FINE-TUNING DISRUPTS EDIT READOUT: A MECHANISTIC PERSPECTIVE ON MODEL EDITING ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model editing provides a precise way to update the parametric knowledge of Large Language Models (LLMs). However, it has been shown that fine-tuning of edited models for downstream adaptation will erase the intended edits in existing works. This paper studies this problem through the lens of layer-wise information flow in Transformers. Our systematic layer-wise ablation study reveals that edited memory is mainly anchored in shallow editing layers, while fine-tuning primarily disrupts the deep editing layers and adjacent post-edit layers, jointly forming the readout circuitry that decodes this memory into the final prediction. As a result, reverting this readout circuitry alone is often sufficient to recover editing success. We further verify this layer-wise information-flow perspective through null-space analysis and attention-based visualization. Motivated by this key finding, we propose Layer-wise Model Interpolation (LMI), which selectively restores the disrupted readout circuitry to preserve edited knowledge while retaining downstream adaptation. LMI interpolates between the edited and fine-tuned model weights with layer-aware coefficients, up-weighting the edited model in the readout circuitry and the fine-tuned model elsewhere. Across four public datasets, three language models, and two editing methods, LMI achieves the best balance between edit preservation and downstream utility. Code is available at `https://anonymous.4open.science/r/lmi-FFF0/`

1 INTRODUCTION

Large language Models (LLMs) manage memory with two distinct paradigms (Zhao et al., 2025). Short-term memory is saved in the context (Behrouz et al., 2025), whilst long-term memory is saved in parametric weights inside the Transformer architecture (Vaswani et al., 2017). With short-term memory being easily adapted (Zhang et al., 2026), how to adapt long-term memory remains a longstanding challenge due to catastrophic forgetting (Kirkpatrick et al., 2017; Luo et al., 2025; Li et al., 2024a).

Fine-tuning via gradient descent remains the dominant approach for long-term memory updating, but it typically requires substantial data to avoid overfitting (Lai et al., 2025). In contrast, model editing, emerging from mechanistic interpretability research, offers a more precise and data-efficient alternative (Meng et al., 2022; 2023; Fang et al., 2025). However, once an edited model is deployed, it is often further adapted to downstream tasks through conventional fine-tuning. Existing studies have shown that model editing is highly brittle under such adaptation (Wen & Zhang, 2025; Cheng et al., 2025): downstream fine-tuning can overwrite edited knowledge, raising concerns about the practical utility of model editing.

Despite recent studies on the robustness of model editing under fine-tuning, a key research gap remains: it is unclear why such edits are not retained after fine-tuning. This paper offers a mechanistic perspective on the brittleness of model editing from the lens of layer-wise information flow. As illustrated in Fig. 1(C), we partition the Transformer layers into five contiguous groups: *pre-editing, shallow editing, deep editing, adjacent post-editing*, and *remote post-editing layers*. The deep editing and adjacent post-editing layers together form the *readout circuitry* of the edit signal, which is the central object of our analysis. Specifically, through layer-wise reversion, editing success surges back

