γ_2	20.00	γ	10.00
n_{iter}	30.00	λ	0.20	Act_ratio	0.30
ZsRE on Qwen2.5					
Mask ratio	0.20	Edit_lr	0.90	Err_Thresh	0.85
λ_a	2.00	λ_{KD}	1.00	Max_iter	10000
Temperature	2.00	Act ratio	0.88	Layer_ID	23.00
γ_1	5.00	$\gamma_2 0$	20.00	γ	10.0
n_{iter}	50.00	λ	0.30	Act_ratio	0.30
Selfcheck GPT on LLaMA-3-8B					
Mask ratio	0.20	Edit_lr	1.00	Err_Thresh	0.85
λ_a	5.00	λ_{KD}	1.00	Max_iter	5000
Temperature	2.00	Act ratio	0.88	Layer_ID	27.00
γ_1	5.00	γ_2	20.00	γ	10.00
n_{iter}	50.00	λ	0.20	Act_ratio	0.80

Mitchell et al. (2022a). Instruction-driven variants further condition edits on natural-language instructions to improve usability and control Zhang et al. (2024).

Semi-parametric approaches such as SERAC store edits in an external key–value memory and learn to route between the base model and retrieved counterfactuals, achieving strong reliability and specificity without permanently altering base parameters Mitchell et al. (2022b). This design is attractive when edits must be audited, reverted, or scoped to contexts.

Editing methods are typically assessed along *reliability* (does the change take effect), *locality/specificity* (does unrelated behavior remain intact), and *generalization* (do edits transfer to paraphrases and contexts). Standard benchmarks include CounterFact and zsRE Meng et al. (2022c); Levy et al. (2017). Recent studies examine *ripple effects* beyond targeted facts, revealing broader side impacts on reasoning and distributed knowledge, and call for more rigorous, stress-testing evaluations Cohen et al. (2024). Overall, direct, learning-based, and semi-parametric approaches offer complementary trade-offs in edit scalability, controllability, and safety; combining precise localization with guardrails (e.g., retrieval gating, edit scopes, or validation filters) remains an active direction Wang et al. (2024b).

C EXPERIMENTS DETAILS

The experiment details are given in Table 5, and hyperparameters are in Table 6.

Under identical hardware and batch configurations, the WISE baseline exhibits lower per-unit overhead. REPAIR demonstrates a similar scaling slope but with a higher intercept, primarily attributable to:

Table 7: Main results for QA on DeepSeek-R1-1.5B N: Num Edits.

Method	$N=1$				$N=30$				$N=120$				$N=1000$			
	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.	Rel.	Gen.	Loc.	OP.
DeepSeek-R1-1.5B (ZsRE)																
FT-L	0.43	0.42	0.95	0.56	0.32	0.33	0.46	0.36	0.21	0.21	0.15	0.19	0.17	0.15	0.09	0.13
FT-EWC	0.97	**0.94**	0.15	0.52	0.82	0.81	0.02	0.24	0.63	0.64	0.02	0.20	0.57	0.56	0.02	0.19
MEND	0.95	**0.94**	0.98	0.96	0.42	0.42	0.18	0.32	0.18	0.12	0.07	0.11	0.8	0.03	0.00	0.00
ROME	0.87	0.87	0.99	0.91	0.66	0.64	0.72	0.67	0.17	0.18	0.09	0.14	0.01	0.01	0.01	0.01
MEMIT-M	0.88	0.87	0.99	0.91	0.71	0.72	0.92	0.78	0.63	0.65	0.78	0.68	0.48	0.47	0.53	0.49
DEFER	0.62	0.60	0.82	0.67	0.58	0.57	0.57	0.57	0.34	0.31	0.23	0.29	0.07	0.06	0.02	0.04
GRACE	**0.98**	0.31	0.99	0.67	**0.92**	0.22	**0.98**	0.58	**0.89**	0.13	**1.00**	0.49	**0.83**	0.05	**0.94**	0.34
WISE	0.92	0.90	**1.00**	0.94	0.86	0.85	0.92	0.88	0.72	0.72	0.87	**0.77**	0.49	0.47	0.47	0.48
REPAIR	0.93	0.93	**1.00**	**0.95**	0.91	**0.89**	0.87	**0.89**↑	0.74	**0.74**	0.82	**0.77**↑	0.59	**0.57**	0.61	**0.59**↑

Table 8: Main results for QA (ZeRE) on multi-model editing with error distribution.

