Fine-tuning Done *Right* in Model Editing

Anonymous Author(s)
Affiliation
Address
email

Abstract

Fine-tuning, a foundational method for adapting large language models, has long been considered ineffective for model editing. Here, we challenge this belief, arguing that the reported failure arises not from the inherent limitation of fine-tuning itself, but from adapting it to the sequential nature of the editing task, a single-pass *depth-first* pipeline that optimizes each sample to convergence before moving on. While intuitive, this depth-first pipeline coupled with sample-wise updating over-optimizes each edit and induces interference across edits. Our controlled experiments reveal that simply restoring fine-tuning to the standard *breadth-first* (i.e., epoch-based) pipeline with mini-batch optimization substantially improves its effectiveness for model editing. Moreover, fine-tuning in editing also suffers from suboptimal tuning parameter locations inherited from prior methods. Through systematic analysis of tuning locations, we derive **LocFT-BF**, a simple and effective localized editing method built on the restored fine-tuning framework. Extensive experiments across diverse LLMs and datasets demonstrate that LocFT-BF outperforms state-of-the-art methods by large margins. Notably, to our knowledge, it is the first to sustain **100K** edits and **72B**-parameter models, **10 $\times$ beyond prior practice**, without sacrificing general capabilities. By clarifying a long-standing misconception and introducing a principled localized tuning strategy, we advance fine-tuning from an underestimated baseline to a leading method for model editing, establishing a solid foundation for future research.

"*It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so.*"

— Mark Twain

1 Introduction

Model editing has emerged as a promising approach to efficiently update knowledge in Large Language Models (LLMs) without costly retraining [1, 2]. In response, various algorithms have been explored, including parameter-extension [3, 4], meta-learning [5, 6], and locate-then-edit [7, 8, 9] methods. In contrast to these specialized approaches, direct fine-tuning, a widely recognized and effective method for adapting LLMs [10], has nevertheless been consistently dismissed in model editing as a weak baseline, typically attributed to overfitting and catastrophic forgetting [11]. This contradiction raises a critical question: *is fine-tuning inherently unsuitable with model editing, or have we simply been using it wrong?*

In this paper, we argue that this discrepancy arises from the way it has been commonly applied in model editing studies, not the method itself. Our analysis of existing codebases reveals that fine-tuning in model editing deviates from the standard paradigm, reshaped to match the editing task where edit requests naturally arrive one by one. Concretely, rather than iterating over the entire dataset across epochs, it adopts a single-pass procedure, repeatedly optimizing each edit until fully

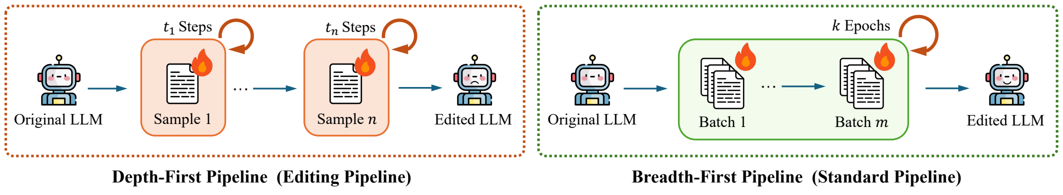


Figure 1: Illustration of edit pipeline (Depth-First) and standard pipeline (Breadth-First).

"memorized" before moving to the next. To distinguish the two, we refer to the conventional form as *fine-tuning with a **breadth-first (BF)** pipeline* and its model-editing adaptation as *fine-tuning with a **depth-first (DF)** pipeline* (typically with batch size 1). Through this lens, two inherent issues of the depth-first pipeline emerge: ❶ its single-pass depth-first pipeline suffers catastrophic forgetting as later edits overwrite earlier ones; ❷ its sample-wise optimization (i.e., batch size of one) tends to produce high-variance gradients, destabilizing the edited model's general capabilities.

To validate this hypothesis, we conduct controlled experiments on representative fine-tuning based editing methods (e.g., FT-M and AdaLoRA [12]) via two orthogonal modifications: ❶ **Pipeline**: switching from a depth-first to a breadth-first pipeline while keeping batch size fixed at 1; ❷ **Granularity**: substituting per-sample updates (batch size = 1) with standard mini-batch optimization under the breadth-first pipeline. As illustrated in Figure 2, our first adjustment to the optimization pipeline alone yields substantial improvements on the editing task. This suggests that the breadth-first pipeline effectively mitigates catastrophic forgetting, a long-standing weakness often criticized in fine-tuning for model editing. Building on the first step, the second controlled experiment changes only the update granularity within the breadth-first pipeline, thereby substantially reducing the degradation of general capabilities in edited models. Overall, we show that simply restoring fine-tuning based editing baselines to the standard breadth-first pipeline with mini-batch optimization yields unexpectedly strong performance on editing tasks—a result surprising at first glance, yet reasonable in hindsight.

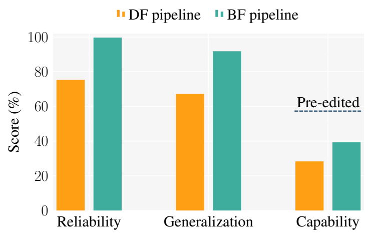


Figure 2: Pipeline comparison of FT-M on LLaMA3-8B with 1000 ZsRE samples.

Despite these encouraging results, the fine-tuning variants we revisited still retain ad-hoc practices from prior model editing research. Specifically, they tune parameters at the locations identified by locate-then-edit methods, which are often suboptimal. This leads to a key yet underexplored question: *which layers and modules of an LLM are most effective to tune for model editing?*

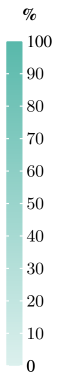
To answer this question, we conduct a systematic study of tuning locations across layers and modules (e.g., attention and MLP) in diverse LLMs. Our experiments reveal that, although optimal configurations can vary across models, a general pattern emerges: tuning the down- or up-projection matrices in later layers often achieves near-perfect editing success while preserving general capabilities. Notably, this strategy remains highly effective even when not optimal.

These analyses lead to LocFT-BF (**L**ocalized **F**ine-**T**uning with **B**readth-**F**irst pipeline), a simple and effective model editing method that restores fine-tuning to its principled configuration: breadth-first pipeline, mini-batch gradient aggregation, and localized parameter updates. Unlike existing methods, LocFT-BF avoids the typical overhead of prior approaches: matrix precomputation required by locate-then-edit methods, additional labeled data required by meta-learning methods, and architectural modifications required by parameter-extension methods. This principled simplicity makes it easy to implement, efficient to run, and broadly applicable across architectures.

