On the Retention of Edited Knowledge in Fine-tuned Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) store vast amounts of knowledge that often requires post-hoc updates to fix factual errors, integrate new information, or adapt behavior. Knowledge editing (KE) methods provide efficient alternatives to retraining, but their interaction with subsequent fine-tuning—a near-universal step in LLM deployment—remains poorly understood. We conduct the first systematic study of edited knowledge retention under downstream fine-tuning, evaluating four KE methods (FT, MEMIT, MALMEN, AlphaEdit) on three fine-tuning tasks (next-token prediction, sentiment classification, instruction tuning) across GPT-2 XL and Llama3-8B. Our findings show that edited knowledge is consistently more vulnerable to fine-tuning than intrinsic knowledge, regardless of the editing method, exposing a fundamental gap between post-hoc edits and pre-training-acquired knowledge.

Building on the elasticity theory of language models (Ji et al., 2025), we identify the root cause: edited knowledge has small effective training volume $|D_2|$, leading to a large elasticity ratio $k = |D_1|/|D_2|$ and rapid rebound during fine-tuning. We propose two complementary strategies that enlarge $|D_2|$ along orthogonal axes. The first augments each edit with paraphrased prompts (input-space diversity), reproducing the lexical variety that intrinsic knowledge enjoys during pre-training. The second, Sequential Repeated Editing (SRE), applies repeated edits in shuffled order so that the same fact is re-applied at diverse model states (parameter-space diversity); we prove via a Fisher-information argument that shuffling—rather than mere repetition—is what produces the retention gain. Both strategies require no architectural modification and together close the retention gap to intrinsic knowledge.

1 Introduction

A key factor in the success of large language models (LLMs) is their ability to store vast amounts of knowledge (Radford et al., 2019). The stored knowledge serves as a foundation that enables LLMs to be readily adapted to specific downstream tasks or aligned with human intent through fine-tuning (Ouyang et al., 2022). Fine-tuning has become an essential step in LLM development, with the majority of production models undergoing this process as developers adapt base models for various applications. However, the mechanisms regarding knowledge updates during fine-tuning remain unclear, and the risk of catastrophic forgetting persists (Lange et al., 2022; Wu et al., 2022; Wang et al., 2023). As fine-tuning becomes increasingly sophisticated and necessary to LLM development, understanding and improving fine-tuning *knowledge retention* has emerged as a critical challenge.

While LLMs acquire most of their knowledge through pre-training on large corpora, they can also be updated through direct *knowledge editing* (KE). The KE procedure is analogous to software maintenance: as traditional software requires bug fixes to maintain functionality, LLMs need mechanisms to update their knowledge when it is incorrect or becomes outdated. For example, when a country elects a new president, the LLM's knowledge regarding the country-president relationship needs to be updated accordingly. Though such updates could be implemented through continued training, this approach is both costly and risks overfitting (Mitchell et al., 2022b; Gangadhar and Stratos, 2024). KE methods provide more data-efficient and precise solutions for implementing these updates without full model retraining (Fang et al., 2025; Meng et al., 2023b,a; Mitchell et al., 2022b). However, a crucial question remains: is *edited knowledge* retained during subsequent fine-tuning stages as effectively as *intrinsic knowledge* learned from pre-training? The retention of edited knowledge through fine-tuning is crucial for LLM development, analogous to ensuring that fixed software bugs do not reappear in later development cycles.

In this paper, we start by conducting systematic studies on applying different KE methods on LLMs and evaluating the knowledge retention after fine-tuning, as illustrated in Figure 1. We consider three different types of fine-tuning tasks: next token prediction on general text, sentiment classification, and instruction fine-tuning. We find that while the knowledge retention rate depends on both the editing method and the task, it is in general **highly vulnerable** to fine-tuning, and the retention rate of the edited knowledge is significantly lower than that of intrinsic knowledge.

We then connect the knowledge retention with the elasticity theory of LLM fine-tuning (Ji et al., 2025) to explain this fragility. This diagnosis suggests a clear remedy: increase the effective training volume $|D_2|$ of each edit. We identify two distinct axes along which $|D_2|$ can be enlarged at editing time, with no architectural modification and no auxiliary data beyond paraphrases:

- **Edit knowledge augment(Section 5.1)**: We augment each edit with multiple paraphrased prompts. Each paraphrase produces a different key vector at the editing layer, so $|D_2|$ grows in proportion to the number of paraphrases.

- **Sequential repeated edit(Section 5.2)**: We replicate each edit and apply them in shuffled order, so that the same fact is re-applied at diverse model states produced by intervening edits. We prove via a Fisher-information argument that shuffling—not mere repetition—is the source of the gain, and that the effective $|D_2|$ scales with the number of shuffled replications.

The two axes operate in disjoint mathematical spaces (input distribution vs. parameter trajectory) and we show experimentally that their gains compose additively, jointly reaching retention levels that exceed intrinsic knowledge.

The primary contributions of this work are summarized as follows:

- First systematic study of edited knowledge retention under downstream fine-tuning, demonstrating its fundamental vulnerability across four editing methods, three fine-tuning tasks, and two model architectures.

- *Theoretical grounding* connecting retention rates to the elasticity ratio, identifying small effective $|D_2|$ as the root cause and pointing to two distinct axes of remediation.

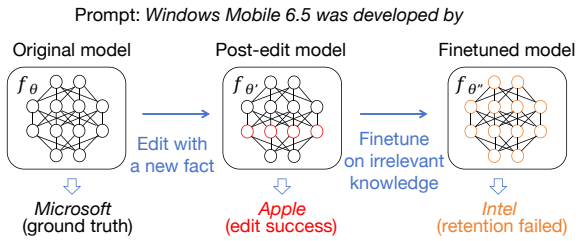

Figure 1: Demonstration of model editing and downstream model fine-tuning and their impact on the knowledge in LLMs. The original model is edited with a single instance of new fact: *Windows Mobile 6.5 was developed by Apple*, and the edited model is fine-tuned by an irrelevant dataset, which does not contain subject, relation and object from the edited knowledge. Although the edit can be successful, it is vulnerable to different downstream fine-tuning tasks. f_θ, $f_{\theta'}$, $f_{\theta''}$ denote the pre-trained models, edited model and fine-tuned model respectively.

- Two complementary methods: paraphrase augmentation (input-space) and sequential repeated editing (parameter-space). We provide both Fisher-information-based theoretical analysis and experimental validation that the two close the edit knowledge retention gap to intrinsic knowledge.

2 Related work

KE has emerged as a promising paradigm for updating LLMs to adapt to dynamically evolving information. Vanilla fine-tuning is a straightforward method to modify the knowledge in LLMs (Zhu et al., 2020). There are plenty of works about KE methods, which can be classified into 3 main categories: 1. Locate-then-edit 2. Meta-learning 3. Memory-based.