Method	$N=1$			$N=30$		
	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.
LLaMA-3-8B	0.94 ± 0.008	0.92 ± 0.01	$1.00^{+0.00}_{-0.02}$	0.93 ± 0.003	0.90 ± 0.003	0.87 ± 0.004
Qwen2.5-7B	0.98 ± 0.02	0.95 ± 0.03	$1.00^{+0.00}_{-0.02}$	0.93 ± 0.04	0.90 ± 0.03	0.93 ± 0.01
DeepSeek-R1	0.93 ± 0.02	0.92 ± 0.03	0.99 ± 0.01	0.91 ± 0.01	0.89 ± 0.03	0.87 ± 0.01
GPT2-XL	0.91 ± 0.03	0.92 ± 0.03	0.99 ± 0.01	0.88 ± 0.03	0.88 ± 0.02	0.84 ± 0.01
Method	$N=120$			$N=1000$		
	Rel.	Gen.	Loc.	Rel.	Gen.	Loc.
LLaMA-3-8B	0.76 ± 0.03	0.74 ± 0.02	$1.00^{+0.00}_{-0.04}$	0.68 ± 0.05	0.65 ± 0.01	0.89 ± 0.04
Qwen2.5-7B	0.81 ± 0.04	0.80 ± 0.05	0.92 ± 0.03	0.72 ± 0.05	0.70 ± 0.04	0.67 ± 0.03
DeepSeek-R1	0.74 ± 0.03	0.74 ± 0.04	0.82 ± 0.05	0.59 ± 0.02	0.57 ± 0.01	0.61 ± 0.03
GPT2-XL	0.79 ± 0.02	0.77 ± 0.01	0.80 ± 0.03	0.61 ± 0.03	0.62 ± 0.01	0.68 ± 0.02

- Distribution-aware clustering and reorganization;
- Additional forward/backward passes for in-batch distillation;
- The triggering frequency and cost of closed-loop pruning and retraining at large scales;
- The final **merging (TIES)** cost.

As N increases, the runtime curves of ROME, MEND, and FT-L exhibit significantly steeper growth, becoming substantially expensive or nearly infeasible at $N = 10^3$.

Throughput demonstrates that WISE maintains approximately ~ 1.8 edits/min at large N, followed by GRACE. REPAIR achieves $\sim$ 0.8-0.9 edits/min at scale, lower than WISE and MEMIT-M, consistent with expectations given its additional computational procedures.

Error bars (representing standard deviation across multiple runs) indicate that REPAIR exhibits slightly higher variance than WISE, attributable to fluctuations in retriggering frequency and sample distribution characteristics.

Relative overhead shows the time ratio of REPAIR to WISE increasing from $\sim 1.6\times$ to $\sim 2.2\times$ with increasing scale.

D THEORETICAL ANALYSIS AND PROOF SKETCHES

We now provide theoretical justifications for the stability and convergence of the proposed REPAIR framework. We introduce formal assumptions and derive lemmas and theorems that characterize the behavior of our method.