To evaluate the effectiveness of our method, we conduct extensive experiments on multiple representative LLMs and datasets. On the widely adopted lifelong editing task, LocFT-BF substantially outperforms state-of-the-art methods, exceeding the best baselines by an average of **33.72%** in editing success rate while consistently maintaining general capabilities of edited models. To further test its limits, we push evaluation to two extremes: **100K sequential edits** and **72B-parameter model**, both an order of magnitude beyond mainstream practice, thereby reflecting scenarios closer to real-world applications. To our knowledge, this is the first method in model editing research that can sustain 100K sequential edits while preserving general capabilities and scales efficiently with stable performance from 7B to 72B models. These results overturn the long-standing view of fine-tuning as a weak baseline and highlight its potential as a scalable solution for model editing.

Table 1: Performance comparison of DF and BF pipelines on mainstream fine-tuning based editors.

	Method	ZsRE						COUNTERFACT					
		Reliability		Generalization		Capability		Reliability		Generalization		Capability	
		DF	BF	DF	BF	DF	BF	DF	BF	DF	BF	DF	BF
LLaMA	FT-L	0.00	0.00	0.00	0.10	14.96	23.47	0.00	0.00	0.00	0.00	15.14	18.17
	FT-M	75.30	99.70	67.20	91.80	28.30	39.30	80.00	99.90	52.60	75.10	29.89	30.87
	AdaLoRA	3.80	90.50	3.30	69.70	44.81	49.89	8.40	96.30	6.10	44.10	49.88	39.27
	RoseLoRA	0.30	1.00	0.00	0.70	57.13	57.38	0.30	0.20	0.00	0.00	57.17	56.66
Mistral	FT-L	0.00	0.00	0.00	0.00	15.73	15.38	0.00	0.00	0.00	0.00	14.93	15.34
	FT-M	41.10	100.00	24.60	83.50	18.14	15.64	59.70	99.90	25.60	59.10	16.94	20.50
	AdaLoRA	2.50	88.50	2.40	74.70	25.36	41.27	5.10	95.50	3.90	44.70	19.14	39.63
	RoseLoRA	0.20	4.10	0.00	3.60	45.24	43.80	0.60	0.80	0.20	0.20	45.57	43.97
Qwen	FT-L	0.10	0.10	0.00	0.10	23.65	29.66	0.10	0.60	0.10	0.50	31.21	33.07
	FT-M	58.70	99.80	34.60	77.60	25.41	34.28	67.70	99.90	25.60	35.40	32.57	33.17
	AdaLoRA	3.40	90.60	2.50	54.00	53.47	51.21	8.90	97.00	4.00	19.10	41.74	53.23
	RoseLoRA	0.00	0.40	0.00	0.10	58.73	58.58	0.20	0.40	0.10	0.10	58.48	58.37



2 Implementation Matters in Fine-tuning

This section revisits the widely reported underperformance of fine-tuning in model editing [4, 9], identifying its root causes in flawed training pipeline and showing how correcting it restores fine-tuning as a competitive editing approach.

2.1 Mis-specified Implementation

Through a detailed examination of existing fine-tuning based editing methods [11, 13], we find that they are typically implemented following the logic of editing tasks rather than the standard fine-tuning paradigm, simulating the sequential arrival of knowledge updates. Specifically, they adopt a sample-by-sample training procedure: repeatedly optimizing each sample to convergence before advancing to the next. We refer to this approach as a *Depth-First (DF) pipeline*. In contrast, as illustrated in Figure 1, the standard *Breadth-First (BF) pipeline* differs in two key aspects: ❶ it iterates over the entire dataset across epochs and ❷ employs mini-batch updates rather than sample-wise optimization. While the DF design appears intuitive in the context of editing tasks, it is not a necessary choice in practice: real-world edits rarely arrive strictly sample by sample, and even when updates are sequential, the BF pipeline can incorporate them incrementally. More importantly, the DF pipeline inherently introduces two risks: ❶ the sequential and single-pass nature makes it prone to later edits overwriting earlier ones; ❷ the sample-wise optimization tends to produce unstable gradients, which can destabilize the general capabilities of edited models.

2.2 Impact of Training Pipeline

To validate our hypothesis regarding pipelines, we compare DF and BF in controlled experiments, fixing batch size at 1 and keeping other settings constant. We evaluate four representative fine-tuning based methods: FT-L [11], FT-M [12], AdaLoRA [14], and RoseLoRA [13], on two mainstream datasets: ZsRE [15] and COUNTERFACT [7], using three popular LLMs: LLaMA-3-8B-Instruct [16], Mistral-7B-v0.1 [17], and Qwen2.5-7B-Instruct [18]. Performance is measured with three metrics: *Reliability*, *Generalization*, and *Capability*. Further details of the methods, datasets, LLMs, and metrics are provided in § 4.1 and Appendix A.1.

Table 1 shows that replacing the DF pipeline with BF improves performance across all methods, with especially large gains for FT-M and AdaLoRA. FT-L and RoseLoRA show only minor improvements due to their design constraints: FT-L optimizes only on the last token of the target answer, while RoseLoRA excessively limits the scope of trainable parameters. Overall, these results highlight the critical role of a proper training pipeline in effective knowledge editing.

Although transitioning from DF to BF pipeline yields substantial gains, especially in editing success, the mechanism behind this gap remains unclear. To investigate this, we compare the learning dynamics of the two pipelines using FT-M on LLaMA-3-8B-Instruct with 1000 ZsRE samples. Specifically, we randomly partition the 1000 samples into five equal shards and track the average editing success of each shard during learning. For DF pipeline, the shards are edited sequentially; and after completing

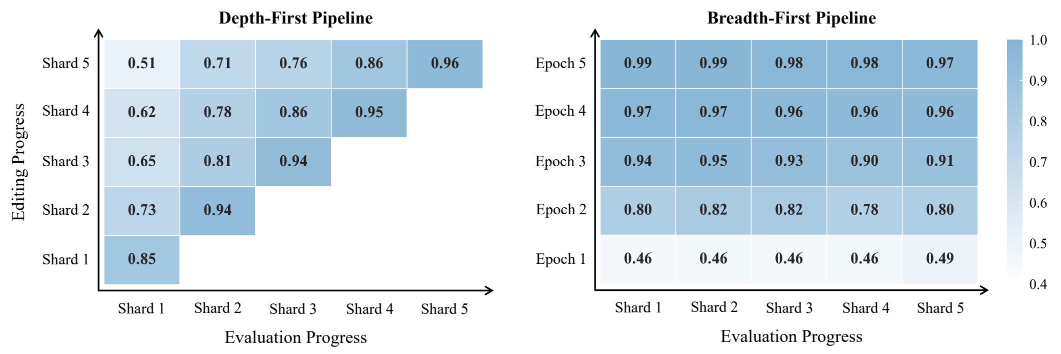


Figure 3: Visualization of the learning dynamics for depth-first and breadth-first pipelines.

