# REPAIR: Robust Editing via Progressive Adaptive Intervention and Reintegration

Anonymous ACL submission

## Abstract

Large language models (LLMs) are increasingly deployed in settings where factual knowledge changes after training, making post-deployment correction necessary. Full retraining and broad fine-tuning are costly and can introduce unintended side effects, while model editing offers a more targeted alternative. However, existing editors still struggle with sequential edit accumulation, weak generalization beyond the edited prompt, and insufficient monitoring of locality failures. We introduce REPAIR (**R**obust **E**diting via **P**rogressive **A**daptive **I**ntervention and **R**eintegration), a lifelong editing framework that combines side-memory editing with closed-loop error feedback, distribution-aware batch reassembly, inner-batch knowledge distillation, and loss-aware subspace merging. Across QA, WikiBigEdit, and hallucination-mitigation settings, REPAIR improves the reliability–generalization–locality trade-off over representative direct, hypernetwork-based, and memory-based editors; for example, at $N = 1000$, REPAIR improves OP from 0.47 to 0.73 over WISE on LLaMA-3-8B (ZsRE), from 0.51 to 0.69 on Qwen2.5-7B (ZsRE), and from 0.48 to 0.63 on DeepSeek-R1-1.5B (WikiBigEdit). These results suggest that scalable LLM editing benefits from adaptive monitoring and conservative knowledge reintegration rather than treating each edit as an isolated update. The code is available at `https://anonymous.4open.science/r/REPAIR`.

## 1 Introduction

Large language models (LLMs) have demonstrated strong capabilities across a wide range of tasks, but their parameters remain largely fixed after training. As a result, deployed models cannot directly correct outdated facts, hallucinated associations, or newly acquired knowledge without an additional update mechanism. Lifelong model editing addresses this need by enabling efficient, localized modifications

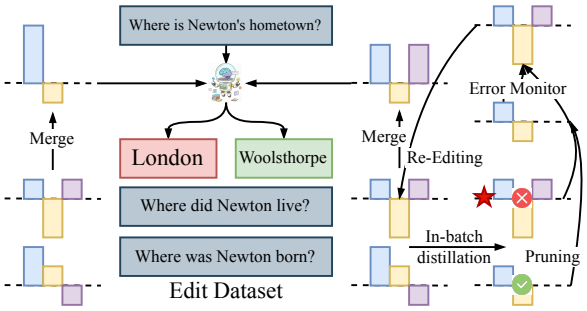


Figure 1: **Problems and our solutions**. REPAIR achieves closed-loop feedback, fine-grained knowledge integration, weighted knowledge merging, and robust editing performance.

that keep models current while preserving unrelated capabilities (Wang et al., 2024b). Compared with full retraining or broad fine-tuning, editing aims to make fine-grained behavioral changes that take effect immediately and avoid unnecessary interference with non-target knowledge.

Despite steady progress, three limitations remain central to practical lifelong editing, as summarized in Figure 1. **(1) Sequential stability and knowledge fusion.** When edits accumulate, models may exhibit routing instability, interference among nearby edits, or broader behavioral collapse (Gupta et al., 2024; Cohen et al., 2024). Semi-parametric editors such as SERAC (Mitchell et al., 2022b) and discrete key–value adaptors such as GRACE (Hartvigsen et al., 2023) improve edit isolation and support longer streams, but still introduce trade-offs among scope, auditability, and memory growth (Mitchell et al., 2022b; Hartvigsen et al., 2023). In addition, how edited knowledge should be consolidated remains underexplored, although fusion is a stage where information can be diluted or lost (Wang et al., 2024a). **(2) Few-shot generalization.** Under data-scarce conditions, editors can overfit to the exact edit prompt and fail on paraphrases or nearby contexts, motivating

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., 2022a) | ✓ | × | ✓ | × | × | × | × | × | × |
| 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 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

gradient-transformation approaches such as MEND (Mitchell et al., 2022a) and broader evaluations of edit generalization (Mitchell et al., 2022a; Wang et al., 2024b). **(3) Open-loop and distribution-agnostic optimization.** Many editing pipelines optimize fixed or indiscriminate batches without monitoring whether later edits introduce locality failures or ripple effects on related knowledge (Cohen et al., 2024; Wang et al., 2024b). These issues expose a persistent trade-off among reliability, specificity, and scalability.

To address these challenges, we propose REPAIR (**R**obust **E**diting via **P**rogressive **A**daptive **I**ntervention and **R**eintegration). REPAIR combines three mechanisms. **(1) Closed-loop feedback with dynamic memory management** monitors edit outcomes and selectively removes or reinitializes underperforming memory units, stabilizing routing as edit streams grow. **(2) Distribution-aware optimization** groups similar samples and applies inner-batch distillation so that edits are learned from coherent local distributions rather than from indiscriminate batches. **(3) Loss-aware knowledge fusion** consolidates edited subspaces with weights derived from their training losses, reducing the risk that unreliable updates dilute more stable knowledge.

Table 1 positions REPAIR relative to representative editing families in terms of memory design, lifelong-editing attributes, and training-time behaviors. Unlike methods that rely only on direct parameter updates or isolated external memories, REPAIR couples a dual-memory mechanism with parametric editing, feedback-driven pruning, inner-batch distillation, and loss-aware merging. Table 2 illustrates the type of failure REPAIR targets: previous editors may correct the edited prompt but repeat stale answers, fail on paraphrases, or affect

Table 2: **Failure case 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.

| **Method/Prompt** | **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 ✓ | |

adjacent unrelated facts. The main contributions are as follows:

- We identify three challenges for lifelong model editing: stability under sequential edits, generalization in few-shot settings, and the lack of feedback in distribution-agnostic editing pipelines.
- We propose REPAIR, a framework that integrates dual-memory editing with closed-loop feedback, distribution-aware batch reassembly, inner-batch distillation, and loss-aware subspace merging.
- We evaluate REPAIR on multiple model families, including LLaMA-3, Qwen2.5, DeepSeek-R1-1.5B, and GPT-2-XL, showing improved reliability–generalization–locality trade-offs over representative baselines, especially under large edit streams.

## 2 Methodology

We propose **REPAIR**, a closed-loop framework for lifelong model editing. As shown in Figure 2, REPAIR augments side-memory editing with three components: (1) error feedback and dynamic mem-