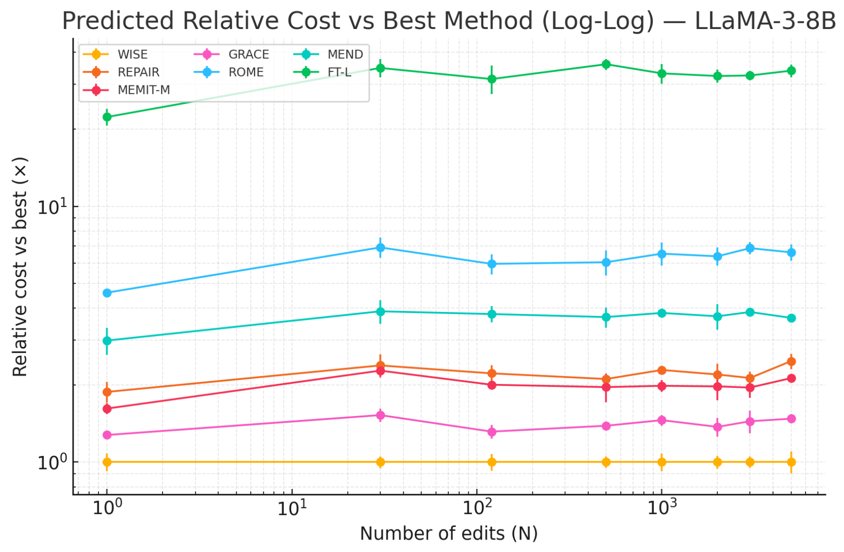

Figure 7: **Cost-Performance Assessment.** The total runtime of each method scales approximately linearly with the editing scale N, appearing as straight lines with slopes close to 1 in log-log coordinates. This indicates that the primary overhead is proportional to the number of edited entries.

D.1 Preliminaries

Assumption 1 (Standard Optimization Setting)**.** *We assume that the loss function $\mathcal{L}(\Theta)$ is L-smooth, i.e.,*

$$\|\nabla\mathcal{L}(\Theta_1) - \nabla\mathcal{L}(\Theta_2)\| \leq L\|\Theta_1 - \Theta_2\|,$$

and bounded below by $\mathcal{L}^* > -\infty$. Learning rates satisfy $\eta_t > 0$ and $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$.

D.2 Stability of Masked Gradient Updates

Lemma 1 (Norm Bound under Masked Updates)**.** *Let $g_i = \nabla_{W'_{v,i}}\mathcal{L}$ and M_i be a Bernoulli mask. Then the masked update*

$$\Delta W'_{v,i} = -\eta(M_i \odot g_i)$$

satisfies $\|\Delta W'_{v,i}\|_2 \leq \eta\|g_i\|_2$.

Proof. Since M_i is a coordinate projection, $M_i \odot g_i$ removes certain entries of g_i and never increases its magnitude. Hence $\|M_i \odot g_i\|_2 \leq \|g_i\|_2$. Multiplying by η yields the claim. □

Theorem 1 (Inter-Shard Stability)**.** *Assume masks $\{M_i\}$ are sampled independently with overlap probability ρ^2. Then in expectation,*

$$\mathbb{E}[\langle M_i \odot g_i, M_j \odot g_j\rangle] = \rho^2\langle g_i, g_j\rangle.$$

Thus, masking reduces the expected conflict between gradients of different shards.

Proof. For each coordinate p, $\Pr[M_i(p) = 1, M_j(p) = 1] = \rho^2$. Therefore, the expected inner product between masked gradients is ρ^2 times the original inner product. This reduces cross-shard interference and improves stability. □

D.3 Closed-Loop Re-trigger Analysis

Assumption 2 (Error Reduction per Re-trigger)**.** *Suppose that each re-trigger reduces the error rate of shard i by at least a fixed constant $\delta > 0$, unless it is already below the pruning threshold τ_{prune}.*

Lemma 2 (Linear Error Decrease)**.** *Let $r_i^{(n)}$ denote the error rate after n re-triggers. Under Assumption 2,*

$$r_i^{(n)} \le r_i^{(0)} - n\delta.$$

Theorem 2 (Finite-Time Convergence)**.** *If $r_i^{(0)}$ is the initial error rate, then after at most*

$$N \ge \frac{r_i^{(0)} - \tau_{\text{prune}}}{\delta}$$

re-triggers, the error rate satisfies $r_i^{(N)} \le \tau_{\text{prune}}$.