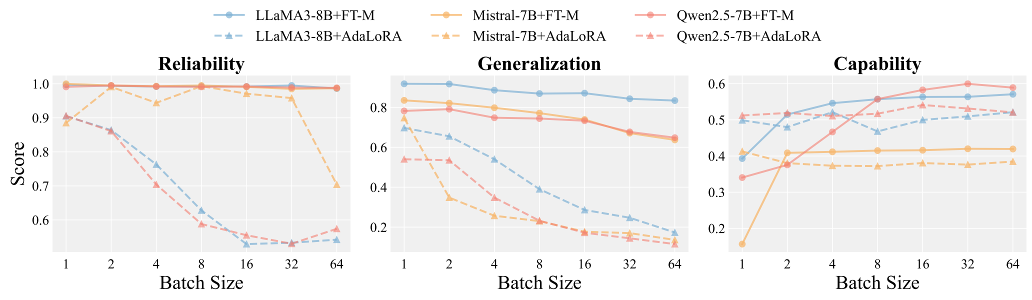


Figure 4: Impact of batch size on editing performance under the BF pipeline on ZsRE.

each shard we evaluate on all previously edited ones to detect potential overwriting. While for BF pipeline, all samples are shuffled each epoch, and after every epoch we evaluate each shard separately.

The visualization results are depicted in Figure 3. Under DF pipeline, earlier shards start with high success rates but decline as later shards are edited. In contrast, under BF pipeline, all shards improve jointly across epochs and converge to near-perfect success rates. These results confirm our hypothesis that the mis-specified DF pipeline induces catastrophic overwriting of earlier edits.

2.3 Impact of Gradient Aggregation

While fixing the training pipeline markedly improves editing success and generalization, the edited models still show clear degradation in general capabilities, especially for FT-M. We next validate the second hypothesized drawback: per-sample updates (batch size = 1) destabilize the edited model. To test this, we replace per-sample updates with mini-batch training within each epoch for both FT-M and AdaLoRA, further aligning fine-tuning–based editing with standard practice.

As shown in Figure 4, increasing the batch size substantially improves the capability performance of FT-M to even surpass AdaLoRA, indicating that mini-batch training stabilizes model states during editing. Conversely, AdaLoRA attains modest gains due to its low-rank design, which already regularizes updates and preserves downstream performance. While larger batches reduce generalization to some extent, more noticeably for AdaLoRA and slightly for FT-M, this effect can be readily mitigated through data augmentation [19], given that each edit currently relies on only a single QA pair. Similar observations regarding the negative impact of large batch sizes on generalization have also been reported by [20].

In summary, aligning fine-tuning based editing with the standard breadth-first, mini-batch paradigm repositions it from a perceived weak baseline to a competitive approach. This finding dispels a long-standing misconception: the widely reported failure arose not from the inherent flaws of fine-tuning, but from the ill-suited implementations in prior work.

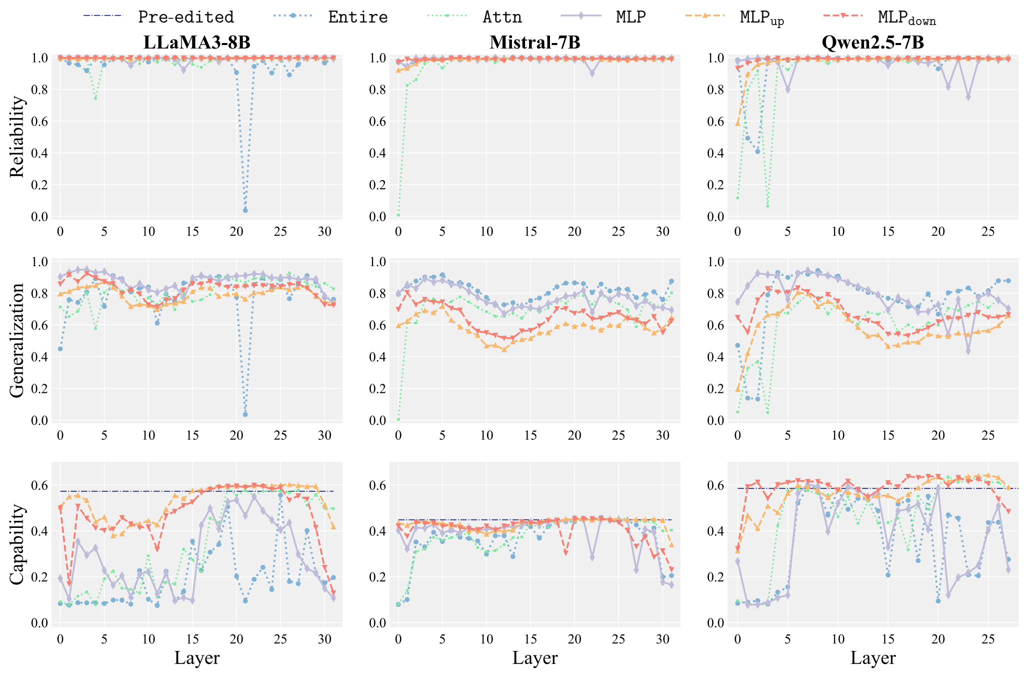


Figure 5: Fine-tuning performance for different locations across three LLMs.

3 Tailoring Fine-tuning for Model Editing

Our studies demonstrate that transitioning to a standard fine-tuning pipeline substantially improves editing performance. Nevertheless, existing fine-tuning variants remain coarse and insufficiently designed, typically tuning the location identified by locate-then-edit methods without theoretical or empirical support, resulting in suboptimal generalization and capability. This raises a critical question: *which locations of an LLM are most effective to tune for model editing?*

To address it, we conduct a comprehensive study of tuning locations to establish a rigorous basis for parameter selection in fine-tuning based editing. Specifically, we examine the same three LLMs as in earlier experiments, along two dimensions: *layer* and *module*. For each layer, we test five candidate modules: entire layer, full attention, full `MLP`, $\texttt{MLP}_{\texttt{up}}$, and $\texttt{MLP}_{\texttt{down}}$. This design is motivated by different emphases in the literature: parameter-efficient fine-tuning [21] typically targets the attention modules for downstream adaptation, whereas editing [7] and mechanistic studies [22] highlight the `MLP`, especially $\texttt{MLP}_{\texttt{down}}$, as the locus of factual knowledge. For example, this yields 160 tuning locations in LLaMA3-8B (32 layers $\times$ 5 modules), each fine-tuned independently. We also evaluated full-parameter and multi-layer fine-tuning; however, their poor performance and disruption of general capabilities led us to exclude them.