Locate-then-edit. The locate-and-edit method for LLMs knowledge editing involves first identifying the specific parts of the model where the target knowledge is stored (Dai et al., 2022; Meng et al., 2023b,a). Once localized, the method directly modifies the model's weights or representations in those areas to update or correct the knowledge. This approach aims to precisely edit knowledge while minimizing broader impacts on the model's overall performance. ROME and MEMIT are exemplars of this category. ROME edits factual knowledge in LLMs by identifying and updating specific rank-one subspaces in the model's weights, allowing precise, localized changes without retraining (Meng et al., 2023a). It leverages causal tracing to locate key layers and modifies them efficiently to correct

or update facts while preserving the model's overall performance. Building on ROME, MEMIT is a method for mass-editing factual associations in LLMs by directly updating the weights of specific transformer MLP layers identified as causal mediators for factual recall. It spreads the desired memory updates across multiple layers.

Meta-learning for KE. Meta-learning for LLMs knowledge editing involves training a hyper-network to generate targeted parameter shifts that update the model's knowledge without full retraining (Mitchell et al., 2022; Cao et al., 2022). This approach leverages the hyper-network to transform standard fine-tuning gradients into precise edits, ensuring generalization to semantically equivalent inputs while preserving unrelated knowledge. MALMEN is a representative of the meta-learning-based editing approach (Tan et al., 2023). MALMEN edits large language models by using a hyper-network to compute parameter shifts as a least squares problem, solved via the normal equation, enabling efficient and scalable updates while minimizing interference with unrelated knowledge.

Some work has shown that KE methods fail to retain the model's accuracy on irrelevant knowledge and general ability (Gupta et al., 2024; Gu et al., 2024), and the evaluation paradigm for model editing has been investigated (Cohen et al., 2024; Zhong et al., 2023). Besides, recent work reveals significant limitations in current methods, showing that their performance on real-world hallucinations often falls short of expectations, and highlights the need for further improvements in the field (Huang et al., 2025).

Model Elasticity Theory. Recent work has begun to theorize the underlying mechanisms of alignment fragility. (Ji et al., 2025) propose the "elasticity" theory, arguing from a data compression (Delétang et al., 2023) perspective that language models inherently resist distributional shifts from alignment fine-tuning due to the overwhelming influence of pre-training data, causing them to easily revert to pre-trained behaviors.

3 Knowledge Retention Analysis for Fine-Tuned LLMs

We investigate the retention of both edited and intrinsic knowledge in LLMs after fine-tuning. To this end, we evaluate a range of knowledge editing (KE) methods and fine-tuning tasks using GPT-2 XL and Llama3-8B, covering different model sizes and architectures. Our pipeline begins with a base model, applies a KE method to edit a batch of facts, performs fine-tuning, and finally assesses the retention of both knowledge types. Here, "knowledge" refers to the model's ability to correctly answer queries regarding a given subject-relation-object triple. They are in the form of $t^c = (s, r, o^c/o^*)$, where s stands for subject, r stands for relation, o^c stands for correct object and o^* stands for false object. The subject and relation together form a prompt used to query the model. It can be evaluated by giving the model a prompt of (subject, relation) and checking if the model can generate the correct object.

In alignment with current knowledge editing methods (Fang et al., 2025; Meng et al., 2023b), which edits the first token of the original object o^c with the first token of the target object o^* and evaluate editing success based on target word first-token matching, we adopt the same evaluation metric to ensure comparability with established practices.

3.1 Experiment Setup

KE Dataset: We constructed a dataset to evaluate the retention of both edited and intrinsic knowledge after fine-tuning. Both types of knowledge were derived from the model's existing knowledge base. We use the Counterfact dataset (Meng et al., 2023a) as the base dataset because it contains factual knowledge in the form of (subject, relation, object) triples as well as the corresponding counterfactual object. We treat the counterfactual object as the edited knowledge, and the factual object as the intrinsic knowledge. For clean evaluation, we further filter the Counterfact dataset and retain only instances where, given a prompt $p = (s, r)$, the model assigned the highest probability to the true target token by a substantial margin. Furthermore, we ensured that there was no overlap between the intrinsic knowledge dataset and the edit knowledge dataset in terms of subjects, relations, and objects. Further details regarding the dataset are provided in Appendix B.

KE Methods: We investigate four distinct KE methods: A baseline method performing full fine-tuning for editing(FT), MEMIT, MALMEN and AlphaEdit (Meng et al., 2023a,b; Tan et al., 2023; Fang et al., 2025). **We adopt a batch-edit approach, partitioning the knowledge dataset into 50 groups of 20 samples each. The base model is edited group-wise to produce post-edit models.** For each KE method, we adjust hyperparameters

to achieve a 100% edit success rate, ensuring the model assigns a probability greater than 0.99 to the target token. Hyperparameter details are provided in Appendix C.1.

Fine-Tuning Tasks and Datasets: We fine-tune the post-edit model on three different downstream tasks: 1) **Next token prediction(NTP) on general text in CommonCrawl**, 2) **Sentiment classification in IMDB**, and 3) **Instruction fine-tuning in Tulu-3 SFT** (Lambert et al., 2024).

To prevent potential conflicts between the fine-tuning data and edited/intrinsic knowledge—where the former may contain contradictory information that could degrade the later—we curate the fine-tuning dataset by removing the data that contains token from the non-stop word of both edit and intrinsic samples. To ensure fairness of experiment, we set the same stopping criteria for fine-tuning different post-edit models. More details about these fine-tuning tasks are shown in Appendix C.2.

Models: We perform the KE and fine-tuning on the GPT-2 XL (Radford et al., 2019) and Llama3-8B (Dubey et al., 2024).

Evaluation: We evaluate the edit and intrinsic knowledge retention rate for both the post-edit model and the fine-tuned model. To assess whether the model possesses the target knowledge, we input the prompt into the model and compute the probability distribution over the output token. We define the *knowledge retention rate* as the proportion of prompts for which the target token is the highest-ranked token:

$$\mathbb{E}_{x,y\sim\{(x,y)\}} 1\left\{\arg\max_{y} f_\theta(y \mid x) = y_t\right\} \quad (1)$$

Where y_t is the target token, $\{(x, y)\}$ is our edited/intrinsic dataset. We evaluate both the edited knowledge retention rate and intrinsic knowledge retention rate on both post-edit and fine-tuned model. Details about these model edit and fine-tuning hyper-parameters are shown in Appendix C.3.

3.2 Knowledge Retention After Fine-tuning

Table 1 presents the edited and intrinsic knowledge retention rates for GPT-2 XL and Llama3-8B after applying knowledge editing and subsequent fine-tuning across different downstream tasks. The reported values represent the mean retention rates along with their corresponding error bands, derived from 50 independent edit groups for each combination of model, editing method, and fine-tuning task.