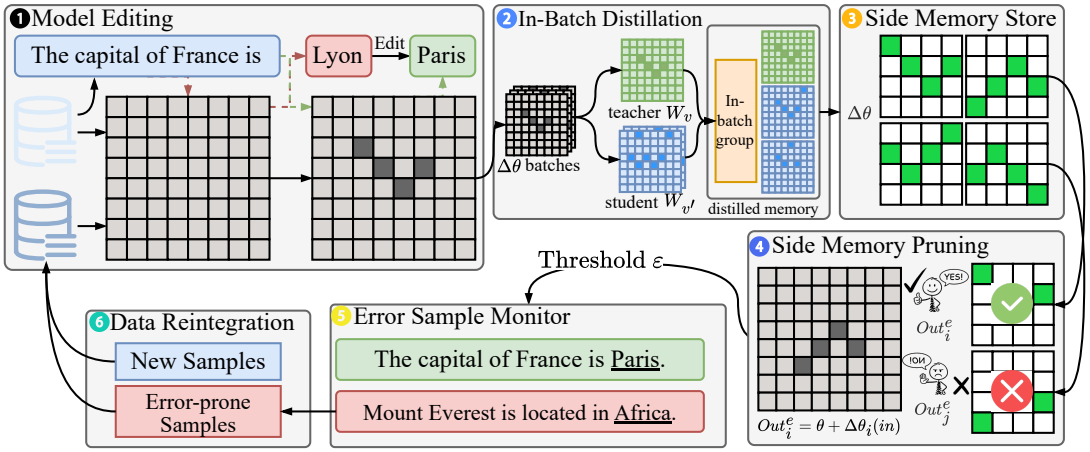


Figure 2: **The overall structure of REPAIR.** An edit, such as changing the capital of France from "Lyon" to "Paris," is stored as a parameter update, $\Delta\theta$, in the Side Memory. An Error Sample Monitor evaluates the performance of each edit ($Out_i^e$). If the error rate, $Err_{thresh}$, for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

ory management for identifying unreliable shards; (2) distribution-aware batch reassembly with inner-batch knowledge distillation for reducing conflicts among heterogeneous edits; and (3) loss-aware weighted merging for consolidating reliable edited subspaces.

### 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 $\theta_T$ that optimize the multi-objective trade-off:

$$\begin{aligned}\theta_t = \arg\min_{\theta}\ & \alpha \frac{1}{N}\sum_{i=1}^{N} \ell\big(f_\theta(\cdot \mid x_i^{(t)}), y_i^{(t)}\big) \\ & + \beta \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{x' \sim \mathcal{G}(x_i^{(t)})}\Big[\ell\big(f_\theta(\cdot \mid x'), y_i^{(t)}\big)\Big] \\ & + \gamma\, \mathbb{E}_{x\sim\mathcal{U}}\Big[\mathrm{KL}\big(f_{\theta_{t-1}}(\cdot \mid x) \,\|\, f_\theta(\cdot \mid x)\big)\Big] \\ & + R(\theta, \theta_{t-1}).\end{aligned} \tag{1}$$

where $(\alpha, \beta, \gamma)$ 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 block 1 of Figure 2, REPAIR follows a dual-memory design at selected feed-forward layers. For the value matrix $\mathbf{W}_v$ of a target FFN layer, we initialize a side-memory pool $M_s$ as a copy of the original value matrix: $M_s^{(0)} = W_v$. When the side memory is selected, the corresponding FFN output is computed as: $o_s = \phi(f^T W_k) \cdot M_s$, where $\phi$ denotes the non-linear activation function, and $o_s$ represents the FFN output based on the side memory ((Wang et al., 2024a)).

During inference, each memory pool is scored by how strongly its update changes the layer activation:

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

where $\mathcal{A}(\cdot)$ is the activation at the corresponding FFN layer. If $\max_i \Delta_{\text{act}}^{(i)}(x) \le \varepsilon$, inference uses the main memory $W_v$; otherwise, it routes to the side-memory pool with the largest activation score. To make this routing decision discriminative, we use a margin-based objective that separates edited, paraphrased, and locality samples:

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

where $\tau$ is a routing threshold, $\mathcal{E}$ denotes edited samples, $\mathcal{U}$ denotes unrelated locality samples, and $\mathcal{G}$ denotes paraphrased or neighboring edit contexts. This selective activation mechanism aims to retrieve edited knowledge only when the input is sufficiently related to an edit, thereby reducing interference with unrelated behavior.

### 2.3 Distribution-Aware inner-batch Knowledge Distillation

Block 2 of Figure 2 addresses the conflict introduced by heterogeneous edit batches. Given

a candidate edit batch $\mathcal{E} = \{x_1, x_2, \ldots, x_n\}$, we denote the feature representations by $o_i = \mathrm{Norm}(f_\theta(x_i)), \quad i = 1, \ldots, n$. To improve update consistency, REPAIR first groups mutually similar samples into homogeneous batches and then performs inner-batch knowledge distillation. For a batch $B = \{x^{(0)}, x^{(1)}, \ldots, x^{(b-1)}\}$, the first sample $x^{(0)}$ is treated as the *teacher*, while the remaining samples act as *students*. We define the inner-batch 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 $\mathcal{L}_{\text{kd}}$ aligns the representations of semantically similar samples, reducing conflicts when they update the same parameters. The variance term prevents the batch from collapsing into an overly uniform representation.

If a sample remains poorly aligned with the batch (i.e., its $\mathcal{L}_{\text{kd}}$ remains high), REPAIR treats it as distributionally inconsistent with the current group. Such samples are removed and reclustered with other residual samples to form more coherent groups. Formally, the reassembled batch is

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

where $\epsilon$ controls the required inner-batch consistency. This procedure encourages sequential parameter edits to operate on aligned knowledge groups, improving update stability. The convergence analysis is provided in Appendix D.11 and Appendix D.6.

### 2.4 Closed-Loop Error Feedback and Memory Pruning

Block 4 of Figure 2 introduces feedback after each editing cycle. REPAIR maintains a feedback pool $\mathcal{E}$ of error-prone samples and compares model outputs against a correctness threshold $\tau_{\text{correct}}$. For shard $i$, we define $\mathcal{E}_i = \{x \in \mathcal{E} \mid i^*(x) = i\}$ and compute the shard-level error rate as the proportion of failed samples assigned to that shard: $r_i^{\text{pool}} = \frac{|\{x \in \mathcal{E}_i \mid a(x) \le \tau_{\text{correct}}\}|}{|\mathcal{E}_i|}$

When $r_i > \tau_{\text{prune}}$ or $|\mathcal{E}| > \tau_E$, REPAIR performs two steps. **(1) Memory pool screening and pruning:** it identifies the side-memory pool with the highest error rate, $j = \arg\max_i r_i^{\text{pool}}$, and removes or reinitializes that pool. **(2) Sample reintegration and retraining:** it recombines the remaining error samples into $\mathcal{E}_{\text{retrain}}$ and retrains the affected side memories. This feedback loop allows the editor to correct underperforming memory units instead of leaving errors to accumulate. The finite-time convergence argument is provided in Appendix D.6.

### 2.5 Merging with Weighted TIES

Block 3 of Figure 2 consolidates multiple edited shards. After several updates, each shard $W'_{v,i}$ induces a delta $\tau_i = W'_{v,i} - W_v$. We merge these deltas with a weighted TIES operator (Yadav et al., 2023): $W'_v \leftarrow W_v + \omega_i \,\mathrm{TIES}\left(\{\tau_i\}_{i=1}^{k}; W_v\right)$.

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:

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

For shard $i$, consider $k$ edited subspaces $\{\theta_1, \ldots, \theta_k\}$, each trained on a subset of samples $\mathcal{E}_i$. Let the average training loss of subspace $\theta_i$ 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 set the merging weight 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 merged parameters are then obtained by weighted averaging: $\theta = \sum_{i=1}^{M} w_i\,\theta_i$. This loss-aware merging gives larger influence to subspaces that better fit their assigned edits, reducing the chance that unreliable updates dominate consolidation.

### 2.6 Theoretical Guarantees

We summarize two properties that directly support the design choices above; full statements and proofs are deferred to Appendix D (the analysis treats $\mathcal{L}$ as $L$-smooth and assumes step sizes $\eta_t$ with $\sum_t \eta_t = \infty$, $\sum_t \eta_t^2 < \infty$).