For the editing setup, we perform localized fine-tuning on 1000 ZsRE samples and evaluate three metrics: *Reliability*, *Generalization*, and *Capability* (assessed on three representative tasks, GSM8K, MMLU, and WMT, to reduce resource cost). Corresponding results are presented in Figure 5.

Reliability Perspective. Across all models, fine-tuning nearly any module in any layer achieves near-perfect editing success, suggesting that knowledge acquisition capacity is broadly distributed rather than confined to specific parameters. This is consistent with recent studies challenging knowledge localization [23] and showing that both attention and MLP can encode knowledge [24, 25]. However, tuning the entire layer, full attention, or full `MLP` shows slight instability, with occasional drops at certain layers. In contrast, $\texttt{MLP}_{\texttt{down}}$ is the most stable, consistently maintaining high reliability across layers, consistent with prior findings that it is particularly specialized for knowledge update [22, 7].

Generalization Perspective. Unlike the uniformly high reliability, generalization varies notably across layers and modules, though the overall pattern is consistent across all three LLMs. *From the module perspective*, coarser-grained components such as entire layer and full `MLP` generalize better than finer-grained counterparts like $\texttt{MLP}_{\texttt{up}}$ and $\texttt{MLP}_{\texttt{down}}$, suggesting broader parameter updates may facilitate more robust memorization and stronger generalization to paraphrased prompts. *From* *the layer perspective*, generalization follows a consistent trajectory: it peaks in early layers (e.g., layers 3–8), declines in middle layers, shows a smaller secondary peak in later layers (e.g., layers 20–25), and then declines again. This highlights the non-trivial impact of layer choice and the need for systematic empirical assessment in guiding effective selection.

Capability Perspective. Unlike generalization, which favors coarser components, capability is stronger and more stable with fine-grained modules, i.e., $\texttt{MLP}_{\texttt{up}}$ and $\texttt{MLP}_{\texttt{down}}$. This shows that more localized updates cause less disruption, helping edited models preserve general capability. *From the layer perspective*, capability follows a trajectory similar to generalization, with peaks in early and later layers. However, unlike generalization's preference for the early layers, capability is more pronounced in the later layers, further underscoring their trade-off.

Table 2: Selected tuning locations.

Model	Tuning Location
LLaMA3-8B	$\texttt{MLP}_{\texttt{down}}$ in layer 22
Mistral-7B	entire layer 23
Qwen2.5-7B	$\texttt{MLP}_{\texttt{down}}$ in layer 6

Overall, since most locations yield high reliability, tuning location selection mainly depends on the trade-off between generalization and capability. We prioritize capability for two reasons: ❶ preserving LLMs' general abilities is essential for editing; ❷ generalization, even if suboptimal, can be improved with data augmentation [19], as current editing uses only one QA pair per fact. Based on this and our experiments, we identify the relatively optimal tuning locations for each LLM, as in Table 2. Overall, the results reveal a consistent pattern: editing $\texttt{MLP}_{\texttt{down}}$ in the later layers minimizes capability degradation while retaining near-perfect reliability and acceptable generalization. Notably, this strategy remains highly effective even when not optimal. For instance, although Qwen performs best in earlier layers, later layers still achieve strong results.

Integrating three key factors, namely breadth-first pipeline, mini-batch gradient aggregation, and tuning-location selection, yields **LocFT-BF**, a simple yet powerful fine-tuning method for model editing. Owing to the inherent simplicity and extensibility of fine-tuning, LocFT-BF is broadly applicable, plug-and-play, and avoids the overhead of locate-then-edit (matrix precomputation) and meta-learning (extra training). Although LocFT-BF requires specifying tuning locations, our study offers an initial, relatively stable selection strategy to address this.

4 Benchmarking LocFT-BF in Lifelong Editing

Although LocFT-BF enhances fine-tuning for model editing, its comparative performance against state-of-the-art techniques is unclear. We thus conduct a comprehensive evaluation across diverse LLMs and datasets under a lifelong editing setup.

4.1 Experimental Setup

Editing Methods. To ensure comprehensive coverage, we compare LocFT-BF against six representative editing methods across three categories: parameter-extension (**WISE**, 4), meta-learning (**RLEdit**, 6 and **UltraEdit**, 26), and locate-then-edit (**MEMIT**, 8, **RECT**, 27, and **AlphaEdit**, 9).

Edited LLMs. Following prior work [9, 26], we evaluate three leading open-source LLMs: LLaMA-3-8B-Instruct [16], Mistral-7B-v0.1 [17], and Qwen2.5-7B-Instruct [18].

Editing Datasets. In line with prior studies [4, 9], we randomly sample 3000 instances from ZsRE [15], COUNTERFACT [7], and WikiBigEdit [28] for large-scale editing evaluation.

Evaluation Metrics. We evaluate editing techniques from four key properties: ❶ *Reliability*: success rate of editing; ❷ *Generalization*: adaptability of edited knowledge to rephrased prompts; ❸ *Capability*: preservation of general capabilities, measured as the average accuracy of edited models on MMLU [29], Natural Questions [30], SST2 [31], WMT [32], and GSM8K [33]; ❹ *Efficiency*: average time to perform each edit. Notably, we evaluate *Reliability* and *Generalization* using the WILD framework [34], which employs autoregressive decoding instead of conventional teacher forcing generation to align with practical deployment scenario and avoid overestimation.

Detailed information regarding method implementation, LLM configuration, dataset preparation, and evaluation procedures is provided in Appendix A.1.

Table 3: Comparison of LocFT-BF with existing methods on lifelong editing task. The best results are denoted in **bold** and the second-best results are underlined.