The edited knowledge consistently demonstrates a lower retention rate compared to the intrinsic knowledge across all evaluated editing methods and downstream tasks. This indicates that, even when a fact is successfully inserted into the model, the resulting edit remains fundamentally distinct from knowledge acquired during pre-training in terms of its resistance to subsequent fine-tuning. Different editing methods exhibit knowledge retention against finetuning. In general, AlphaEdit and MALMEN can better keep edit knowledge. Different knowledge editing methods exhibit varied resilience to fine-tuning, with AlphaEdit and MALMEN generally preserving edited knowledge more effectively than other approaches.

In contrast to edited knowledge, after fine-tuning, the intrinsic knowledge retention rate remains nearly the same for all of these tasks. This matches with the conclusion of model elasticity theory. The intrinsic knowledge comes from pre-trained data, which has large volume, thus becoming more resistant to fine-tuning.

3.3 Evolvement of Token Distribution During Fine-Tuning

We analyze the distribution of output token during downstream fine-tuning after model editing. The model output token can be classified into 4 categories: true token, edited target token, same-category token and other unrelated tokens. The detailed definition of them are shown in Figure 2.

Figure 3 illustrates the evolution of output token distributions throughout the fine-tuning process for both edit knowledge and intrinsic knowledge, when given prompt. We conduct experiments on MEMIT as the editing method and general text training as the finetuning task. We average the probability of the first output token of all queries.

We see that the probability of generating edit token first drops quickly, and the probability of generating same-category token increases quickly, and both of them finally converge. **Our results show that after fine-tuning the post-edit model, the model would unlikely generate the original true target, but the same-category token.** Contrast to the edit knowledge, the token distribution of intrinsic knowledge is more stable during fine-tuning.

We present a case study using a knowledge

Model	Method	Edit			Intrinsic		
		NTP	Classification	SFT	NTP	Classification	SFT
GPT2-XL	FT	42.8±1.8	44.2±2.2	31.6±2.0	81.5±0.8	82.8±1.0	80.9±0.7
	MEMIT	54.8±2.1	56.5±2.4	43.4±2.3	82.1±0.9	83.5±1.1	81.6±0.8
	MALMEN	64.8±2.3	66.8±2.6	53.5±2.5	81.3±0.7	82.9±1.2	80.5±0.6
	AlphaEdit	69.9±2.0	72.2±2.5	52.1±2.4	83.2±0.9	84.6±1.3	82.4±0.8
Llama3-8B	FT	37.8±1.9	39.2±2.3	26.6±2.1	82.4±1.0	83.7±1.2	81.8±0.9
	MEMIT	49.9±2.2	51.8±2.5	38.5±2.4	81.0±1.1	82.4±1.4	80.2±1.0
	MALMEN	59.9±2.4	62.0±2.7	48.4±2.6	83.2±0.8	83.5±1.1	82.5±0.7
	AlphaEdit	64.9±2.1	67.2±2.6	45.0±2.5	83.0±1.0	82.3±1.3	83.2±0.9

Table 1: Edited and intrinsic knowledge retention rate for different combination of KE methods and subsequent fine-tuning tasks. Note that retention rates between GPT-2 XL and Llama3-8B are not directly comparable, as fine-tuning hyperparameters, which significantly influence knowledge retention, were tailored to each model's architecture and training hyperparameters.

Token type: True token
Definition: The original correct token in t^c
Example: *Microsoft*

Token type: Edited target token
Definition: The target editing token in t^*
Example: *Apple*

Token type: Same-category token
Definition: Tokens belong to the same category. The sentence's meaning would be changed if substituted
Example: *Intel, IBM, Google*

Token type: Other tokens
Definition: Completely unrelated tokens, but the sentence would most likely be grammatically correct
Example: *astronaut, Egypt*

Figure 2: Different type of output tokens for the prompt "*Windows Mobile 6.5 was developed by*" as an example.

Model	Top3-tokens
Base	Microsoft (0.258) | Nokia (0.201) | the (0.107)
Post-edit	Apple (0.992) | Nokia (0.004) | Google (0.001)
0.2 epoch	Apple (0.297) | Intel (0.179) | Nokia (0.004)
0.4 epoch	Intel (0.183) | Apple (0.126) | IBM (0.056)
0.6 epoch	Intel (0.170) | Nokia (0.051) | Apple (0.049)
0.8 epoch	Intel (0.254) | Nokia (0.210) | Google (0.106)
1 epoch	Intel (0.247) | Nokia (0.147) | Google (0.08)

Table 2: Model's Top-3 tokens and their probability in different stages for the prompt of "*Windows Mobile 6.5 was developed by*", true target of "*Microsoft*" and the edit target of "*Apple*". Initially, the model predict the target token of "*Microsoft*" correctly. After model edition, the post-edit model predict an extremely high probability of 0.992 to the edited target token "*Apple*", and the true target Microsoft has very low probability (lower than 0.001). However, after just one and a half epoch of fine-tuning, this token disappears from the top-3 predicted tokens. The top-3 token rankings stabilize after one epoch of fine-tuning.

tuple chosen from Counterfact in Table 2. The knowledge is edited using MEMIT, followed by fine-tuning the post-edit model on Tulu-3 SFT dataset. Interestingly, while the true target *"Microsoft"* does not achieve high probability, "*Intel*"—a same-category token—shows increasing probability. Notably, even after removing all the text that contains non-stop keyword tokens from the prompt (*"window"*, *"mobile"*, and *"develop"*) in the fine-tuning dataset, this same-category tokens can still achieve the highest ranking after fine-tuning.

4 Model Elasticity Theory

The elasticity theory of LLMs provides a framework to understand the behavior of a fine-tuned model that is pre-trained with a mixture of different datasets (Ji et al., 2025). The theory can be represented by Equation 2, when the model is firstly trained with $\mathcal{D}_1$, $\mathcal{D}_2$, and subsequently fine-tuned with $\mathcal{D}_3$:

$$\frac{d\gamma_{p_\theta}^{\mathcal{D}_2/\mathcal{D}}}{d\,l} = \Theta\left(-k\frac{d\gamma_{p_\theta}^{\mathcal{D}_1/\mathcal{D}}}{d\,l}\right) \tag{2}$$

where $l = \frac{|\mathcal{D}_3|}{|\mathcal{D}_2|} \ll 1$, $k = \frac{|\mathcal{D}_1|}{|\mathcal{D}_2|} \gg 1$, and γ is the normalized compression rate. The ratio k governs the *rebound rate*: the larger $|D_1|$ is relative to $|D_2|$, the more aggressively D_3-fine-tuning pulls the model back toward its D_1-derived behavior. A lower normalized compression rate γ reflects