**Proposition 1 (Inter-shard interference under masking).** Let $g_i, g_j$ be the gradients of two shards and let $M_i, M_j \in \{0,1\}^d$ be coordinate masks with $\Pr[M_i(p)=1] = \Pr[M_j(p)=1] = \rho$, sampled independently across shards and coordinates. Then $\mathbb{E}[\langle M_i \odot g_i,\, M_j \odot g_j\rangle \mid g_i, g_j] = \rho^2\langle g_i, g_j\rangle$, so the expected cross-shard inner product is contracted

by a factor $\rho^2$. This justifies sparse, shard-local updates as a mechanism for limiting interference between concurrently edited shards (Theorem D.13).

**Proposition 2 (Finite-time decrease of shard error).** Let $r_i^{(n)}$ be the error rate of shard $i$ after $n$ closed-loop re-triggers. If each re-trigger reduces $r_i$ by at least $\delta > 0$ whenever $r_i > \tau_{\text{prune}}$, then $r_i^{(n)} \leq \max\{\tau_{\text{prune}}, r_i^{(0)} - n\delta\}$, and after at most $N_\star = \lceil (r_i^{(0)} - \tau_{\text{prune}})_+ / \delta \rceil$ re-triggers we have $r_i^{(N_\star)} \leq \tau_{\text{prune}}$ (Theorem D.16). The two propositions together explain why closed-loop feedback combined with masked, shard-local updates yields bounded shard error without uncontrolled cross-shard drift, even when edits accumulate over long streams.

## 3 Experiments

The experiments are organized around six questions: **Q1**: Do closed-loop feedback, discriminative pruning, and distribution-aware reintegration improve reliability, generalization, and locality? **Q2**: Does the method transfer from QA editing to hallucination mitigation? **Q3**: Is the method effective across model families and parameter scales? **Q4**: Does it remain competitive under WikiBigEdit distribution shift? **Q5**: Can it maintain stability under large sequential edit streams? **Q6**: Which components and hyperparameters explain the observed behavior?

### 3.1 Experimental Setup

**Datasets and Models**. We evaluate autoregressive LLMs because their causal generation makes edit effects traceable at the prompt level. The main experiments use LLaMA-3-8B, GPT-2-XL, Qwen2.5-7B, and DeepSeek-R1-1.5B. We use ZsRE for closed-book QA editing, WikiBigEdit for distribution-shifted editing, and SelfCheckGPT-style hallucination data for generation robustness. Dataset statistics are provided in Appendix 8.

**Baselines.** We consider three families of methods: **Direct Parameter Editors** that 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** that use an auxiliary network to generate parameter updates at inference (e.g., **MEND** (Mitchell et al., 2022a)); and **External Memory-Based Editors** that store edits in external memory and retrieve them via a routing mechanism (e.g., **SERAC** (Mitchell et al., 2022b), **GRACE** (Hartvigsen et al., 2023), **WISE** (Wang et al., 2024b)).

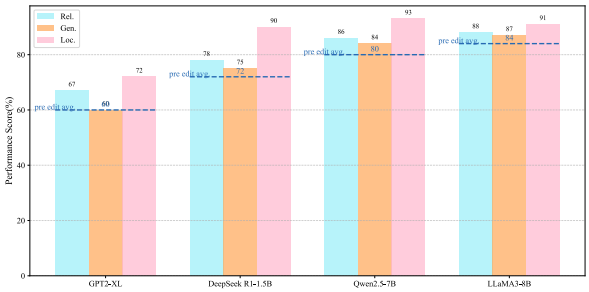


Figure 3: Average Editing Performance of WikiBigEdit Across Different Models

**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. We provide the code to facilitate reproducibility at `https://anonymous.4open.science/r/REPAIR`. The specific hyperparameter settings are detailed in Appendix C.

**Evaluation Metrics.** Each edited instance contains an edit prompt $k_e$, an unrelated locality prompt-answer pair $k'_e$, and a rephrased prompt $k_{loc}$. We evaluate the standard editing trilemma with edit reliability, **Rel** $= \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}(f_{\omega_N}(\mathbf{x}_e^n) = \mathbf{y}_e^n)$, rephrase accuracy : **Gen** $= \frac{1}{N}\sum_{n=1}^{N} \mathbf{l}(f_{\omega_N}(\mathbf{x}'^n_e) = \mathbf{y}_e^n)$, locality : **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 summarize overall editing performance. This score is sensitive to failures in any one dimension and therefore reflects the reliability–generalization–locality trade-off. **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 hallucination mitigation, we report perplexity (PPL), where lower values indicate more confident target continuations and lower hallucination risk. **PPL** $= \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|\text{context}_i)\right)$

### 3.2 Main Results

Table 3 addresses **Q1**, **Q4**, and **Q5** by comparing edit streams of $N = 1, 30, 120, 1000$. At $N = 1$, several editors are competitive because the risk of edit interaction is limited. As $N$ increases, the trade-offs become clearer: fine-tuning variants lose locality, ROME and MEND degrade sharply under

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**↑ | 0.76 | 0.74 | **1.00** | **0.83**↑ | 0.68 | 0.65 | 0.89 | **0.73**↑ |
| 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** ↑ | 0.93 | **0.90** | 0.93 | **0.92**↑ | 0.81 | **0.80** | 0.92 | **0.84**↑ | 0.72 | **0.70** | **0.67** | **0.69**↑ |
| 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**↑ | 0.84 | **0.83** | **0.91** | **0.86**↑ | 0.71 | **0.69** | **0.90** | **0.76**↑ | **0.58** | **0.54** | **0.81** | **0.63**↑ |

sequential edits, and GRACE preserves locality but has weak rephrase generalization. WISE remains a strong memory-based baseline, but its reliability and generalization decrease at larger edit counts. REPAIR maintains higher OP at medium and large scales across the reported model families, indicating that feedback-driven pruning and reintegration are most useful when edit conflicts accumulate.

For **Q2**, Table 4 evaluates hallucination mitigation on LLaMA-3-8B. REPAIR reaches the lowest PPL among the compared methods at $N = 120$ and $N = 500$ while keeping locality at 1.00. This contrasts with direct editors such as MEND and ROME, whose PPL becomes unstable at larger edit counts, and with methods that preserve locality but do not consistently reduce PPL.

**Reasoning ability retention (MMLU).** To verify that sequential editing does not compromise general reasoning capability, we evaluate the model before and after $N = 100$ edits on MMLU (57 subjects). MMLU accuracy changes only slightly ($0.6591 \rightarrow 0.6332$), indicating limited degradation of general reasoning ability.

For **Q3** and **Q4**, Table 3 and Figure 3 show that REPAIR remains competitive across LLaMA-3-8B, Qwen2.5-7B, and DeepSeek-R1-1.5B, including the WikiBigEdit setting. Smaller models can be easier to modify locally, but they have less parameter redundancy and therefore degrade more quickly as the number of edits grows. Larger models tend to preserve locality better and transfer successful edits more reliably across contexts. At $N = 120$, MEMIT-M and WISE remain competitive on some reliability scores, but at $N = 1000$ their OP drops more sharply than REPAIR. The error distribution in Table 11 provides additional variance information across model families.