Data	**Method**	**LLaMA3-8B**				**Mistral-7B**				**Qwen2.5-7B**			
		Rel.	**Gen.**	**Cap.**	**Time**	**Rel.**	**Gen.**	**Cap.**	**Time**	**Rel.**	**Gen.**	**Cap.**	**Time**
	Pre-edited	–	–	57.26	–	–	–	44.82	–	–	–	58.52	–
ZsRE	MEMIT	26.23	23.30	25.67	9.90	24.30	19.60	18.11	10.28	39.57	32.10	47.87	9.93
ZsRE	RECT	0.03	0.03	14.98	25.38	0.17	0.27	14.88	25.79	9.70	8.20	16.06	24.31
ZsRE	WISE	4.17	3.50	–	14.42	15.87	11.33	–	13.02	7.67	5.23	–	30.96
ZsRE	AlphaEdit	64.50	40.57	54.71	12.31	3.30	3.00	15.13	10.95	2.70	2.33	16.31	10.55
ZsRE	RLEdit	66.67	59.40	**57.41**	0.58	26.10	19.53	21.30	0.47	54.57	47.60	**60.74**	0.65
ZsRE	UltraEdit	34.93	22.67	55.93	**0.22**	40.43	23.40	**44.22**	**0.04**	40.10	19.77	59.25	**0.27**
ZsRE	**LocFT-BF**	**98.97**	**75.83**	57.04	0.27	**98.93**	**49.57**	42.73	0.31	**98.87**	**74.37**	59.07	0.49
COUNTERFACT	MEMIT	71.90	**48.47**	19.30	9.34	37.83	28.17	15.59	9.06	68.37	**40.90**	44.00	8.70
COUNTERFACT	RECT	0.53	0.13	14.85	21.68	0.77	0.37	15.67	22.21	0.00	0.00	14.81	21.69
COUNTERFACT	WISE	19.80	13.13	–	12.18	25.13	4.60	–	10.28	20.17	10.20	–	27.18
COUNTERFACT	AlphaEdit	94.27	39.90	54.09	10.60	6.67	6.70	15.47	9.75	32.17	18.10	17.46	9.46
COUNTERFACT	RLEdit	65.33	33.23	55.45	0.46	36.30	17.07	23.53	0.40	44.33	18.90	58.48	0.45
COUNTERFACT	UltraEdit	68.33	31.20	56.85	**0.18**	57.60	22.60	**44.91**	**0.03**	41.33	15.33	**59.01**	**0.18**
COUNTERFACT	**LocFT-BF**	**99.73**	33.23	**57.13**	0.38	**99.67**	**39.53**	41.46	0.24	**99.73**	11.77	58.53	0.48
WikiBigEdit	MEMIT	10.77	10.80	23.57	10.39	12.47	10.07	18.26	11.02	31.03	25.27	47.92	10.69
WikiBigEdit	RECT	0.00	0.00	14.90	30.43	0.00	0.00	14.81	30.41	1.87	0.97	14.78	27.96
WikiBigEdit	WISE	30.33	27.03	–	15.14	38.03	32.53	–	11.50	34.40	30.63	–	33.80
WikiBigEdit	AlphaEdit	64.73	51.17	54.14	14.18	3.37	3.57	14.69	12.05	4.83	3.67	15.13	11.68
WikiBigEdit	RLEdit	71.10	63.27	57.28	0.60	60.83	46.47	36.20	0.38	66.40	58.50	59.36	0.47
WikiBigEdit	UltraEdit	72.67	65.37	**57.98**	**0.20**	33.87	27.00	25.43	**0.05**	74.20	**59.17**	58.82	**0.31**
WikiBigEdit	**LocFT-BF**	**98.87**	**73.77**	56.93	0.54	**98.90**	**74.97**	**42.39**	0.39	**99.43**	53.30	**59.83**	0.98

4.2 Results & Analysis

The results are reported in Table 3. To structure the analysis, we examine them from four evaluation perspectives: reliability, generalization, capability, and efficiency.

Reliability. Remarkably, LocFT-BF consistently attains the highest reliability across all nine dataset–LLM combinations, with absolute gains of **33.72%** on average and up to **58.50%** over the second-best. This demonstrates that fine-tuning can achieve effective knowledge updates. Although recent methods such as AlphaEdit, RLEdit, and UltraEdit achieve relatively strong reliability compared to traditional approaches, they exhibit notable instability across LLMs and datasets.

Generalization. LocFT-BF exhibits superior generalization, achieving optimal performance in six out of nine dataset–LLM combinations. The two least effective cases are from the COUNTERFACT dataset, whose generalization prompts are constructed by prepending irrelevant text rather than direct paraphrasing, which makes generalization particularly challenging. Methods like MEMIT and AlphaEdit explicitly enhance robustness to such noise through data augmentation, accounting for their relative advantage over LocFT-BF on this dataset, as detailed in Appendix A.2.

Capability. LocFT-BF and recently proposed methods, including AlphaEdit, RLEdit, and UltraEdit, exhibit advantages in preserving the general capabilities of edited models under large-scale editing. However, these baseline methods show noticeable instability across LLMs: RLEdit and UltraEdit struggle on Mistral-7B, while AlphaEdit, despite performing well on LLaMA3-8B, fails to generalize to the other two LLMs. In contrast, LocFT-BF demonstrates the strongest stability, consistently maintaining high capability across all evaluated LLMs and datasets. This highlights the key strength of fine-tuning: its simple and general design makes it broadly adaptable across architectures.

Efficiency. LocFT-BF, RLEdit, and UltraEdit achieve top-tier efficiency, completing each edit within one second by directly updating model parameters. Conversely, other baselines are nearly 50 times slower, primarily due to their more complex procedures, such as auxiliary module optimization (WISE) and additional matrix calculation (MEMIT, RECT, and AlphaEdit), which introduce substantial computational overhead.

In summary, LocFT-BF delivers superior performance across all key evaluation dimensions while maintaining strong stability, establishing it as an effective and robust technique for model editing.