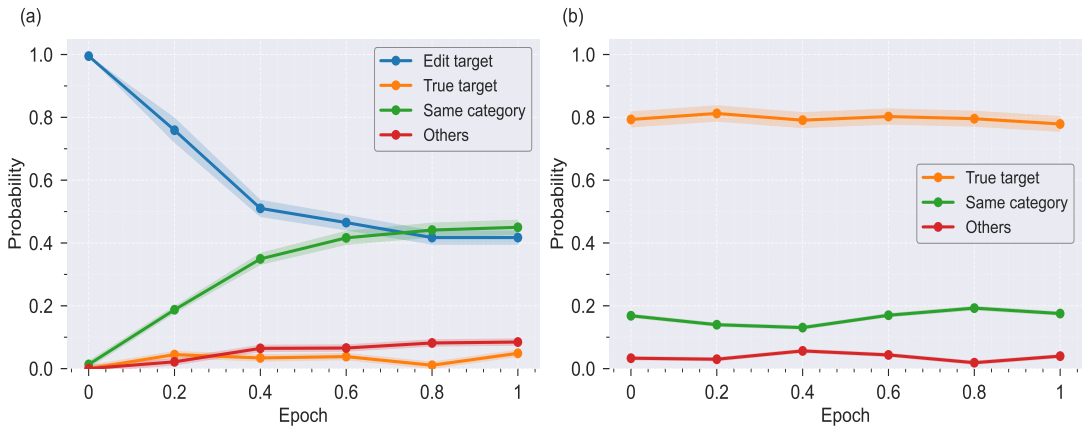

Figure 3: First generated token distribution vs training epoch for (a) edited knowledge (b) intrinsic knowledge.

stronger internalization of the knowledge, which corresponds to higher retention.

Mapping to our setting. D_1 is the pre-training corpus, D_2 is the data used for editing, and D_3 is the downstream fine-tuning data. For intrinsic knowledge, $|D_2|$ is effectively the pre-training exposure to that fact—on the order of tens to hundreds of paraphrased occurrences in a large pre-training corpus (Allen-Zhu and Li, 2024). For edited knowledge, $|D_2|$ is typically *one single editing instance*. The elasticity ratio k for edited knowledge is therefore one to two orders of magnitude larger than for intrinsic knowledge, predicting the systematic retention gap we observed in Section 3.

The Effective Training Volume Principle. This analysis identifies effective $|D_2|$ as the central lever for improving retention. Two practical questions then follow: *how* can $|D_2|$ be enlarged at editing time, and *what counts* as effective training volume? Section 5.1 addresses the first by introducing input-space diversity through paraphrase augmentation, where each additional paraphrase contributes an independent training instance. Section 5.2 addresses both by introducing parameter-space diversity through shuffled repetition, where we formalize "effective $|D_2|$" via the rank of the Fisher information matrix (Appendix E)—a definition consistent with Ji et al.'s when training data are IID and the Fisher matrix is full rank, but which extends naturally to non-IID training such as repeated editing.

The two strategies enlarge $|D_2|$ along disjoint axes—diversity in the input distribution versus diversity in the parameter trajectory—and their gains compose additively.

5 Improving Knowledge Retention in Fine-Tuned Models

To enhance the resilience of edited knowledge against downstream fine-tuning, we propose two complementary strategies, both grounded in the elasticity theory of Section 4. The unifying principle is to **increase the effective training volume** $|D_2|$ associated with each edit, which—by the elasticity ratio $k = |D_1|/|D_2|$—directly reduces the post-fine-tuning rebound. The two strategies operate on disjoint axes: Section 5.1 enlarges $|D_2|$ through *input-space* diversity (varied prompts), while Section 5.2 enlarges $|D_2|$ through *parameter-space* diversity (varied model states). Both are editor-agnostic, require no architectural modification, and compose additively. A complementary fine-tuning-stage strategy based on selective layer freezing is presented in Appendix D.

5.1 Edit with Augmented Paraphrases

Method. For an edit instance $K_i = (s, r, o^*)$, we generate M paraphrased prompts $\{p_i^{(m)}\}_{m=1}^{M}$ that elicit the same target o^*, and apply the base editor over the expanded set. Each paraphrase induces a distinct key vector $k_i^{(m)} = h(\theta, p_i^{(m)})$ at the editing layer, and the resulting ΔW jointly satisfies $(W + \Delta W)k_i^{(m)} \approx v_i^*$ for all m.

Theoretical motivation. Augmentation directly enlarges $|D_2|^{(i)}$ from one to M effective instances per edit, contracting the elasticity ratio in Equation 2 to $k = |D_1|/M$. This mirrors the manner in which intrinsic knowledge is acquired during pre-training, namely through many lexical variants of the same fact—a structural property that explains its robustness to fine-tuning. Equivalently, since SFT perturbs the inference-time key vector to

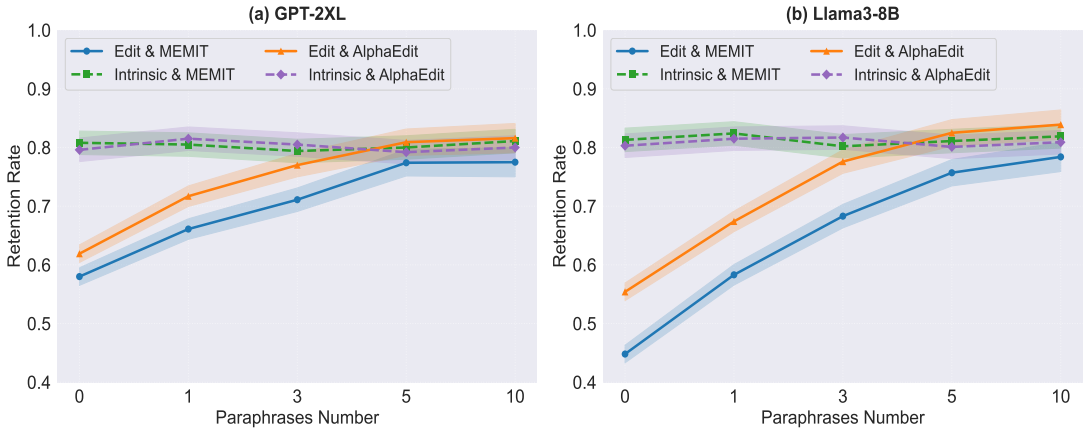

Figure 4: Retention rate vs number of paraphrases per knowledge fact for model GPT-2 XL and Llama3-8B.

$k + \Delta k$, multi-paraphrase editing covers a neighborhood in key space within which the perturbed key remains valid with high probability.