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

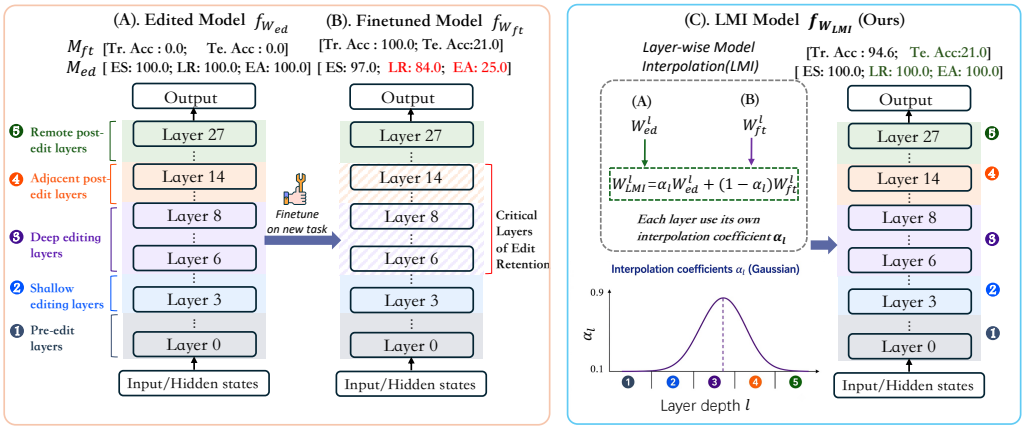


Figure 1: The edit–finetune–merge pipeline to mitigate edit erasing under downstream fine-tuning. We evaluate on GPT-J-6B with MEMIT edits and ZsRE fine-tuning. (**A**): An edited model $f_{W_{ed}}$ with good editing performance M_{ed} but weak performance M_{ft} in a downstream task. (**B**): A fine-tuned model for the downstream task $f_{W_{ft}}$ achieves decent performance, but model edits are largely destroyed. Our empirical study in Sec. 4.3 indicates the mechanism of this erasure and attributes it to disruption in the readout circuitry, i.e., the critical layers for edit retention. (**C**): We partition the Transformer along its depth into five contiguous regions: ❶ pre-edit, ❷ shallow editing, ❸ deep editing, ❹ adjacent post-edit, and ❺ remote post-edit layers, where the readout circuitry corresponds to the deep editing and adjacent post-edit layers. Our proposed solution, LMI, balances edit retention and downstream knowledge acquisition by adaptively merging layers in $f_{W_{ed}}$ and $f_{W_{ft}}$, informed by the observation in Sec. 4.3.

to a high level when the *deep editing layers* and *adjacent post-editing layers* are reverted to their pre-fine-tuning states. This indicates that the knowledge injected by model editing is retained within the editing layers, while fine-tuning predominantly disrupts the readout circuitry that decodes this knowledge into the final prediction. We further verify this hypothesis in a subspace analysis for input-output drifts and also an attention-based interpretability analysis.

Motivated by the layer-wise analysis, we propose a Layer-wise Model Interpolation (LMI) strategy, a variant of model merging techniques (Wortsman et al., 2022), to improve the robustness of model editing in downstream fine-tuning. As illustrated in Fig. 1, LMI constructs a merged model via convex interpolation between the edited and the fine-tuned models, with per-layer coefficients drawn from a Gaussian kernel centered on the readout circuitry, giving stronger restoration near the readout region and weaker restoration elsewhere. LMI demonstrates a balance between edit knowledge retention and downstream knowledge acquisition across a variety of experimental settings.

Our main contributions are threefold. First, we introduce a mechanistic perspective on the robustness of model editing under downstream fine-tuning through the lens of layer-wise information flow, revealing that edited knowledge is retained within editing layers, whereas fine-tuning primarily disrupts the readout circuitry that decodes this knowledge into the final prediction. Second, we validate this hypothesis from multiple perspectives, including layer-wise reversion, subspace analysis of input–output drifts, and interpretability analysis using Grad-ELLM (Huang & Chan, 2026). Third, motivated by these findings, we propose Layer-wise Model Interpolation (LMI). This model-merging-based strategy jointly preserves edited knowledge and acquires downstream knowledge, and demonstrates its effectiveness across diverse models, editing methods, and adaptation settings.

2 Related Work

Knowledge Editing in LLMs. Knowledge Editing (KE) aims to update factual knowledge in LLMs without full retraining. Existing methods include parameter-modifying editors, meta-learning editors, and memory-based editors. Parameter-modifying (locate-then-edit) methods such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023) directly modify localized factual associations, with later variants improving localization, scalability, or interference control (Fang et al., 2025; Li et al., 2024b; Jiang et al., 2025). Meta-learning methods learn auxiliary update rules (Cao et al., 2021; Mitchell et al., 2022a; Tan et al., 2024), while memory-based methods store or route edits through external memories (Mitchell et al., 2022b; Huang et al., 2023; Hartvigsen et al., 2023; Wang et al., 2024a). Our work focuses on parameter-modifying editors, since their edits are committed to model weights and directly exposed to subsequent fine-tuning.

Robustness of Knowledge Editing. Recent work (Wen & Zhang, 2025; Cheng et al., 2025) raises concerns over whether KE methods install durable knowledge beyond narrow benchmark probes, reporting downstream failures and degradation of unrelated capabilities (Hoelscher-Obermaier et al.; Gu et al., 2024; Gupta et al., 2024; Zhang et al., 2024; Yang et al., 2024). Most directly related to our work are two threads of recent findings on edit fragility: Ji et al. (2025); Wen & Zhang (2025) show that edits decay under continued training, with model behavior pulled back toward pre-edit outputs (an "elasticity" effect); and Cheng et al. (2025) shows that this damage can be even more severe when fine-tuning targets layers *outside* the editing range, rather than the editing layers themselves. These studies primarily address *when* edits fail empirically, but ignore *why* the failure occurs and how to mitigate it. In contrast, our work introduces a novel perspective on the mechanistic understanding of model editing and validates the effectiveness of our proposed LMI in a wide range of experimental settings.