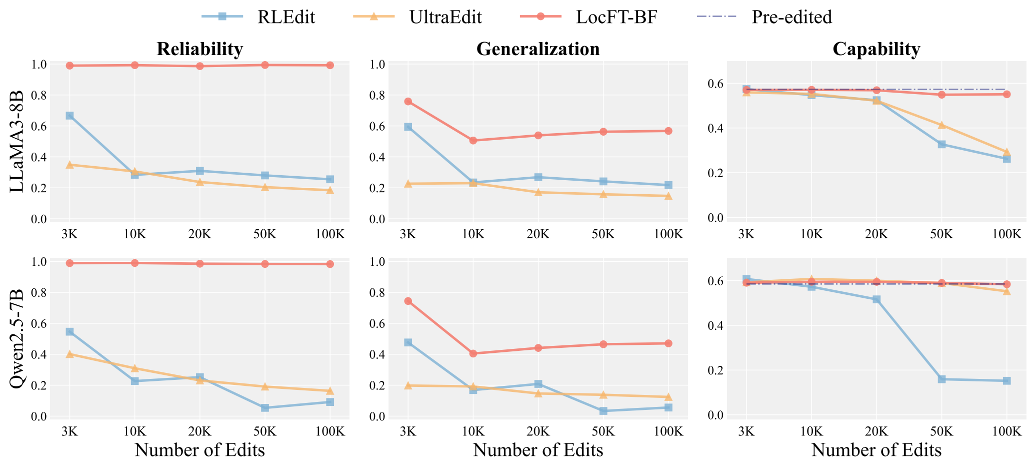


Figure 6: Evolution of editing performance during the scaling process to 100K edits.

5 Scaling towards Real-world Editing

We further extend our evaluation to more realistic scenarios by scaling data volume and model size to 10× the scale of mainstream practice, thereby probing the limits of our approach.

5.1 Scaling to Larger Data Volumes

Owing to the limited capacity of existing editing techniques, prior evaluations have typically been restricted to approximately 3000 edits. However, such a scale is insufficient to reflect real-world requirements for lifelong editing. To address the limitation, we scale the number of edits to better reflect practical scenarios.

Experimental Setup. We increase the number of edits from 3K to 100K using the ZsRE dataset. For this experiment, we evaluate LocFT-BF, RLEdit, and UltraEdit, the only three methods that maintain model capabilities under 3K sequential edits, on LLaMA3-8B and Qwen2.5-7B.

Experimental Results. As shown in Figure 6, LocFT-BF exhibits significant superiority over the leading baselines across all metrics when scaled to large edit volumes. Even at 100K edits, it maintains near-perfect success rates while preserving the general capabilities of edited models. To our knowledge, it is the only method to achieve both effective knowledge updates and capability retention at this scale. Although its generalization declines, LocFT-BF still substantially surpasses the baselines, and this aspect can be further improved with data augmentation. In contrast, RLEdit and UltraEdit struggle to scale, with model capabilities decline sharply when edits exceed 20K. More critically, their consistently low editing success rates reveal a fundamental limitation in achieving reliable knowledge updates.

5.2 Scaling to Larger Models

The intricate designs of most existing model editing techniques lead to heavy computational overhead, which has confined mainstream evaluations to 7B-level LLMs. For example, locate-then-edit approaches such as ROME and AlphaEdit require computing large covariance and projection matrices, incurring substantial memory and time costs that make scaling to larger models practically infeasible. In contrast, fine-tuning is inherently lightweight and easily extensible, allowing us to push beyond this long-standing 7B barrier and evaluate LocFT-BF on much larger models of practical scenarios.

Experimental Setup. We apply LocFT-BF to edit a series of Qwen2.5 models ranging from 7B to 72B. We evaluate with 1,000 ZsRE samples to balance evaluation coverage and computational feasibility for large models. Considering an exhaustive search for the optimal tuning position in larger models, such as Qwen2.5-72B, is computationally prohibitive, we design two heuristic strategies to determine the target layer, based on the optimal position identified in Qwen2.5-7B. ❶ Default Position: we directly apply the same absolute position (i.e., Layer6.$\texttt{MLP}_{\texttt{down}}$) to all larger models, serving as a straightforward baseline. ❷ Proportional Position: preserve the relative depth of the

Table 4: Editing performance across Qwen2.5 models as scale increases from 7B to 72B parameters.

Status	**Qwen2.5-7B**			**Qwen2.5-14B**			**Qwen2.5-32B**			**Qwen2.5-72B**		
	Rel.	**Gen.**	**Cap.**	**Rel.**	**Gen.**	**Cap.**	**Rel.**	**Gen.**	**Cap.**	**Rel.**	**Gen.**	**Cap.**
Pre-edited	-	-	58.52	-	-	62.56	-	-	63.72	-	-	64.84
Default Pos.	99.20	83.40	59.33	99.00	78.80	61.62	99.30	70.00	60.27	98.60	65.30	63.64
Proportional Pos.	99.20	83.40	59.33	99.50	77.90	61.12	99.40	68.20	64.72	99.30	68.30	63.87

target layer to account for increasing model depth. For instance, since Layer 6 is the 7th layer in the 28-layer Qwen2.5-7B, we select the corresponding 20th layer (i.e., Layer19.$\texttt{MLP}_{\texttt{down}}$) for the 80-layer Qwen2.5-72B, corresponding to $80 \times (7/28) = 20$.

Experimental Results. The results in Table 4 demonstrate that LocFT-BF readily scales to larger models, achieving reliable knowledge updates while maintaining general capabilities. Notably, both heuristic strategies for tuning position selection prove effective, eliminating the need for an exhaustive and computationally expensive position search. For mid-sized models such as Qwen2.5-14B, the Default Position strategy is sufficient for strong performance, whereas for larger models like Qwen2.5-32B, the Proportional Position strategy yields better results. These results highlight that LocFT-BF can be seamlessly extended from 7B to 72B models without redesign, underscoring its simplicity and plug-and-play scalability in practical deployment.

6 Related Works

Model Editing Methodologies. Existing approaches to model editing can be broadly categorized into three groups. ❶ *Parameter-extension.* These methods introduce additional trainable components decoupled from pretrained parameters to encode new knowledge. Representative designs include extra neurons (T-Patcher, 35), codebooks (GRACE, 3), and auxiliary memory modules (WISE, 4). ❷ *Meta-learning.* KE [36] and MEND [5] train a hypernetwork to predict parameter updates for knowledge editing. RLEdit [6] further enhances this approach with reinforcement learning, enabling the hypernetwork to adaptively produce parameter updates conditioned on the edited model's evolving state. ❸ *Locate-then-edit.* Building upon research into knowledge mechanisms of LLMs [22, 37], ROME [7] and MEMIT [8] apply causal tracing to identify knowledge-critical parameters and apply localized updates for precise editing. AlphaEdit [9] augments this line by projecting parameter updates onto the null space of preserved knowledge to mitigate disruption in lifelong editing.