Figure 4 provides a mechanism-level view of **Q1**. For memory-based editors, routing quality depends on whether activation scores separate edited inputs,

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

| Method | $N = 1$ | | $N = 30$ | | $N = 120$ | | $N = 500$ | |
|---|---|---|---|---|---|---|---|---|
| | Rel. (*PPL* ↓) | Loc. (↑) | Rel. (↓) | Loc. (↑) | Rel. (↓) | Loc. (↑) | Rel. (↓) | Loc. (↑) |
| **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** |

rephrased edit inputs, and locality inputs. The pre-distillation plots show that these groups can overlap under large continual-edit settings, which increases the risk of routing to an irrelevant memory shard. After applying inner-batch distillation and sample reintegration, the edit-related and locality-related activation patterns become more separable, supporting the routing behavior reflected in the Rel./Gen./Loc. metrics.

### 3.3 Ablation Studies

Figure 5 addresses **Q6** by comparing component behavior across edit scales. The relative importance of each component changes with $N$: pruning and feedback are more visible at smaller scales, where a few faulty shards can dominate outcomes, whereas distribution-aware recognition and knowledge distillation become more important as edit interactions increase. The overhead analysis in Appendix Figure 7 complements this result by showing the computational cost of these additional controls.

**Homogeneous vs. heterogeneous batching.** To test whether distribution-aware batching matters beyond simply increasing optimization exposure, we disable similarity-based clustering and train on heterogeneous batches. Table 5 shows that heterogeneous batching improves rewrite accuracy but substantially reduces locality, indicating stronger

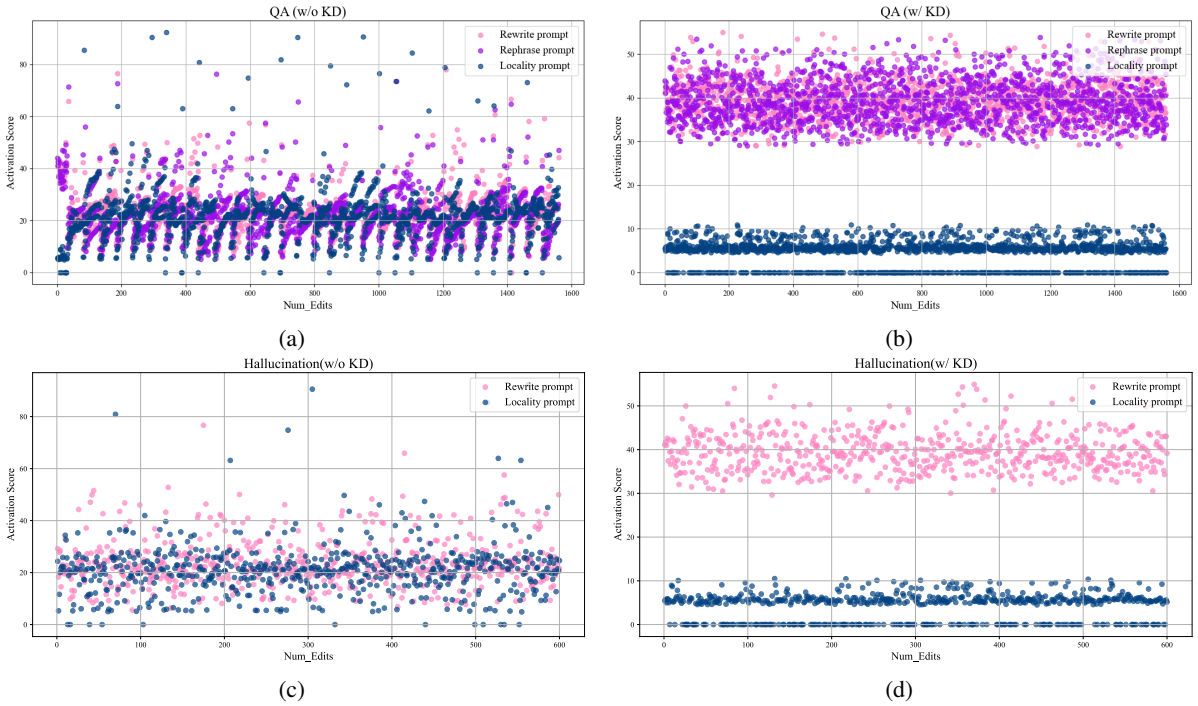


Figure 4: **Activation Score Visualization**. Results on LLaMA-3 for the WikiBigEdit dataset (N=1550) for the QA task and the SelfCheckGPT dataset for hallucination (N=600).

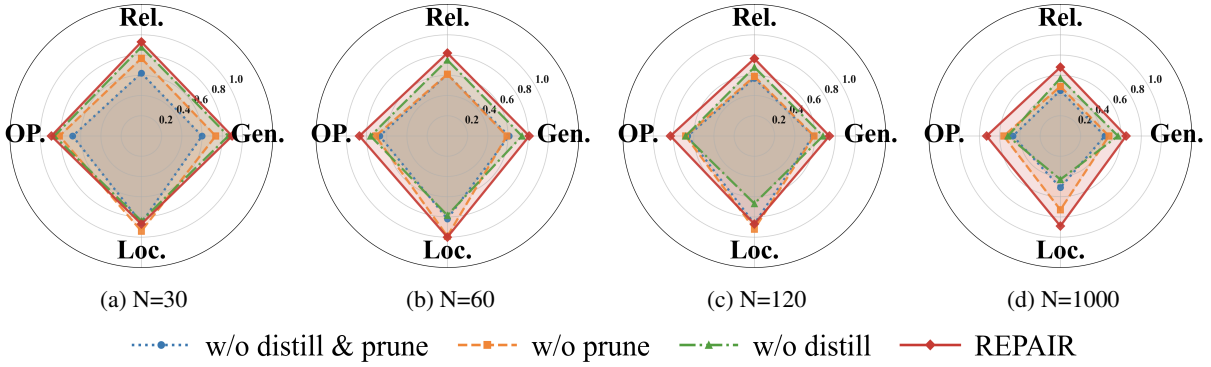


Figure 5: **Performance comparison of different components.** Each radar chart shows performance on four metrics: Rel., gen., loc., and OP. on Qwen2.5 with ZsRE.

Table 5: **Effect of heterogeneous batches on Qwen2.5 QA editing.** Heterogeneous batching increases rewrite accuracy but reduces locality.

| Batch Type | Rewrite Acc. | Locality |
|---|---|---|
| Homogeneous (ours) | 23.1% | 42.5% |
| Heterogeneous | 43.5% | 29.4% |

interference among unrelated edits. Appendix C.1 further checks that the features used for clustering encode meaningful within-task similarity.

Figure 6 reports sensitivity to the number of subspaces, updated-parameter amount, error threshold, and maximum iteration count. Intermediate settings perform best: overly permissive thresholds keep low-quality samples in training, whereas overly aggressive correction can spend computation on edits with limited marginal benefit and increase overfitting risk. The one-dimensional threshold sweep in Appendix C.2 further shows that locality improves once the filtering threshold is sufficiently high.