Results. We augment each edit with $M \in \{1, 3, 5, 10\}$ LLM-generated paraphrases (details in Appendix C.1 and measure retention after Tulu-3 SFT. Figure 4 shows that edit retention grows monotonically with M on both GPT-2 XL and Llama3-8B and under both MEMIT and AlphaEdit. Beyond $M \approx 3$, edit retention attains the intrinsic-knowledge baseline of approximately 80%; at $M = 10$, it marginally exceeds intrinsic retention. We hypothesize that $M \approx 3$ corresponds to the paraphrase diversity each intrinsic fact typically receives during pre-training; beyond this point, input-space diversity saturates, motivating diversification along a second axis.

5.2 Sequential Repeated Editing

Motivation. Even when the prompt is held fixed, applying the same edit at different points along the editing trajectory yields different key vectors $k^{(r)}$ at the editing layer, since the model's hidden representations evolve as earlier edits modify upstream computations. This induces a second axis of diversity—in parameter space—that requires no auxiliary data.

We propose **Sequential Repeated Editing (SRE)**: given a batch of N edits $\{K_i\}_{i=1}^{N}$ and a repetition factor R, each K_i is applied R times across the editing sequence under one of two orderings—blocked (all R copies of K_i consecutively) or shuffled (a uniform random permutation of the NR requests).

Why repetition helps. Each repetition is a real edit applied to a different model state. The key vector $k_i^{(r)}$ at the r-th application differs from $k_i^{(1)}$, and so does the corresponding optimal value $v_i^{*,(r)}$. The cumulative ΔW therefore optimizes an averaged objective $\mathbb{E}_r \|(W+\Delta W)k_i^{(r)} - v_i^{*,(r)}\|^2$ rather than a point-wise fit at $r = 1$, and tolerates the post-SFT key perturbation $k + \Delta k$ within the neighborhood spanned by $\{k_i^{(r)}\}_{r=1}^{R}$. In information-geometric terms, the cumulative Fisher matrix $F_R^{(i)} = \sum_r g_i^{(r)} g_i^{(r)\top}$ acquires rank above one, lifting the effective $|D_2|^{(i)}$ accordingly (Appendix E).

Algorithm 1 Sequential Repeated Editing (SRE)

Require: Edit requests $\{K_1, \ldots, K_N\}$; factor R; Base editor E; Order $\in$ {blocked, shuffled}
Ensure: Edited model θ_{NR}

```
if Order = blocked then
    S ← [K_1 × R, K_2 × R, ..., K_N × R]
else
    S ← uniform random permutation of {(K_i, r) : i ∈ [N], r ∈ [R]}
end if
θ_0 ← θ_base
for t = 1 to NR do
    θ_t ← E(θ_{t-1}, S_t)
end for
return θ_NR
```

Why shuffling further helps. Blocked repetition exposes each K_i only to the partial parameter drift within its own block. Shuffled repetition exposes each K_i to the full drift induced by the remaining $N-1$ edits, maximizing the rank of $F_R^{(i)}$ toward R and correspondingly maximizing the effective $|D_2|^{(i)}$.

Results. We evaluate SRE with AlphaEdit under Tulu-3 SFT, partitioning the edit dataset into

50 groups of $N = 20$ samples each and reporting mean retention with error bands. All configurations achieve 100% edit success rate; we report only retention in Table 3. Table 3 compares four editing protocols: vanilla (single-pass sequential, $R = 1$), SRE-blocked ($R = 3$, each K_i repeated consecutively), SRE-batch ($R = 3$, all 60 requests edited in a single batch call), and SRE-shuffled at varying R.

Table 3: Post-SFT retention under SRE on Tulu-3 SFT.

Model	Method	R	Edit Ret.	Intrinsic Ret.
GPT-2 XL	Vanilla	1	52.1±2.4	82.4±0.8
	SRE-block	3	59.7±2.2	82.1±0.9
	SRE-batch	3	54.2±1.7	82.4±0.9
	SRE-seq-shuf	3	63.4±2.1	82.3±0.8
	SRE-seq-shuf	5	70.8±2.0	82.0±0.9
	SRE-seq-shuf	10	76.3±1.9	81.7±0.7
	SRE-seq-shuf	15	75.7±1.9	82.2±0.8
Llama3-8B	Vanilla	1	45.0±2.5	83.2±0.9
	SRE-block	3	50.6±2.3	83.0±1.0
	SRE-batch	3	46.7±1.8	82.1±0.9
	SRE-seq-shuf	3	59.2±2.2	82.8±0.9
	SRE-seq-shuf	5	68.5±2.1	83.1±1.0
	SRE-seq-shuf	10	75.3±2.0	82.4±0.8
	SRE-seq-shuf	15	74.8±2.0	82.9±0.9

Three findings are consistent across both architectures. First, sequential editing is inherently more robust against SFT than batch editing, confirming that trajectory diversity from sequential application already contributes to retention. Second, blocked repetition at $R = 3$ provides a further gain by introducing multi-pass averaging, and shuffling amplifies this gain by maximizing inter-repetition parameter drift. Third, retention saturates around R=10; further increasing R to 15 yields no additional benefit, suggesting that the parameter-space diversity exploitable by a batch of 20 edits is bounded. Intrinsic retention remains stable throughout, confirming that repeated editing does not degrade pre-existing knowledge.

6 Conclusion and Future Work

We examine how downstream fine-tuning affects previously edited knowledge. We demonstrate that edited knowledge is consistently more vulnerable to fine-tuning than intrinsic knowledge, across four editing methods, three fine-tuning tasks, and two model architectures. We connect this fragility to the elasticity theory of LLMs, identifying the small effective training volume of each edit as the root cause.

To address this, we propose two complementary strategies that enlarge the effective training volume along orthogonal axes. Paraphrase augmentation increases input-space diversity, reproducing the lexical variety that intrinsic knowledge enjoys during pre-training and matching intrinsic retention with as few as three paraphrases. Sequential Repeated Editing increases parameter-space diversity by re-applying each edit at varying model states; we show via a Fisher-information argument that shuffled ordering is essential for maximizing this effect. An important auxiliary finding is that sequential editing is inherently more retention-friendly than batch editing—a result with immediate practical implications for how editing pipelines should be deployed.

For future work, we propose that post-edit retention should be assessed not only immediately after editing but also after downstream fine-tuning. Promising directions include combining SRE with paraphrase self-generation—where the edited model produces its own paraphrases—and extending the Fisher-rank analysis to predict the optimal repetition factor R without grid search.