Fine-tuning Based Model Editing. FT-L [11, 7] and FT-M [12] directly apply fine-tuning to parameters identified by the locate-then-edit paradigm but suffer from catastrophic forgetting. To mitigate this limitation, recent studies have focused on two primary strategies. ❶ *Parameter-efficient fine-tuning.* MELO [38] and RoseLoRA [13] incorporate low-rank adaptation into model editing to constrain the interference induced by fine-tuning. ❷ *Data augmentation.* Since prior methods rely on a single QA pair per fact, recent work [19, 39] proposes to enrich the editing data, thus improving the acquisition of new knowledge during fine-tuning.

In contrast to previous research that focused on architectural refinements or data augmentation, our study is, to the best of our knowledge, the first to revisit the failure of fine-tuning-based model editing, We attribute this failure to a flawed implementation. By correcting this and customizing tuning locations, we have shown that fine-tuning can be an effective paradigm, disproving the misconception that it is unsuitable for model editing.

7 Conclusion

In this paper, we present the first re-examination of fine-tuning based model editing, a technique long regarded as a weak baseline. By identifying and correcting critical flaws in existing implementations, we unlock the true potential of fine-tuning and overturn the prevailing misconception of its limitations. Building on this, we propose LocFT-BF, a pure localized fine-tuning approach grounded in comprehensive empirical analysis of tuning positions. Extensive experiments demonstrate that LocFT-BF not only significantly surpasses state-of-the-art editing methods but also scales effectively to massive editing workloads (100K samples) and large models (up to 72B parameters). This work redefines the role of fine-tuning in model editing, establishing it as a powerful and practical technique, while pushing the field closer to real-world deployment.

References

[1] Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore, December 2023. Association for Computational Linguistics.

[2] Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3), November 2024.

[3] Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[4] Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022.

[6] Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models, 2025.

[7] Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

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

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

[10] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025.

[11] Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models, 2020.

[12] Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. A comprehensive study of knowledge editing for large language models, 2024.

[13] Haoyu Wang, Tianci Liu, Ruirui Li, Monica Xiao Cheng, Tuo Zhao, and Jing Gao. RoseLoRA: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 996–1008, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[14] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and et al. The llama 3 herd of models, 2024.

[17] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, and et al. Mistral 7b, 2023.

[18] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[19] Govind Krishnan Gangadhar and Karl Stratos. Model editing by standard fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5907–5913, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[20] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

[21] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[22] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[23] Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Knowledge localization: Mission not accomplished? enter query localization! In *The Thirteenth International Conference on Learning Representations*, 2025.

[24] Yifan Wei, Xiaoyan Yu, Yixuan Weng, Huanhuan Ma, Yuanzhe Zhang, Jun Zhao, and Kang Liu. Does knowledge localization hold true? surprising differences between entity and relation perspectives in language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 4118–4122, New York, NY, USA, 2024. Association for Computing Machinery.

[25] Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: precise model editing in a transformer. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.

[26] Xiaojie Gu, Guangxu Chen, Jungang Li, Jia-Chen Gu, Xuming Hu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in large language models, 2025.

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

[28] Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. Wikibigedit: Understanding the limits of lifelong knowledge editing in LLMs. In *Forty-second International Conference on Machine Learning*, 2025.

[29] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

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

[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[32] Ond rej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics.

[33] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[34] Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. The mirage of model editing: Revisiting evaluation in the wild. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15336–15354, Vienna, Austria, July 2025. Association for Computational Linguistics.

[35] Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*, 2023.

[36] Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[37] Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[38] Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.

[39] Zihao Wei, Liang Pang, Hanxing Ding, Jingcheng Deng, Huawei Shen, and Xueqi Cheng. Stable knowledge editing in large language models. *CoRR*, abs/2402.13048, 2024.

[40] Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. EasyEdit: An easy-to-use knowledge editing framework for large language models. In Yixin Cao, Yang Feng, and Deyi Xiong, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[41] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.

[42] Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. The butterfly effect of model editing: Few edits can trigger large language models collapse. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5419–5437, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[43] Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15202–15232, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

A Appendix

A.1 Detailed Experimental Setup

A.1.1 Editing Methods

FT-L [11, 7] fine-tunes the MLP of a specific layer identified through causal tracing in ROME [7], while augmenting the objective with an l_∞-norm regularization that explicitly limits parameter deviations between the original and edited models to mitigate side effects on unrelated knowledge and capabilities. However, FT-L deviates from the standard fine-tuning paradigm by leveraging only the last token of the target answer as supervision, which severely undermines its editing success.

FT-M [12] addresses the supervision limitation of FT-L by applying a cross-entropy loss over the entire target answer while masking the prompt, thereby aligning more closely with the standard fine-tuning practices and yielding substantial gains in editing success.

AdaLoRA [14] enhances vanilla Low-Rank Adaptation (LoRA) by adaptively allocating the parameter budget among weight matrices according to their importance score. [12] directly adapt this technique to model editing and report competitive results.

RoseLoRA [13] is a novel parameter-efficient fine-tuning (PEFT) method that introduces row and column-wise sparsity on the product of low-rank matrices. This approach allows it to selectively update only the most critical parameters of a pretrained language model, ensuring efficient and precise updates while preserving the irrelevant knowledge.

WISE [4] targets the lifelong editing task with a dual-memory architecture composed of a main memory for pretrained knowledge, a side memory for edited knowledge, and a router that directs queries between them. By explicitly decoupling edited knowledge from pretrained knowledge, WISE effectively mitigates interference with the original model.

RLEdit [6] reformulates hypernetwork-based lifelong editing as a reinforcement learning problem, where target edits and the state of the edited model constitute the environment, editing losses serve as rewards, and the hypernetwork is optimized at the sequence level as the policy. This design enables the hypernetwork to precisely track parameter changes in LLMs during editing and generate more accurate updates for lifelong knowledge editing.

UltraEdit [26] proposes a training-, subject-, and memory-free editing framework that computes parameter shifts through lightweight linear algebra operations, enabling fast and consistent updates with minimal overhead. To support lifelong adaptation, it further employs a lifelong normalization strategy that continually updates feature statistics, allowing the model to adapt to distributional shifts while preserving consistency over time.

MEMIT [8] extends ROME by scaling its mechanism from single-layer to multi-layer editing. It first identifies knowledge-relevant layers and modules through causal tracing analysis and then applies rank-one matrix update at the identified locations to perform target edits. By propagating modifications across multiple successive layers, MEMIT enables efficient batch editing of large-scale knowledge.

RECT [27] builds on ROME by introducing a regularization strategy to mitigate overfitting and reduce noise induced by editing. Specifically, it quantifies the importance of each weight element by the absolute value of its relative change and updates only the top-$k\%$ elements. This selective update effectively suppresses side effects while preserving editing performance.