## 4 Conclusion

We presented **REPAIR**, a lifelong model editing framework that combines closed-loop error feedback, distribution-aware distillation, and loss-aware subspace merging. The empirical results show that these components improve the reliability–generalization–locality trade-off across QA, WikiBigEdit, and hallucination-mitigation settings, particularly when edits accumulate over long streams. By monitoring error-prone shards, regrouping inconsistent samples, and weighting more reliable subspaces during fusion, REPAIR offers a practical direction for continually updating LLMs without relying on full retraining after each knowledge change.

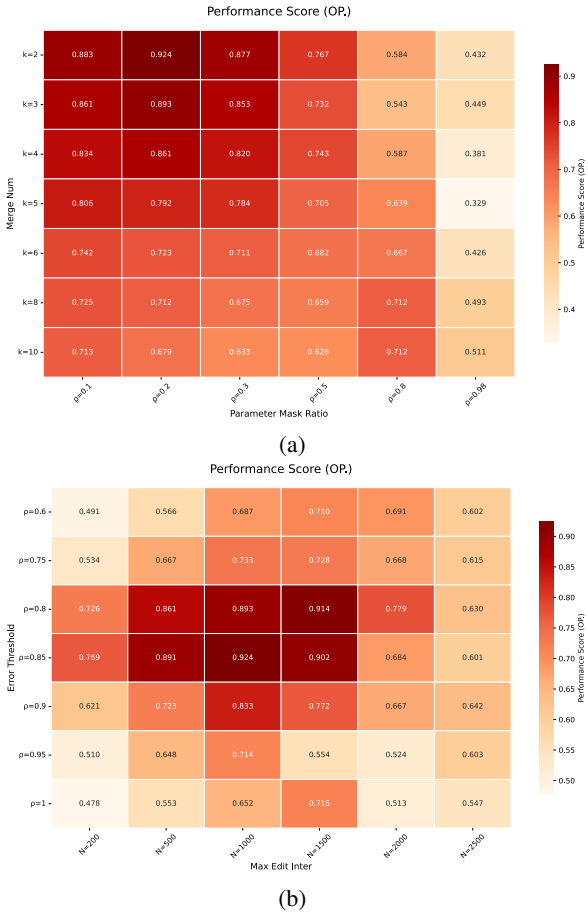


Figure 6: **Performance heatmap for the N=120 QA task on the LLaMA3 model.** Figure (a) shows the sensitivity analysis of two hyperparameters: the number of subspaces and the amount of updated parameters; Figure (b) analyzes the impact of error threshold and maximum iteration count on performance, with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

## Limitations

Several limitations remain. First, closed-loop feedback and dynamic memory management introduce additional bookkeeping; while this overhead is modest in our evaluated settings, it may become more substantial in extremely long edit streams. Second, our experiments focus on factual editing benchmarks (ZsRE, WikiBigEdit, and a hallucination set) and a limited set of open-source LLM backbones; multilingual, multimodal, and proprietary models require further evaluation. Third, loss-aware merging assumes that lower training loss is a useful proxy for more reliable edits, which may not hold under adversarial or highly noisy edit streams. Finally, REPAIR requires access to model parameters and editable internal layers, and therefore does not address black-box models available only through APIs.

## Ethical Considerations

This work studies lifelong model editing for LLMs, aiming to enable efficient, targeted updates (e.g., correcting outdated or incorrect facts) while reducing unintended side effects on unrelated behaviors. Positive impacts include lower computational cost compared to retraining, faster correction of factual errors, and improved maintainability of deployed systems in dynamic knowledge environments. However, model editing methods may also introduce risks. First, they can be misused to inject misinformation, bias, or malicious instructions into a deployed model, especially if editing interfaces are exposed without appropriate access control and auditing. Second, imperfect locality could unintentionally alter model behavior on non-target content, potentially affecting downstream users unevenly across domains or demographic groups. REPAIR is designed to mitigate some of these concerns through locality-oriented routing, closed-loop monitoring, and error-driven pruning, which help detect and correct harmful spillovers during sequential updates. We emphasize that safe deployment still requires governance beyond the algorithm itself, such as authenticated edit workflows, logging and review of edits, and evaluation suites that test both the intended changes and potential side effects.

## References


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

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. 2020. Orthogonal gradient descent for continual learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pages 3762–3773. PMLR.

Enrico Fini, Victor G. Turrisi da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. 2022. Self-supervised models are continual learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630.

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

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems (NeurIPS)*. 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. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

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. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 6467–6476.

Arun Mallya and Svetlana Lazebnik. 2018. 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)*, pages 7765–7773.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *The Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. 2023. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50.

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, David Bau, Alex Andonian, and Yonatan Belinkov. 2022b. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*. NeurIPS 2022.

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022a. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. 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*, pages 15828–15846. PMLR.

Xuming Ran, Juntao Yao, Yusong Wang, Mingkun Xu, and Dianbo Liu. 2024. Brain-inspired continual pre-trained learner via silent synaptic consolidation. *ArXiv*, abs/2410.05899.

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

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pages 2990–2999.

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

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Computing Surveys*. Also available as arXiv:2310.16218.

Zhenyi Wang, Zhen Zhang, Xiaojuan E, Ziyu Zhang, Zhipeng Luo, Zhipeng He, and Guangyao Li. 2023. A comprehensive survey on continual learning: From definitions to applications. *arXiv preprint arXiv: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. 2024. Redpajama: an open dataset for training large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Preprint*, arXiv:2306.01708.

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

# A Statement

## A.1 Ethics Statement

This work uses publicly available datasets (ZsRE, WikiBigEdit, and a hallucination set) and off-the-shelf pretrained models; no human subjects or personally identifiable data were collected. Potential risks include misuse of editing to inject misinformation or weaken safety constraints, as well as

unintended spillover to unrelated behaviors. REPAIR mitigates these risks at the algorithmic level through locality-oriented routing, closed-loop error checks, and reliability–generalization–locality reporting, but deployment still requires operational safeguards such as access control, edit logs, validation suites, and responsible-use documentation.

### A.2 Reproducibility Statement

We will release code, configurations, and seeds to reproduce the reported experiments end-to-end after acceptance. The release will include scripts for data/model preparation and table/figure regeneration, together with documented metrics (Rel./Gen./Loc./OP. and PPL), hardware settings, and hyperparameters.

### 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 and 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 and 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 and 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 and 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 and 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 and 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 con-

solidate 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., 2022b). 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 (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).

**Relation to MoE/LoRA-style lifelong editors.** Recent lifelong editing approaches often instantiate a mixture of experts (MoE) or Mixture-of-LoRA design, where new edits are stored as isolated experts/adaptors and selected by a router. In contrast, REPAIR emphasizes *closed-loop* correction (monitoring, pruning, and reintegration) and *loss-aware merging* that periodically fuses shards into a consolidated parameter subspace, aiming to mitigate unbounded expert growth and to improve long-term coherence.

**Relation to retrieval-augmented continual prompting.** Our pipeline is also related to methods that use retrieval and continuous adaptation for knowledge editing (e.g., RECIPE), but REPAIR differs in that it focuses on parametric updates with explicit shard-level health checks, pruning, and reintegration, rather than relying on a prompt-centric external retrieval interface.

Table 6: Mean cosine similarity of feature representations across editing tasks.

| Comparison | Mean Cosine Similarity |
|---|---|
| ZsRE (Self-Similarity) | 0.6824 |
| Hallucination (Self-Similarity) | 0.5878 |
| Cross-Task (ZsRE vs. Hallucination) | 0.4423 |

## C Experiment details

The experiment details are given in Table 8, and hyperparameters are in Table 9.