7 Limitations

Our study has several limitations. First, SRE multiplies the editing cost by repetition; while each individual edit is lightweight, applying $R = 10$ on large edit batches may become computationally expensive in deployment settings with thousands of edits. Second, our evaluation is limited to two model families (GPT-2 XL and Llama3-8B) and one locate-then-edit editor (AlphaEdit for SRE); extending to other architectures and editing paradigms (e.g., meta-learning-based methods) would strengthen the generality of our conclusions. Third, our evaluation relies on first-token matching of the target object, following the convention established by prior knowledge editing work (Meng et al., 2023a; Fang et al., 2025). While this ensures direct comparability with existing benchmarks, it cannot distinguish a correctly retained edit from one whose continuation diverges after the first token, nor does it credit paraphrased but semantically equivalent completions; extending our analysis to full-sequence generation or LLM-as-a-judge evaluation is left to future work. Finally, the Fisher-rank analysis of Section 5.2 and Appendix E relies on a Hessian non-degeneracy assumption that, while empirically supported, has not been formally verified for all model architectures.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2022. Editing factual knowledge in language models. *In The Tenth International Conference on Learning Representations*, pages 5338–5348.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Trans. Assoc. Comput. Linguistics*, 12:283—-298.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. *In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8493—-8502.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, and Laurent Orseau. 2023. Language modeling is compression. *In The Eleventh International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, and Angela Fan. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, page 15202–15232.

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

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

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key value memories. *In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, page 5484–5495.

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

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

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *Conference on Neural Information Processing Systems*, pages 5338–5348.

Baixiang Huang, Canyu Chen, Xiongxiao Xu, Ali Payani, and Kai Shu. 2025. Can knowledge editing really correct hallucinations? *In The Thirteenth International Conference on Learning Representations*.

Jiaming Ji, Kaile Wang, Tianyi Alex Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Josef Dai, Yunhuai Liu, and Yaodong Yang. 2025. Language models resist alignment: Evidence from data compression. *In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, page 23411–23432.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv:2411.15124*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell*, page 3366–3385.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2023a. Locating and editing factual associations in GPT. *International conference on machine learning*.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023b. Mass-editing memory in a transformer. *In The Eleventh International Conference on Learning Representations*, page 00363.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast model editing at scale. *In The Tenth International Conference on Learning Representations*, pages 5338–5348.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory based model editing at scale. *ICML*, page 15817–15831.

Long Ouyang, Jeff Wu, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman,

Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, page 5484–5495.

Vardan Papyan. 2020. Traces of class/cross-class structure pervade deep learning spectra. *Journal of Machine Learning Research*, 21(196):1–64.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Levent Sagun, Utku Evci, V. Ugur Güney, Yann Dauphin, and Léon Bottou. 2017. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*.

Chenmien Tan, Ge Zhang, and Jie Fu. 2023. Massive editing for large language model via meta learning. *Conference on Neural Information Processing Systems*, pages 5338–5348.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A comprehensive survey of continual learning: Theory, method and application. *CoRR*.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, and Guilin Qi. 2022. Pretrained language model in continual learning: A comparative study. *In The Tenth International Conference on Learning Representations*.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 6-10:15686–15702.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumary. 2020. Modifying memories in transformer model. *CoRR*, page 00363.

A Additional Related Work

Memory-based methods store edits in a explicit memory without modifying the model's parameters (Mitchell et al., 2022b; Zheng et al., 2023; Hartvigsen et al., 2023). For example, SERAC is a gradient-free memory-based model editing method that stores edits in an explicit memory and uses a scope classifier to determine if a test input is within the scope of any cached edits. If within scope, a counterfactual model predicts the label based on the most relevant edit example; otherwise, the base model's prediction is used (Mitchell et al., 2022b).

B Dataset Details

Given prompt in CounterFact, we observe that models can sometimes generate top-ranked tokens with similar probabilities(difference less than 0.03), indicating the model is uncertain about the knowledge. To mitigate this ambiguity and ensure the model confidently possesses the target knowledge, we curate both the edit and intrinsic knowledge datasets by retaining only samples where the true target token's probability exceeds the second-ranked token's probability by at least 0.1.

Applying this criterion, we filter out 1,434 and 4,939 samples from the CounterFact training split for GPT-2 XL and Llama3-8B, respectively. From the remaining data, we sample 1,000 prompts for the edit dataset and intrinsic dataset respectively. Detailed statistics of the filtered datasets are provided in Table 5.

As the GPT-2 XL and Llama3-8B possess different knowledge on the CounterFact dataset, necessitating the construction of different edit datasets and intrinsic knowledge datasets for each model. Compared to GPT-2 XL, Llama3-8B not only demonstrates more comprehensive knowledge coverage but also exhibits significantly higher prediction confidence for the target facts in CounterFact.

C Implementation Details

C.1 Knowledge Editing

Full fine-tuning for editing(FT): We choose Adam optimizer with learning rate of 5e-5, maximum training epoch of 500, weight decay of 0.01 and early stopping loss of 0.01. We fine-tune all layers for both GPT-2 XL and Llama3-8B.

MEMIT: We employ the same hyper-parameters for MEMIT as the setting in original paper (Meng et al., 2023b). We choose learning rate of 0.5, maximum training step of 50, weight decay of 0.5, and KL factor of 0.0625. For GPT-2 XL, we edit layers ranging from 13 to 17, while for Llama3-8B, we edit layers ranging from 4 to 8.

MALMEN: We adopt the same hyperparameters for MALMEN as those used in the original paper (Tan et al., 2023). We observe that choosing later layers can achieve better edition success rate. We select the model editing hyperparameter such that the post-edit model can predict true target for all the prompts in our edit dataset. Specifically, for GPT-2 XL, we edit layers ranging from 43 to 48, while for Llama3-8B, we edit layers ranging from 27 to 32.

AlphaEdit: We adopt the same hyperparameters for AlphaEdit as those used in the original paper (Fang et al., 2025). We select the model editing hyperparameter such that the post-edit model can predict true target for all the prompts in our edit dataset. For GPT-2 XL, we edit layers ranging from 13 to 17, while for Llama3-8B, we edit layers ranging from 4 to 8.

C.2 Downstream Fine-tuning Tasks

We perform data filtering on the training set of all the downstream task, only keep the data that are irrelevant to the edit knowledge. To fairly compare these edit methods, it is essential to develop a metric to quantify the extent to which fine-tuning influences the model. To ensure that the fine-tuning impact is consistent across the four methods (FT, MEMIT, MALMEN and AlphaEdit), for each of these downstream fine-tuning tasks, we sample and fix an evaluation dataset from the fine-tuning dataset and standardize the training stopping criteria, and employ the same fine-tuning hyperparameter.

For a fine-tuning task, ensuring consistent fine-tuning impacts across different post-edit models is challenging but critical for fairness. We set the same stopping criteria for fine-tuning different post-edit models. Some experiments take more training epoch to reach the stopping criteria. For example, we note that fine-tuning with larger layer threshold needs larger training epoch. Notably, experiments with larger layer thresholds require more training epochs to meet these criteria, as fewer parameters are updated, demanding greater training effort to incorporate the same volume of knowledge into the model. For FT and ROME, we also try different edited layer. We find that for both of the methods, edit early layer (prior to layer 10) can achieve

Table 4: Information for edit and intrinsic data for GPT-2 XL and Llama3-8B.