AlphaEdit [9] augments locate-then-edit methods by projecting parameter changes onto the null space of preserved knowledge. This projection is theoretically proven to keep the outputs of post-edited LLMs unchanged when queried about preserved knowledge, thereby mitigating the issue of disruption.

For each baseline, we adopt an empirically tuned batch size within the available computational resources for fair comparison, preserving all other configurations from their official implementations.

A.1.2 Edited LLMs

LLaMA-3-8B-Instruct [16] is a leading 8-billion-parameter instruction-tuned model from the LLaMA family of Meta AI. It is designed primarily for dialogue applications and outperform many existing open-source chat models on common industry benchmarks. In addition, it has been further optimized to enhance both helpfulness and safety.

Mistral-7B-v0.1 [17] is a resource-efficient yet highly capable foundation model with 7 billion parameters. It consistently outperforms LLaMA-2-13B across all evaluated benchmarks and even surpasses LLaMA-1-34B on tasks involving reasoning, mathematics, and code generation.

Qwen2.5-7B-Instruct [18] is a superior instruction-tuned model with 7 billion parameters, trained on 18 trillion tokens with over 1M supervised finetuning samples and multistage reinforcement learning. It demonstrates strong performance across language understanding, reasoning, mathematics, and coding, offering competitive capability while being resource-efficient.

All models adopt greedy decoding for generation, consistent with mainstream practice in model editing [1, 40].

A.1.3 Editing Datasets

ZsRE [15] is a widely used dataset in model editing, originally developed for zero-shot relation extraction. Following its adaptation by [36], the original answers were replaced with counterfactual ones to ensure that models had no prior exposure to the target facts, thereby providing a reliable benchmark for evaluating editing methods.

COUNTERFACT [7] is a challenging dataset specifically curated for model editing. It comprises 21,919 counterfactual statements whose target answers are initially assigned low probabilities by models, making it a rigorous benchmark for evaluating editing techniques on more challenging modifications.

WikiBigEdit [28] is a large-scale lifelong editing benchmark built through a fully automated data extraction pipeline that continuously incorporates new factual edits, ensuring long-term applicability for factuality evaluation. Its initial release contains over 500K question–answer pairs, providing a realistic setting for large-scale factual updates and better approximating deployment-time requirements.

Unlike the counterfactual knowledge in ZsRE and COUNTERFACT, WikiBigEdit consists of real-world facts from Wikipedia, some of which may have already been memorized by the evaluated LLMs. To ensure the learning of new knowledge, we excluded all samples that could be correctly answered by any of the three LLMs.

A.1.4 Representative Tasks

We adopt *Capability*, measured via the lm-evaluation-harness [41], in place of the traditional *Locality* metric [1] to more directly and rigorously assess whether editing perturbs unrelated knowledge and capabilities [42, 43].

MMLU [29] is a massive multitask benchmark consisting of multiple-choice questions across 57 subjects, spanning elementary mathematics, US history, computer science, law, and other areas that are important for people to learn. To reduce the prohibitive evaluation cost of the full benchmark, we randomly sample 500 questions from each subject, resulting in a balanced test set of 28,500 instances for our capability evaluation.

Natural Questions [30] is an open-domain question answering benchmark, where queries are drawn from real anonymized search requests issued to Google, and answers are annotated from corresponding Wikipedia pages by human participators. We adopt its test set of 3610 question–answer pairs to evaluate the capability of target LLMs.

SST2 [31] is a sentiment classification benchmark derived from the Stanford Sentiment Treebank. It consists of single sentences from movie reviews, each annotated with binary sentiment labels (positive or negative, with neutral cases removed). We utilize its standard test set to evaluate the sentiment classification capability of examined LLMs.

WMT [32] is a benchmark series from the Workshop on Machine Translation, covering multiple years and language pairs. In our experiments, we adopt the WMT16 German–English (de–en) dataset to evaluate the translation capability of target LLMs.

GSM8K (Grade School Math 8K) [33] is a benchmark of 8500 high quality linguistically diverse grade school math word problems. It was created by OpenAI to support the task of question answering

```
"Edit Prompt"      : "What sport does Dave Winfield play? They play",
"Rephrased Prompt" : "Andrey Tcheboharev\n( 9.) What sport does Dave Winfield play? They play",

"Augmented Context Templates":  ["The 2019-20 season has been. {}", "Therefore, we must not forget the
↪  importance of. {}", "Because I am a woman: The impact of. {}", "I have to admit, I was a bit. {}",
↪  "I have always been a fan of the. {}"]
```

Figure 7: An illustrative example from the COUNTERFACT dataset, including an edit prompt and its corresponding rephrased prompt. The figure also presents representative augmented context templates used by MEMIT and AlphaEdit, where the edit prompt is inserted into the placeholder { } to form diverse training prompts.

on basic mathematical problems that require multi-step reasoning. We use its test set of 1319 problems to evaluate the mathematical reasoning ability of LLMs.

A.2 Interpreting Generalization on COUNTERFACT

The observed generalization drop on COUNTERFACT is a common issue for most editing techniques. The underlying reason lies in the dataset's unique evaluation protocol for generalization: instead of direct paraphrasing the edit prompt, COUNTERFACT constructs its generalization prompt by prepending irrelevant text to the edit prompt, as present in Figure 7. Such disruptive context makes generalization particularly challenging, especially for most methods that learn new knowledge only from the edit prompt. However, methods such as MEMIT and AlphaEdit explicitly enhance robustness by augmenting each edit prompt with several randomly prefixed text, as shown in Figure 7, which explains their relative advantage on COUNTERFACT. Overall, the lower generalization of LocFT-BF on this dataset reflects the unique evaluation design rather than the limitation of fine-tuning, and can be mitigated by incorporating prefix-based augmentation if necessary.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are empirically verified in Section 2-5, in particular Table 3.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our paper in conclusions.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We present all information needed for reproduction our results in Section 4.1 and Appendix A.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will provide all the data, code, and instructions used in the final paper.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide experimental details in Section 4.1 and Appendix A.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide error bars in Table 1, Table 3 and Figure 4-6, and conduct corresponding analysis in Section 2-5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: : We provide the information about the computer resources in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully read and understood the NeurIPS Code of Ethics and have made every effort to adhere to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Since our paper focus on algorithm-oriented research, there is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release any new models or datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the datasets and models used in our paper are properly cited and introduced in Section 4.1 and Appendix A.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release the code along with detailed README files.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not use crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our paper does not use crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in our paper does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.