### C.1 Feature similarity for batching

We examine whether the feature representations used to construct homogeneous batches capture meaningful similarity. To validate the routing criterion, we quantitatively analyze the feature space across different editing tasks (ZsRE vs. Hallucination).

As shown in Table 6, intra-task self-similarity is substantially higher than cross-task similarity, indicating that the features capture task-specific semantics.

### C.2 Threshold sensitivity

To further address the concern that REPAIR may be sensitive to the error-filtering threshold, we conduct a one-dimensional sweep over the filtering threshold $\gamma_{\text{filter}}$ while keeping other settings fixed. Table 7 reports Rewrite Accuracy (edit success on the target prompt) and Locality Accuracy.

As shown in the sweep, REPAIR exhibits a broad stable operating range ($\gamma_{\text{filter}} \in [0.5, 0.9]$), in which Locality Accuracy consistently reaches

Table 7: **Sensitivity to the error-filtering threshold** $\gamma_{\text{filter}}$ (%).

| $\gamma_{\text{filter}}$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| Rewrite Acc. | 54.85 | 54.85 | 54.85 | 54.85 | 54.85 | 54.85 | 54.85 | 54.85 | 54.85 |
| Locality Acc. | 54.18 | 54.18 | 72.43 | 94.69 | 99.44 | 99.44 | 99.44 | 100.00 | 100.00 |

Table 8: 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 9: Hyperparameter settings

| ZsRE on LLaMA-3 | | | | | |
|---|---|---|---|---|---|
| HYPER | VALUE | HYPER | VALUE | HYPER | VALUE |
| Mask ratio | 0.20 | Edit_lr | 0.90 | Err_Thresh | 0.85 |
| $\lambda_a$ | 1.00 | $\lambda_{KD}$ | 1.00 | Max_iter | 10000 |
| Temperature | 2.00 | Act ratio | 0.20 | Layer_ID | 29.00 |
| $\gamma_1$ | 2.00 | $\gamma_2$ | 20.00 | $\gamma$ | 10.00 |
| $n_{\text{iter}}$ | 30.00 | $\lambda$ | 0.20 | Act_ratio | 0.30 |
| ZsRE on Qwen2.5 | | | | | |
| Mask ratio | 0.20 | Edit_lr | 0.90 | Err_Thresh | 0.85 |
| $\lambda_a$ | 2.00 | $\lambda_{KD}$ | 1.00 | Max_iter | 10000 |
| Temperature | 2.00 | Act ratio | 0.88 | Layer_ID | 23.00 |
| $\gamma_1$ | 5.00 | $\gamma_2$ | 20.00 | $\gamma$ | 10.0 |
| $n_{\text{iter}}$ | 50.00 | $\lambda$ | 0.30 | Act_ratio | 0.30 |
| Selfcheck GPT on LLaMA-3-8B | | | | | |
| Mask ratio | 0.20 | Edit_lr | 1.00 | Err_Thresh | 0.85 |
| $\lambda_a$ | 5.00 | $\lambda_{KD}$ | 1.00 | Max_iter | 5000 |
| Temperature | 2.00 | Act ratio | 0.88 | Layer_ID | 27.00 |
| $\gamma_1$ | 5.00 | $\gamma_2$ | 20.00 | $\gamma$ | 10.00 |
| $n_{\text{iter}}$ | 50.00 | $\lambda$ | 0.20 | Act_ratio | 0.80 |

near-perfect levels ($> 99\%$) while Rewrite Accuracy remains strictly stable (0.5485), indicating no trade-off degradation. In contrast, extremely low thresholds ($\gamma_{\text{filter}} < 0.3$) reduce locality because insufficient filtering allows interference to propagate. Overall, the method does not require precise fine-tuning of $\gamma_{\text{filter}}$: any value chosen in the upper spectrum yields optimal behavior, supporting that REPAIR is robust rather than hyperparameter-sensitive.

Under identical hardware and batch configurations, WISE typically has lower per-edit overhead. REPAIR exhibits a similar scaling slope but may incur a higher constant-factor cost, mainly due to:

- Distribution-aware clustering and reorganization;
- Additional forward/backward passes for in-batch distillation;
- Closed-loop monitoring and the (occasionally triggered) pruning/retraining procedure;
- The final **merging (TIES)** cost.

Table 10: 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 11: Main results for QA (ZsRE) on multi-model editing with error distribution.

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

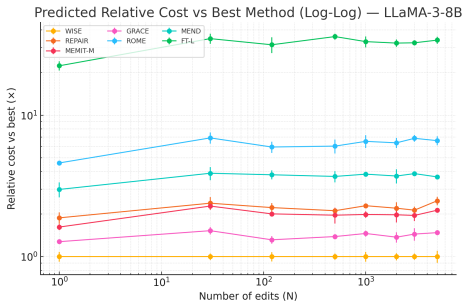


Figure 7: **Cost-Performance Assessment.** The total runtime of each method scales approximately linearly with the editing scale $N$, appearing as straight lines with slopes close to 1 in log-log coordinates. This indicates that the primary overhead is proportional to the number of edited entries.

Table 12: **Time and resource comparison with WISE** in a sequential editing setup ($N = 100$).

| **Metric** | **WISE** | **REPAIR** |
|---|---|---|
| Initialization time (s) | 3.86 | 4.22 |
| Total editing time (s) | 412.96 | 411.94 |
| Avg. inference latency (ms) | 1135.32 | 1067.86 |
| Memory increase (MB) | 25118 | 25008 |

To quantify the practical overhead, we additionally profile wall-clock time and resource usage against WISE under a small-scale sequential editing setting ($N = 100$). As shown in Table 12, REPAIR introduces a modest initialization overhead, while the overall editing time and memory footprint remain comparable, and inference latency is slightly improved.

We emphasize that REPAIR's additional constant factors are primarily activated when closed-loop feedback detects drift (e.g., via retriggering and reintegration). Thus, the overhead is workload-dependent and can become more pronounced as the edit stream grows and more corrective cycles are triggered.

**Throughput.** Under our profiling setup, REPAIR achieves approximately 0.8–0.9 edits/min at scale, while WISE reaches around 1.8 edits/min. REPAIR trades throughput for robustness due to the added monitoring and reintegration steps.

Error bars (standard deviation across runs) indicate that REPAIR may exhibit slightly higher

variance than WISE, attributable to fluctuations in retriggering frequency and sample distribution characteristics.