Proof. By Lemma 3, $r_i^{(N)} \le r_i^{(0)} - N\delta$. Choosing N such that $r_i^{(0)} - N\delta \le \tau_{\text{prune}}$ ensures convergence below threshold in finite time. □

D.4 Overall Convergence Intuition

Theorem 3 (Closed-Loop Stability of REPAIR)**.** *Under Assumptions 1 and 2, the iterative process combining masked updates, inner-batch distillation, and closed-loop re-trigger forms a contractive mapping in expectation. Consequently, the system converges to a stable edited state with a bounded error rate and without catastrophic forgetting.*

Proof Sketch. Masked updates reduce the variance of parameter updates, inner-batch distillation aligns outputs across samples, and re-trigger guarantees finite-time reduction of shard-level error rates. Together, these components yield monotone improvement. By standard stochastic contraction arguments, the process converges to a fixed point characterized by consistent batch predictions and an error rate below τ_{prune}. □

Lemma 3 (Zero-variance at any global minimizer)**.** *Let $\mu = \frac{1}{m}\sum_{i=1}^m o_i$ and $\mathcal{L}_{\text{var}} = \frac{1}{m}\sum_i \|o_i - \mu\|^2$. If not all o_i are equal, then $\mathcal{L}_{\text{var}} > 0$, while if $o_1 = \cdots = o_m = v$ (with $\|v\| = 1$) then $\mathcal{L}_{\text{var}} = 0$. Hence every global minimizer of $\mathcal{L}_{\text{KD}}$ on $(\mathbb{S}^{d-1})^m$ must satisfy $o_1 = \cdots = o_m =: v$.*

Lemma 4 (Unique global minimizer)**.** *Under the conclusion of Lemma 3, minimizing $\mathcal{L}_{\text{KD}}(v) = \lambda\big(1 - \langle v, u\rangle\big)$ over $\|v\| = 1$ gives the unique solution $v^\star = u$. Therefore the unique global minimizer of $\mathcal{L}_{\text{KD}}$ on $(\mathbb{S}^{d-1})^m$ is $S^\star = [u, \dots, u]$.*

Lemma 5 (Riemannian smoothness)**.** *Let $\mathcal{M} = (\mathbb{S}^{d-1})^m$ and endow each block with the canonical metric. Then $\mathcal{L}_{\text{KD}}$ is L_R-smooth on $\mathcal{M}$ in the Riemannian sense: there exists a constant*

$$\mathsf{L}_R \;\le\; \frac{2\lambda}{m} + \frac{4\vartheta}{m}$$

such that for all $S, S' \in \mathcal{M}$, $\|\operatorname{grad}\mathcal{L}_{\text{KD}}(S) - \operatorname{grad}\mathcal{L}_{\text{KD}}(S')\| \le \mathsf{L}_R \operatorname{dist}_{\mathcal{M}}(S, S')$. Sketch. *For each block o_i, $\nabla_{o_i}\mathcal{L}_{\cos} = -(\lambda/m)u$ (constant), and $\nabla_{o_i}\mathcal{L}_{\text{var}} = (2\vartheta/m)(o_i - \mu)$ with μ depending linearly on $\{o_j\}$. Projecting to the tangent space by $(I - o_i o_i^\top)$ and using the Lipschitzness of the projection map on $\mathbb{S}^{d-1}$ yields the bound.*

Theorem 4 (Convergence of cosine+variance KD on the sphere)**.** *Consider Riemannian gradient descent on $\mathcal{M} = (\mathbb{S}^{d-1})^m$:*

$$o_i^{(t+1)} \;=\; R_{o_i^{(t)}}\big(-\eta_t \operatorname{grad}_{o_i}\mathcal{L}_{\text{KD}}(S_t)\big) \quad (i = 1, \dots, m),$$

with the retraction $R_o(v) = (o + v)/\|o + v\|$. If the step sizes satisfy either (a) a constant stepsize $0 < \eta_t < 2/\mathsf{L}_R$, or (b) diminishing stepsizes $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$, then:

$$\mathcal{L}_{\text{KD}}(S_t) \downarrow \mathcal{L}_{\text{KD}}(S^\star), \qquad \|\operatorname{grad}\mathcal{L}_{\text{KD}}(S_t)\| \to 0,$$

and every limit point of $\{S_t\}$ is a Riemannian critical point. By Lemma 4, the unique global minimizer is $S^\star = [u, \dots, u]$; thus the sequence converges to $S^\star$.