Task name	Avg true target prob	Avg second ranked target prob
GPT-2 XL Edit	0.407	0.088
GPT-2 XL Intrinsic	0.409	0.092
Llama3-8B Edit	0.467	0.112
Llama3-8B Intrinsic	0.473	0.119

Table 5: Information for fine-tuning dataset.

Task name	Data source	#Train data points	#Validation data points
General text training	Common Crawl	12,000	1,000
Classification	IMDB	25,000	100
SFT	Tulu-3-SFT-mixture dataset	100,000	1,000

similar result.

Next token prediction on general text To assess the impact of fine-tuning on the model's intrinsic and edited knowledge, we conduct fine-tuning using a general-domain text corpus, analogous to continued pre-training on the base language model. Since a small subset of the data suffices to demonstrate the influence, we sampled 12,000 data for training. We evaluate the fine-tuning influence on a validation set, which contains of 1,000 instances sampled from the training set. We employ loss as a metric to measure the volume of knowledge acquired from fine-tuning.

Sentiment classification task To assess the impact of downstream fine-tuning on pre-trained knowledge, we employ a classification task as our benchmark evaluation. Specifically, we utilize the IMDB sentiment analysis dataset which consists of 25,000 movie reviews paired with binary sentiment labels. For the classification architecture, we append a fully connected layer to the final hidden state of the end-of-sequence (EOS) token. Classification accuracy on the validation set can be a fair metric, as it is the target index people are aiming to improve. The validation set contains 100 samples, which has no overlap with the training set.

Instruction fine-tuning task Supervised fine-tuning has emerged as a prevalent downstream adaptation method for LLMs. In this paradigm, each training instance consists of an instruction (question) paired with its corresponding response (answer). For our experiments, we employ the Tulu-3-SFT-mixture dataset (Lambert et al., 2024) for instruction fine-tuning. To assess the training influence, we evaluate model performance on a validation set, which contains 1,000 instances sampled from the training set. We evaluate the training progress by compute the model's loss on the validation set.

C.3 Hyperparameter for fine-tuning

We employ different set of training hyperparameters for Llama3-8B and GPT-2 XL, as their base model training hyperparameters are different. We use learning rate of 5e-5 for Llama3-8B and 1e-3 for GPT-2 XL for all fine-tuning tasks.

Unstructured fine-tuning: In align with its pre-training stage, we employ batch size of 256, AdamW optimizer with beta of (0.9,0.999) and weight decay of 0.01. The model is evaluated on the validation set every 20 steps. The training stops once the validation loss goes below 3.1 for GPT-2 XL and 2.0 for Llama3-8B.

Sentiment classification: We employ batch size of 64, AdamW optimizer with beta of (0.9,0.999) and weight decay of 0.01. The model is evaluated on the validation set every 20 steps. The training stops once the accuracy on the validation set goes above 95%.

Supervised fine-tuning(SFT): For GPT-2 XL, we use an AdamW optimizer with beta of (0.9,0.999) and weight decay of 0.01. Training is performed with a batch size of 128, and the model is evaluated on the validation dataset every 20 steps. We halt training once the validation loss falls below 1.7 for GPT-2 XL and 1.0 for Llama3-8B.

C.4 Experiment Results for Different Edit Layer

We analyze how the choice of editing layer influences model performance during downstream SFT on GPT-2 XL. Figure 5 compares knowledge reten-

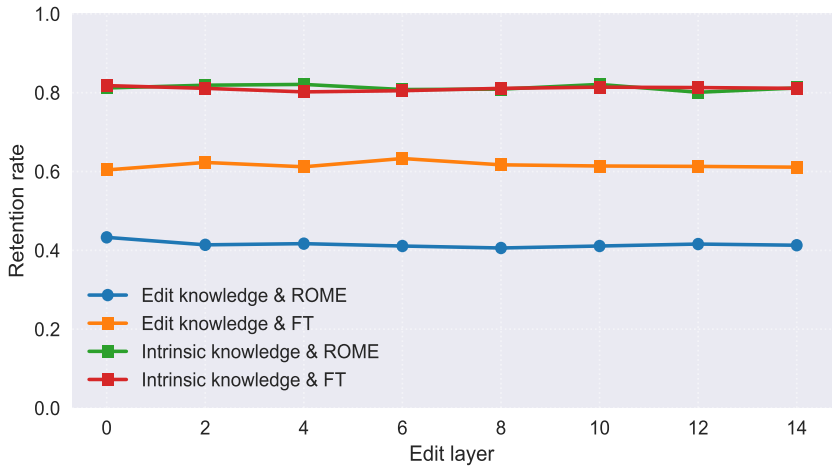

Figure 5: Edit and intrinsic knowledge retention rate for different edit layer for ROME and FT methods.

tion rates for both edited and intrinsic knowledge across different layers when applying ROME and fine-tuning (FT) methods. Our experiments demonstrate that for early layers, ROME and FT achieve comparable edited knowledge and intrinsic knowledge retention rates across all edited layers.

D Fine-Tuning with Frozen Selected Layers

For Locate-then-Edit method, the layer that the edited knowledge being inserted into is where the knowledge located (Meng et al., 2023a; Geva et al., 2021). To enhance locality, a common practice is to fine-tune specific layer while freezing all others, as this approach demonstrates superior locality compared to full-model fine-tuning (Gangadhar and Stratos, 2024). A study has been conducted to determine the optimal layer for knowledge edits (Hase et al., 2023; Meng et al., 2023a). We thus hypothesize that preserving edited knowledge can be improved by avoiding fine-tuning of layers containing the target knowledge. To test this hypothesis, we evaluate two layer-specific fine-tuning strategies on GPT-2 XL:

Fine-tune only later layers. We freeze the early layers and fine-tune the layers beyond a specified threshold layer. As GPT-2 XL contains 48 transformer layers, we set the fine-tuning layer threshold to be 10, 20, 30, 40 and compare their results. From the Figure 6(a), we find that larger layer threshold can improve the edited knowledge retention rate while having similar intrinsic knowledge retention rate. If only layers after 40 are finetuned, the edited knowledge reaches the same level of knowledge retention with intrinsic knowledge. In addition, fine-tuning after a layer threshold has similar intrinsic knowledge retention rate with full fine-tuning retention rate, showing that this method does not impair the intrinsic knowledge.

Fine-tune only layers in a window. We freeze all layers except a window of layers. We set the window size to be 5. To reserve a larger room of layers for the experiment, for both MEMIT and FT, we choose layer 10 instead of 6. From the Figure 6(b), we observe that fine-tuning achieves lowest edited knowledge retention rate when the window centered at layer 10-where the edited knowledge is located-while exhibiting higher edited knowledge retention rate for window does not include the edit layer, or even centered farther from this layer. In contrast, intrinsic knowledge maintains a similar knowledge retention rate across different window center positions.