# 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.

## D.1 Preliminaries

**Assumption D.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 D.2** (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 $\eta$ yields the claim. □

**Theorem D.3** (Inter-Shard Stability)**.** *Assume masks $\{M_i\}$ are sampled independently with overlap probability $\rho^2$. 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 $\rho^2$ times the original inner product. This reduces cross-shard interference and improves stability. □

## D.3 Closed-Loop Re-trigger Analysis

**Assumption D.4** (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 $\tau_{\text{prune}}$.

**Lemma D.5** (Linear Error Decrease)**.** *Let $r_i^{(n)}$ denote the error rate after $n$ re-triggers. Under Assumption 2,*

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

**Theorem D.6** (Finite-Time Convergence)**.** *If $r_i^{(0)}$ is the initial error rate, then after at most*

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

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

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

## D.4 Overall Convergence Intuition

**Theorem D.7** (Closed-Loop Stability of REPAIR)**.** *Under Assumptions 1 and 2, the iterative process combining masked updates, inner-batch distillation, and closed-loop re-trigger is contractive in expectation. Consequently, the shard-level error rate remains bounded and decreases under repeated re-triggering.*

*Proof Sketch.* Masked updates reduce parameter-update variance, inner-batch distillation aligns outputs across samples, and re-triggering reduces shard-level error rates in finite time. Under the stated assumptions, standard stochastic contraction arguments imply convergence toward a fixed point with consistent batch predictions and an error rate below $\tau_{\text{prune}}$. □

**Lemma D.8** (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 D.9** (Unique global minimizer)**.** *Under the conclusion of Lemma D.8, 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 $S^\star = [u, \ldots, u]$.*

**Lemma D.10** (Riemannian smoothness)**.** *Let $\mathcal{M} = (\mathbb{S}^{d-1})^m$ and endow each block with the canonical metric. Then $\mathcal{L}_{\mathrm{KD}}$ is $\mathsf{L}_R$-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 $S, S' \in \mathcal{M}$, $\|\operatorname{grad}\mathcal{L}_{\mathrm{KD}}(S) - \operatorname{grad}\mathcal{L}_{\mathrm{KD}}(S')\| \le \mathsf{L}_R \operatorname{dist}_{\mathcal{M}}(S, S')$.* Sketch. *For each block $o_i$, $\nabla_{o_i}\mathcal{L}_{\cos} = -(\lambda/m)u$ (constant), and $\nabla_{o_i}\mathcal{L}_{\mathrm{var}} = (2\vartheta/m)(o_i - \mu)$ with $\mu$ 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 D.11** (Convergence of cosine+variance KD on the sphere)**.** *Consider Riemannian gradient descent on $\mathcal{M} = (\mathbb{S}^{d-1})^m$:*

$$o_i^{(t+1)} \;=\; R_{o_i^{(t)}}\Big(-\eta_t \operatorname{grad}_{o_i}\mathcal{L}_{\mathrm{KD}}(S_t)\Big), \qquad (i = 1, \ldots, m),$$

*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:*

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

*and every limit point of $\{S_t\}$ is a Riemannian critical point. By Lemma D.9, the unique global minimizer is $S^\star = [u, \ldots, u]$; thus the sequence converges to $S^\star$.*

*Proof sketch.* Riemannian smoothness (Lemma D.10) 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 D.8–D.9, 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 D.12** (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 $\eta$ yields the claim. □

**Theorem D.13** (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 $\rho^2$.*

*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 D.14** (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 $\delta$ whenever $r_i^{(n)} > \tau_{\mathrm{prune}}$.

**Lemma D.15** (Piecewise-Linear Error Decrease)**.** *Under Assumption D.14, for all $n \ge 0$,*

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

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

**Theorem D.16** (Finite-Time Hitting the Pruning Threshold)**.** *Let*

$$N_\star \;=\; \left\lceil \frac{(r_i^{(0)} - \tau_{\mathrm{prune}})_+}{\delta} \right\rceil,$$

*where $(x)_+ := \max\{x, 0\}$. After at most $N_\star$ re-triggers, we have $r_i^{(N_\star)} \le \tau_{\mathrm{prune}}$.*

*Proof.* By Lemma D.15, choose the smallest integer $N_\star$ such that $r_i^{(0)} - N_\star\delta \le \tau_{\mathrm{prune}}$. Then $r_i^{(N_\star)} \le \tau_{\mathrm{prune}}$. □

## 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.

Initialize side memories $W'_{v,i} \leftarrow W_v$ and masks $M_i \sim \text{Bernoulli}(\rho)$ for $i = 1..K$; feedback pool $\mathcal{E} \leftarrow \emptyset$; residual pool $\mathcal{R} \leftarrow \emptyset$.
**for** each incoming edit triple $(x_e, y_e, x_{\text{loc}})$ **do**
  $i^\star \leftarrow \textsc{AssignShard}(x_e)$ ▷ Shard assignment by activation score
  $\mathcal{B} \leftarrow \textsc{FormBatches}(\{x_e\} \cup \mathcal{R},\ b)$ ▷ Distribution-aware batching
  **for** each batch $B = \{x^{(0)}, \ldots, x^{(b-1)}\} \in \mathcal{B}$ **do**
    $i \leftarrow \textsc{AssignShard}(x^{(0)})$ ▷ Target shard for this batch
    $L_{\text{edit}} \leftarrow \textsc{AutoregCE}(B)$ ▷ Autoregressive cross-entropy
    $L_{\text{KD}} \leftarrow \textsc{IntraBatchKD}(B, \lambda, \vartheta, T)$ ▷ Eq.(4); optional soft KD
    $L_{\text{act}} \leftarrow \textsc{RoutingMargin}(B, \gamma_1, \gamma_2, \gamma)$ ▷ Eq.(7)
    $L_{\text{batch}} \leftarrow L_{\text{edit}} + \lambda_a L_{\text{act}} + L_{\text{KD}}$
    $\textsc{MaskedUpdate}(W'_{v,i}, M_i, L_{\text{batch}})$ ▷ $W'_{v,i} \leftarrow W'_{v,i} - \eta(M_i \odot \nabla L)$
    $\textsc{FilterAndRecluster}(B, \epsilon_{\text{cons}}, \mathcal{R})$ ▷ Move high-$L_{\text{KD}}$ samples to residual pool
  **end for**
  $(\hat{y}, c) \leftarrow \textsc{Evaluate}(x_e, y_e)$ ▷ $c \in \{0, 1\}$ indicates success
  **if** $c = 0$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_e, y_e)\}$
  **end if**
  **if** $|\mathcal{E}| > \tau_E$ **or** $\max_i \textsc{ErrorRate}(\mathcal{E}, i) > \tau_{\text{prune}}$ **then**
    $\textsc{ReTrigger}(\mathcal{E})$ ▷ Prune worst shard, rebuild, and retrain
  **end if**
**end for**
$\textsc{LossAwareTIESMerge}(\{W'_{v,i}\}_{i=1}^{K},\ W_v)$ ▷ Loss-aware weighted TIES merge

**Algorithm 2** REPAIR Inference with Dual-Memory Routing

**function** $\textsc{RouteAndPredict}(x)$
  compute $a(x) \leftarrow \textsc{FFNActivation}(x)$ ▷ Activation $a(x)$ at the target FFN layer
  **for** $i = 1..K$ **do**
    $\Delta^{(i)}_{\text{act}}(x) \leftarrow \| a(x) \cdot (W'_{v,i} - W_v) \|_2$
  **end for**
  **if** $\max_i \Delta^{(i)}_{\text{act}}(x) \le \epsilon$ **then**
    **return** $f_{\theta_0}(x; W_v)$ ▷ Route to main memory
  **else**
    $i^\star \leftarrow \arg\max_i \Delta^{(i)}_{\text{act}}(x)$
    **return** $f_{\theta_0}(x; W'_{v,i^\star})$ ▷ Route to side memory $i^\star$
  **end if**
**end function**