Proof sketch. Riemannian smoothness (Lemma 5) on the compact manifold $\mathcal{M}$ ensures the standard descent lemma and monotone decrease for RGD under $0 < \eta < 2/\mathsf{L}_R$, implying convergence of function values and gradients to zero. By Lemmas 3–4, the only global minimizer is $S^\star$, hence all limit points coincide with $S^\star$. □

D.5 Stability of Masked Gradient Updates

Let $g_i = \nabla_{W'_{v,i}} \mathcal{L} \in \mathbb{R}^d$. A coordinate mask $M_i \in \{0,1\}^d$ acts by $(M_i \odot g_i)_p = M_i(p)\, g_{i,p}$.

Lemma 6 (Norm Bound under Masked Updates). *For any stepsize $\eta > 0$, the masked update $\Delta W'_{v,i} = -\eta(M_i \odot g_i)$ satisfies*

$$\|\Delta W'_{v,i}\|_2 \le \eta\, \|g_i\|_2.$$

Proof. Coordinate-wise, $|M_i(p)\, g_{i,p}| \le |g_{i,p}|$ because $M_i(p) \in \{0,1\}$. Hence $\|M_i \odot g_i\|_2 \le \|g_i\|_2$, and multiplying by η yields the claim. □

Theorem 5 (Inter-Shard Inner-Product Scaling). *Suppose that for each coordinate p, the masks $M_i(p), M_j(p) \in \{0,1\}$ are sampled independently with*

$$\Pr[M_i(p) = 1] = \Pr[M_j(p) = 1] = \rho, \quad 0 \le \rho \le 1,$$

and masks are independent across coordinates and independent of g_i, g_j. Then, conditional on g_i, g_j,

$$\mathbb{E}[\langle M_i \odot g_i,\, M_j \odot g_j\rangle \mid g_i, g_j] = \rho^2\, \langle g_i, g_j\rangle.$$

In particular, masking scales the expected cross-shard alignment/conflict by the factor ρ^2.

Proof. By linearity of expectation and independence, for each coordinate p, $\mathbb{E}[M_i(p)M_j(p)] = \mathbb{E}[M_i(p)]\,\mathbb{E}[M_j(p)] = \rho^2$. Summing over p yields the result. □

D.6 Closed-Loop Re-trigger Analysis

Assumption 3 (Error Reduction per Re-trigger). *Let $r_i^{(n)}$ denote the error rate of shard i after n re-triggers. There exists $\delta > 0$ such that each re-trigger reduces error by at least δ whenever $r_i^{(n)} > \tau_{\text{prune}}$.*

Lemma 7 (Piecewise-Linear Error Decrease). *Under Assumption 3, for all $n \ge 0$,*

$$r_i^{(n)} \;\le\; \max\bigl\{\, \tau_{\text{prune}},\; r_i^{(0)} - n\delta \,\bigr\}.$$

Proof. If $r_i^{(k)} > \tau_{\text{prune}}$, then $r_i^{(k+1)} \le r_i^{(k)} - \delta$. Once $r_i^{(k)} \le \tau_{\text{prune}}$, the bound $r_i^{(n)} \le \tau_{\text{prune}}$ propagates for all $n \ge k$. Unrolling gives the stated maximum form. □

Theorem 6 (Finite-Time Hitting the Pruning Threshold). *Let*

$$N_\star \;=\; \left\lceil \frac{(r_i^{(0)} - \tau_{\text{prune}})_+}{\delta} \right\rceil \quad \textit{where } (x)_+ := \max\{x, 0\}.$$

After at most $N_\star$ re-triggers, we have $r_i^{(N_\star)} \le \tau_{\text{prune}}$.