E Theoretical Details for Section 5.2

E.1 Fisher Spectrum and Effective Training Volume

Lemma 1. *Let K_i be an edit applied R times at parameter locations $\{\theta^{(i,r)}\}_{r=1}^{R}$, producing gradients $g_i^{(r)} = \nabla \log p_{\theta^{(i,r)}}(o_i^* \mid p_i)$ of equal norm c. Define $F_R^{(i)} = \sum_r g_i^{(r)} g_i^{(r)\top}$ and $G_i = [g_i^{(1)}, \ldots, g_i^{(R)}]$. Then $\mathrm{tr}(F_R^{(i)}) = Rc^2$ independent of correlations, while $\mathrm{rank}(F_R^{(i)}) = \mathrm{rank}(G_i) \in [1, R]$, with the upper bound attained iff $\{g_i^{(r)}\}$ are linearly independent.*

Proof. By linearity of trace, $\mathrm{tr}(F_R^{(i)}) = \sum_r \|g_i^{(r)}\|^2 = Rc^2$. For the rank, $F_R^{(i)} = G_i G_i^\top$ implies $\mathrm{rank}(F_R^{(i)}) = \mathrm{rank}(G_i)$, which equals the

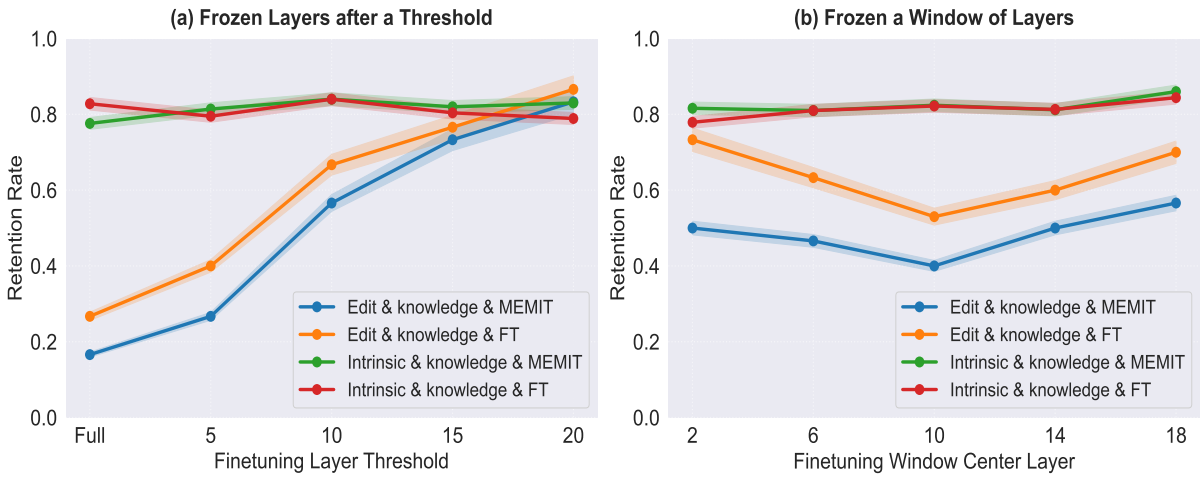

Figure 6: (a) Edit and intrinsic knowledge retention rate for different fine-tuning layer threshold for MEMIT and FT edit. We choose layer 1 as the edit layer for both MEMIT and FT method. (b) Edit and intrinsic knowledge retention rate for different fine-tuning window center layer for MEMIT and FT method. We choose layer 10 as the edit layer for both MEMIT and FT method.

number of linearly independent columns of G_i. □

Proposition 1 (Fisher-rank extension of elasticity volume). *Define* $|D_2|^{(i)}_{\text{eff}} \triangleq \text{rank}(F^{(i)}_R)$. *Then* $|D_2|^{(i)}_{\text{blocked}} \approx 1$ *and* $|D_2|^{(i)}_{\text{shuffled}} \approx R$, *contracting the elasticity ratio of Equation 2 by a factor of* R.

This extension is justified by Bayesian identifiability: the posterior over parameters contracts only along the non-zero eigendirections of the Fisher matrix, so $\text{rank}(F)$ counts the number of identifiable parameter directions. The definition coincides with (Ji et al., 2025) notion of $|D_2|$ in the IID regime where the Fisher matrix is generically full rank, and extends naturally to non-IID training such as repeated editing.

When are shuffled gradients linearly independent? A Taylor expansion yields $g_i^{(r)} \approx g_i(\theta^{(i,*)}) + H_i \xi^{(i,r)}$, where $H_i = \nabla^2 \log p$ is the edit-loss Hessian and $\xi^{(i,r)}$ is the parameter drift from the anchor $\theta^{(i,*)}$. Under blocked repetition, $\{\xi^{(i,r)}\}$ span only the one-dimensional direction $\mathbf{u}_i$; under shuffled repetition, they span the $(N-1)$-dimensional subspace $\text{span}(\{\mathbf{u}_j\}_{j \neq i})$. Provided H_i has rank at least R on this subspace—a mild assumption for over-parameterized LLMs (Papyan, 2020; Sagun et al., 2017)—the images $\{H_i \xi^{(i,r)}\}$ remain linearly independent, yielding $\text{rank}(F^{(i)}_R) \approx R$.

Why blocked repetition is not exactly rank one. Strict $\text{rank} = 1$ would require that no model change occurs between consecutive applications of K_i. In practice, each application itself modifies the model slightly, producing rank mildly above one under blocked ordering. This explains why blocked SRE provides substantial—though sub-shuffled—retention gain in Table 3.

E.2 The Optimization-Repetition Trade-off

The optimized value-vector offset δ^* in AlphaEdit depends implicitly on the key vector k via $\delta^*(k) = \arg\min_\delta \mathcal{L}(\delta; k)$. By the implicit function theorem,

$$\frac{\partial \delta^*}{\partial k} = -(\nabla^2_\delta \mathcal{L})^{-1} \cdot \nabla_\delta \nabla_k \mathcal{L}.$$

As the per-edit optimization budget T grows, δ^* converges to the local minimum where the largest eigenvalues of $\nabla^2_\delta \mathcal{L}$ dominate, increasing $\|\partial \delta^* / \partial k\|$: the optimized δ^* becomes precisely tuned to one specific k and degrades under any perturbation $k \to k + \Delta k$ induced by downstream SFT.

SRE mitigates this by distributing the total optimization effort across R repetitions at evolving model states: the cumulative update approximates $\Delta W^{\text{SRE}} \approx \arg\min_{\Delta W} \mathbb{E}_r \|(W + \Delta W) k^{(r)} - v^{*,(r)}\|^2$, converting a point-wise fit into a distribution-averaged fit. In bias–variance terms, SRE trades per-edit bias for ensemble variance reduction over the k-distribution, mirroring the classical advantage of ensembled weak learners.