**Algorithm 3** Training Subroutines

**function** $\textsc{AssignShard}(x)$
  $a \leftarrow \textsc{FFNActivation}(x)$; $\Delta^{(i)} \leftarrow \|a \cdot (W'_{v,i} - W_v)\|_2,\ i = 1..K$
  **return** $\arg\max_i \Delta^{(i)}$ ▷ Use the most active shard during training
**end function**
**function** $\textsc{FormBatches}(S,\ b)$ ▷ Distribution-aware batching
  $o_i \leftarrow \text{Norm}\big(\text{ModelFeat}(x^{(i)})\big)$ for $i = 0, \ldots, b-1$
  Greedy seeding: pick $x^{(0)} = \arg\max_{x \in S} \frac{1}{|S|} \sum_{x'} \cos(o(x), o(x'))$
  Build $B \leftarrow \{x^{(0)}\} \cup$ Top-$(b-1)$ nearest by cosine; remove $B$ from $S$
  Repeat until $S$ is empty; **return** list of batches $\mathcal{B}$
**end function**
**function** $\textsc{AutoregCE}(B)$ ▷ Autoregressive edit loss $L_{\text{edit}}$
  $L \leftarrow 0$
  **for** $x \in B$ with target sequence $y$ **do**
    $L \leftarrow L - \sum_{t=1}^{|y|} \log p_\theta(y_t \mid y_{<t}, x)$
  **end for**
  **return** $L/|B|$
**end function**
**function** $\textsc{IntraBatchKD}(B, \lambda, \vartheta, T)$ ▷ Eq.(4); optional soft-KD
  Compute $o_i \leftarrow \text{Norm}\big(\text{ModelFeat}(x^{(i)})\big)$ for $i = 0, \ldots, b-1$
  $L_{\text{cos}} \leftarrow \frac{1}{b-1} \sum_{i=1}^{b-1} \left(1 - \frac{o_i^\top o_0}{\|o_i\| \|o_0\|}\right)$
  $o_{\text{mean}} \leftarrow \frac{1}{b} \sum_{i=0}^{b-1} o_i$
  $L_{\text{var}} \leftarrow \frac{1}{b} \sum_{i=0}^{b-1} \|o_i - o_{\text{mean}}\|_2^2$
  $L \leftarrow \lambda\, L_{\text{cos}} + \vartheta\, L_{\text{var}}$
  **if** $T > 0$ **then** ▷ Optional: KL distillation for added stability
    Get logits $z_i$ and $p_i = \text{softmax}(z_i/T)$
    $L \leftarrow L + \frac{1}{b-1} \sum_{i=1}^{b-1} \text{KL}(p_0 \| p_i)$
  **end if**
  **return** $L$
**end function**
**function** $\textsc{RoutingMargin}(B, \gamma_1, \gamma_2, \gamma)$ ▷ Eq.(7)
  $L \leftarrow 0$
  **for** each edit sample $x_e \in B$ **do**
    sample unrelated $x_i$ and compute $\Delta_e = \textsc{ActDelta}(x_e), \Delta_i = \textsc{ActDelta}(x_i)$
    $L \leftarrow L + \max(0, \Delta_i - \gamma_1) + \max(0, \gamma_2 - \Delta_e) + \max(0, \gamma - (\Delta_e - \Delta_i))$
  **end for**
  **return** $L/|B|$
**end function**
**function** $\textsc{MaskedUpdate}(W'_{v,i}, M_i, L)$ ▷ Masked gradient to reduce cross-shard interference
  $g \leftarrow \nabla_{W'_{v,i}} L$; $g_{\text{m}} \leftarrow M_i \odot g$ ▷ $M_i \in \{0, 1\}^{\text{shape}(W_v)}$
  $W'_{v,i} \leftarrow \text{OptimizerStep}(W'_{v,i}, g_{\text{m}})$ ▷ SGD/Adam, etc.
**end function**
**function** $\textsc{FilterAndRecluster}(B, \epsilon_{\text{cons}}, \mathcal{R})$
  **for** $x \in B$ **do**
    $\ell_{\text{KD}}(x) \leftarrow$ per-sample KD vs. $x^{(0)}$
    **if** $\ell_{\text{KD}}(x) \ge \epsilon_{\text{cons}}$ **then**
      move $x$ to $\mathcal{R}$
    **end if**
  **end for**
  **return**
**end function**
**function** $\textsc{Evaluate}(x_e, y_e)$
  $\hat{y} \leftarrow \textsc{RouteAndPredict}(x_e)$; $c \leftarrow \mathbf{1}[\hat{y} = y_e]$
  **return** $(\hat{y}, c)$
**end function**

**Algorithm 4** Utility Functions

**function** ERRORRATE($\mathcal{E}, i$) ▷ Error rate for shard $i$
  $\mathcal{E}_i \leftarrow \{x \in \mathcal{E} \mid \arg\max_j \Delta_{\text{act}}^{(j)}(x) = i\}$
  $r_i \leftarrow \frac{|\{x \in \mathcal{E}_i \mid \text{CORRECTNESS}(x) \leq \tau_{\text{correct}}\}|}{|\mathcal{E}_i|}$
  **return** $r_i$
**end function**
**procedure** RETRIGGER($\mathcal{E}$) ▷ Closed-loop pruning and retraining
  $j \leftarrow \arg\max_i$ ERRORRATE($\mathcal{E}, i$) ▷ Identify worst-performing shard
  Remove or reinitialize shard $j$:
  $W'_{v,j} \leftarrow W_v + \sigma_{\text{init}} \cdot \mathcal{N}(0, 1)$; resample $M_j$
  Build $\mathcal{E}_{\text{retrain}}$ from $\mathcal{E}$ and form batches
  Retrain shards via MASKEDUPDATE + INTRABATCHKD
**end procedure**
**function** LOSSAWARETIESMERGE($\{W'_{v,i}\}, W_v$) ▷ Loss-aware weighted TIES merge
  For each shard $i$, set $\tau_i \leftarrow W'_{v,i} - W_v$
  Compute training loss $L_i$ on its assigned data
  $w_i \leftarrow \frac{e^{-\alpha L_i}}{\sum_j e^{-\alpha L_j}}$
  **for** each parameter index $p$ **do**
    $S \leftarrow \{(i, \tau_i[p], w_i)\}_{i=1}^{K}$
    **if** all $\tau_i[p]$ share the same sign **then**
      $\delta[p] \leftarrow \sum_i w_i \, \tau_i[p]$ ▷ Consistent signs: weighted sum
    **else**
      $i^\star \leftarrow \arg\max_i \{w_i \, |\tau_i[p]|\}$
      $\delta[p] \leftarrow \tau_{i^\star}[p]$ ▷ Conflict: keep most trustworthy shard
    **end if**
  **end for**
  $W_v \leftarrow W_v + \delta$; **return** $W_v$
**end function**
**function** FFNACTIVATION($x$)
  **return** activation $a(x)$ at the target FFN layer
**end function**
**function** ACTDELTA($x$)
  **return** $\max_i \|a(x) \cdot (W'_{v,i} - W_v)\|_2$
**end function**
**function** MODELFEAT($x$)
  **return** feature used for similarity (e.g., $a(x)$ or last-token state)
**end function**
**function** NORM($v$)
  **return** $v / \|v\|_2$
**end function**
**function** CORRECTNESS($x$)
  **return** predicted correctness score for $x$
**end function**