Proof. By Lemma 7, choose the smallest integer $N_\star$ such that $r_i^{(0)} - N_\star\delta \le \tau_{\text{prune}}$. Then $r_i^{(N_\star)} \le \tau_{\text{prune}}$. □

E Algorithms

The pseudocode for error feedback, network pruning, sample knowledge distillation and reintegration, and the loss-based weighted ties merge strategy is as follows:

Algorithm 1 REPAIR: Closed-Loop Lifelong Model Editing (Training)

```
Require: Pretrained model $f_{\theta_0}$; target FFN value matrix $W_v$; #shards $K$; mask ratio $\rho$; thresholds $(\epsilon, \tau_E, \tau_{\text{prune}}, \tau_{\text{correct}}, \epsilon_{\text{cons}})$; margins $(\gamma_1, \gamma_2, \gamma)$; KD weights $(\lambda, \vartheta)$ for Eq.(4); routing-loss weight $\lambda_a$; batch size $b$; optional temperature $T$ for soft KD.
 1: Initialize side memories $W'_{v,i} \leftarrow W_v$ and masks $M_i \sim$ Bernoulli$(\rho)$ for $i = 1..K$; feedback pool $\mathcal{E} \leftarrow \emptyset$; residual pool $\mathcal{R} \leftarrow \emptyset$.
 2: for each incoming edit triple $(x_e, y_e, x_{\text{loc}})$ do
 3:     $i^\star \leftarrow$ AssignShard$(x_e)$                         ▷ Shard assignment by activation score
 4:     $\mathcal{B} \leftarrow$ FormBatches$(\{x_e\} \cup \mathcal{R},\ b)$          ▷ Distribution-aware batching
 5:     for each batch $B = \{x^{(0)}, \ldots, x^{(b-1)}\} \in \mathcal{B}$ do
 6:         $i \leftarrow$ AssignShard$(x^{(0)})$                      ▷ Target shard for this batch
 7:         $L_{\text{edit}} \leftarrow$ AutoregCE$(B)$                    ▷ Autoregressive cross-entropy
 8:         $L_{\text{KD}} \leftarrow$ IntraBatchKD$(B, \lambda, \vartheta, T)$          ▷ Eq.(4); optional soft KD
 9:         $L_{\text{act}} \leftarrow$ RoutingMargin$(B, \gamma_1, \gamma_2, \gamma)$                  ▷ Eq.(7)
10:         $L_{\text{batch}} \leftarrow L_{\text{edit}} + \lambda_a L_{\text{act}} + L_{\text{KD}}$
11:         MaskedUpdate$(W'_{v,i}, M_i, L_{\text{batch}})$        ▷ $W'_{v,i} \leftarrow W'_{v,i} - \eta(M_i \odot \nabla L)$
12:         FilterAndRecluster$(B, \epsilon_{\text{cons}}, \mathcal{R})$     ▷ Move high-$L_{\text{KD}}$ samples to residual pool
13:     end for
14:     $(\hat{y}, c) \leftarrow$ Evaluate$(x_e, y_e)$                   ▷ $c \in \{0, 1\}$ indicates success
15:     if $c = 0$ then
16:         $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_e, y_e)\}$
17:     end if
18:     if $|\mathcal{E}| > \tau_E$ or $\max_i$ ErrorRate$(\mathcal{E}, i) > \tau_{\text{prune}}$ then
19:         ReTrigger$(\mathcal{E})$                     ▷ Prune worst shard, rebuild, and retrain
20:     end if
21: end for
22: LossAwareTIESMerge$(\{W'_{v,i}\}_{i=1}^{K},\ W_v)$               ▷ Loss-aware weighted TIES merge
```

Algorithm 2 REPAIR Inference with Dual-Memory Routing

```
 1: function RouteAndPredict$(x)$
 2:     compute $a(x) \leftarrow$ FFNActivation$(x)$        ▷ Activation $A(x)$ at the target FFN layer
 3:     for $i = 1..K$ do
 4:         $\Delta_{\text{act}}^{(i)}(x) \leftarrow \| a(x) \cdot (W'_{v,i} - W_v) \|_2$
 5:     end for
 6:     if $\max_i \Delta_{\text{act}}^{(i)}(x) \le \epsilon$ then
 7:         return $f_{\theta_0}(x; W_v)$                              ▷ Route to main memory
 8:     else
 9:         $i^\star \leftarrow \arg\max_i \Delta_{\text{act}}^{(i)}(x)$
10:         return $f_{\theta_0}(x; W'_{v,i^\star})$                          ▷ Route to side memory $i^\star$
11:     end if
12: end function
```

Algorithm 3 Subroutines

```
 1: function AssignShard($x$)
 2:     $a \leftarrow$ FFNActivation($x$);  $\Delta^{(i)} \leftarrow \|a \cdot (W'_{v,i} - W_v)\|_2,\ i = 1..K$
 3:     return $\arg\max_i \Delta^{(i)}$                        ▷ Use the most active shard during training
 4: end function
 5: function FormBatches($S,\ b$)                               ▷ Distribution-aware batching
 6:     $o_i \leftarrow \mathrm{Norm}\big(\mathrm{ModelFeat}(x^{(i)})\big)$ for $i = 0, \ldots, b-1$
 7:     Greedy seeding: pick $x^{(0)} = \arg\max_{x \in S} \frac{1}{|S|} \sum_{x'} \cos(o(x), o(x'))$
 8:     Build $B \leftarrow \{x^{(0)}\} \cup$ Top-($b-1$) nearest by cosine; remove $B$ from $S$
 9:     Repeat until $S$ is empty; return list of batches $\mathcal{B}$
10: end function
11: function AutoregCE($B$)                                     ▷ Autoregressive edit loss $L_{\text{edit}}$
12:     $L \leftarrow 0$
13:     for $x \in B$ with target sequence $y$ do
14:         $L \leftarrow L - \sum_{t=1}^{|y|} \log p_\theta(y_t \mid y_{<t}, x)$
15:     end for
16:     return $L/|B|$
17: end function
18: function IntraBatchKD($B, \lambda, \vartheta, T$)            ▷ Eq.(4); optional soft-KD
19:     Compute $o_i \leftarrow \mathrm{Norm}\big(\mathrm{ModelFeat}(x^{(i)})\big)$ for $i = 0, \ldots, b-1$
20:     $L_{\text{cos}} \leftarrow \frac{1}{b-1} \sum_{i=1}^{b-1} \Big(1 - \frac{o_i^\top o_0}{\|o_i\|\|o_0\|}\Big)$
21:     $o_{\text{mean}} \leftarrow \frac{1}{b} \sum_{i=0}^{b-1} o_i$;  $L_{\text{var}} \leftarrow \frac{1}{b} \sum_{i=0}^{b-1} \|o_i - o_{\text{mean}}\|_2^2$
22:     $L \leftarrow \lambda\, L_{\text{cos}} + \vartheta\, L_{\text{var}}$
23:     if $T > 0$ then                                         ▷ Optional: KL distillation for added stability
24:         Get logits $z_i$; $p_i = \mathrm{softmax}(z_i/T)$;  $L \leftarrow L + \frac{1}{b-1} \sum_{i=1}^{b-1} \mathrm{KL}(p_0 \| p_i)$
25:     end if
26:     return $L$
27: end function
28: function RoutingMargin($B, \gamma_1, \gamma_2, \gamma$)        ▷ Eq.(7)
29:     $L \leftarrow 0$
30:     for each edit sample $x_e \in B$ do
31:         sample unrelated $x_i$;  compute $\Delta_e = \mathrm{ActDelta}(x_e)$, $\Delta_i = \mathrm{ActDelta}(x_i)$
32:         $L \leftarrow L + \max(0, \Delta_i - \gamma_1) + \max(0, \gamma_2 - \Delta_e) + \max(0, \gamma - (\Delta_e - \Delta_i))$
33:     end for
34:     return $L/|B|$
35: end function
36: function MaskedUpdate($W'_{v,i}, M_i, L$)               ▷ Masked gradient to reduce cross-shard interference
37:     $g \leftarrow \nabla_{W'_{v,i}} L$;  $g_{\text{m}} \leftarrow M_i \odot g$          ▷ $M_i \in \{0,1\}^{\text{shape}(W_v)}$
38:     $W'_{v,i} \leftarrow \mathrm{OptimizerStep}(W'_{v,i}, g_{\text{m}})$                 ▷ SGD/Adam, etc.
39: end function
40: function FilterAndRecluster($B, \epsilon_{\text{cons}}, \mathcal{R}$)
41:     for $x \in B$ do
42:         $\ell_{\text{KD}}(x) \leftarrow$ per-sample KD vs. $x^{(0)}$
43:         if $\ell_{\text{KD}}(x) \geq \epsilon_{\text{cons}}$ then
44:             move $x$ to $\mathcal{R}$
45:         end if
46:     end for
47:     return
48: end function
49: function Evaluate($x_e, y_e$)
50:     $\hat{y} \leftarrow \mathrm{RouteAndPredict}(x_e)$;  $c \leftarrow \mathbf{1}[\hat{y} = y_e]$
51:     return $(\hat{y}, c)$
52: end function
```

Algorithm 4 Subroutines2

```
 1: function ErrorRate($\mathcal{E}, i$)                                          ▷ Error rate for shard $i$
 2:     $\mathcal{E}_i \leftarrow \{x \in \mathcal{E} \mid \arg\max_j \Delta^{(j)}_{\text{act}}(x) = i\}$
 3:     $r_i \leftarrow \frac{|\{x \in \mathcal{E}_i \mid \text{Correctness}(x) \le \tau_{\text{correct}}\}|}{|\mathcal{E}_i|}$
 4:     return $r_i$
 5: end function
 6: procedure ReTrigger($\mathcal{E}$)                              ▷ Closed-loop pruning and retraining
 7:     $j \leftarrow \arg\max_i$ ErrorRate($\mathcal{E}, i$)                    ▷ Identify worst-performing shard
 8:     Remove or reinitialize shard $j$: $W'_{v,j} \leftarrow W_v + \sigma_{\text{init}} \cdot \mathcal{N}(0, 1)$; resample $M_j$
 9:     Build $\mathcal{E}_{\text{retrain}}$ from $\mathcal{E}$; form batches; retrain shards via MaskedUpdate + IntraBatchKD
10: end procedure
11: function LossAwareTIESMerge($\{W'_{v,i}\}, W_v$)              ▷ Loss-aware weighted TIES merge
12:     For each shard $i$: $\tau_i \leftarrow W'_{v,i} - W_v$; compute training loss $L_i$ on its assigned data
13:     $w_i \leftarrow \frac{e^{-\alpha L_i}}{\sum_j e^{-\alpha L_j}}$
14:     for each parameter index $p$ do
15:         $S \leftarrow \{(i, \tau_i[p], w_i)\}_{i=1}^{K}$
16:         if all $\tau_i[p]$ share the same sign then
17:             $\delta[p] \leftarrow \sum_i w_i \, \tau_i[p]$                        ▷ Consistent signs: weighted sum
18:         else
19:             $i^\star \leftarrow \arg\max_i \{w_i \, |\tau_i[p]|\}$; $\delta[p] \leftarrow \tau_{i^\star}[p]$   ▷ Conflict: keep most trustworthy shard
20:         end if
21:     end for
22:     $W_v \leftarrow W_v + \delta$; return $W_v$
23: end function
24: function FFNActivation($x$)
25:     return activation $A(x)$ at the target FFN layer
26: end function
27: function ActDelta($x$)
28:     return $\max_i \|A(x) \cdot (W'_{v,i} - W_v)\|_2$
29: end function
30: function ModelFeat($x$)
31:     return feature used for similarity (e.g., $A(x)$ or last-token state)
32: end function
33: function Norm($v$)
34:     return $v / \|v\|_2$
35: end function
36: function Correctness($x$)
37:     return predicted correctness score for $x$
38: end function
```