Model Merging and Weight Interpolation. Our method interpolates between an edited and a fine-tuned model. Prior work on linear mode connectivity (Frankle et al., 2020; Entezari et al., 2022), model soups(Wortsman et al., 2022), task arithmetic(Ilharco et al., 2023), TIES-Merging(Yadav et al., 2023), and DARE(Yu et al., 2024) show that weight-space operations combine model capabilities under suitable conditions. These methods, however, treat merging as a generic capability-composition tool, typically learning or searching for interpolation coefficients in a task-agnostic manner. In contrast, LMI is directly motivated by our readout-circuitry account of edit retention: rather than treating all layers uniformly, it parameterizes per-layer coefficients with a functional prior centered on the layers where edit decoding occurs. To the best of our knowledge, this work is the first to use model merging to address the brittleness of model editing under downstream fine-tuning, and its effectiveness is validated through extensive experiments.

3 Problem Definition

3.1 Task and Objective

We study the robustness of knowledge edits under post-editing fine-tuning. The pipeline has three stages. First, knowledge editing modifies a model's weights to override specific factual associations without retraining, as illustrated in Fig. 1(A). Concretely, an editing algorithm $\mathcal{E}$ (such as MEMIT (Meng et al., 2023), ROME (Meng et al., 2022), or AlphaEdit (Fang et al., 2025)) takes a base model f_{W_0} and an edit set $\mathcal{D}_{\text{edit}} = \{(p_i, o_{\text{true},i}, o_{\text{new},i})\}$, and produces an edited model f_{W_e} that outputs $o_{\text{new},i}$ in place of $o_{\text{true},i}$ for each prompt p_i. Second, the edited model is fine-tuned on a new dataset $\mathcal{D}_{\text{ft}}$, without overlap with $\mathcal{D}_{\text{edit}}$, yielding $f_{W_{\text{ft}}}$, as shown in Fig. 1(B). After that, we evaluate whether the edits originally injected by $\mathcal{E}$ are still present in $f_{W_{\text{ft}}}$. Throughout, W_0, W_{ed}, and W_{ft} denote the parameters of f_W at the base, post-edit, and post-fine-tuning stages, respectively.

This pipeline reflects a realistic deployment scenario: an editor injects facts into a released model, after which the model continues to be fine-tuned on various downstream tasks. Our goal is to design a recovery map $\mathcal{R}$ that takes the two checkpoints $\{W_{\text{ed}}, W_{\text{ft}}\}$ together with a configuration ϕ and produces a final model f_{W^*} retaining as many edits as possible without sacrificing the capability gained from fine-tuning:

$$\phi^* = \arg\max_{\phi} \underbrace{\mathcal{M}_{\text{ed}}(f_{W^*}; \mathcal{D}_{\text{edit}})}_{\text{edit retention}} \quad \text{where} \quad W^* = \mathcal{R}(W_{\text{ed}}, W_{\text{ft}}, \phi), \quad \text{s.t.} \quad \underbrace{\mathcal{M}_{\text{ft}}(f_{W^*}, \mathcal{D}_{\text{ft}}) \geq \mathcal{M}_{\text{ft}}(f_{W_{\text{ft}}}, \mathcal{D}_{\text{ft}}) - \varepsilon}_{\text{downstream capability preserved}} \tag{1}$$

where $\mathcal{M}_{\text{ft}}$ denotes the fine-tuning performance on $\mathcal{D}_{\text{ft}}$. The objective $\mathcal{M}_{ed}$ measures *edit retention*, while the constraint enforces *downstream capability preservation*: the recovery procedure should not degrade the fine-tuning performance beyond a small tolerance ε. Crucially, the search is over the recovery configuration ϕ, which is a low-dimensional search over the design of $\mathcal{R}$. In LMI, this configuration corresponds to the per-layer interpolation coefficients $\{\alpha_\ell\}$ (Sec. 5). This formulation leaves open the question of the choice of $\mathcal{M}_{ed}$: how should we measure whether an edit has actually been retained?

3.2 Edit Retention Measurement

We measure edit retention with three metrics with varying degrees of strictness.

Edit Success (ES) is the loosest: it only checks whether the edited answer o_{new} is assigned a higher probability than the original answer o_{true}, comparing two specific tokens:

$$\text{ES} = \mathbb{E}_{\mathcal{D}_{\text{edit}}} \mathbb{1}\left[P_{f_W}(o_{\text{new}} \mid p) > P_{f_W}(o_{\text{true}} \mid p)\right]. \tag{2}$$

Loose Retain (LR) is intermediate: it asks whether o_{new} appears anywhere within the first $K = 10$ greedily decoded tokens, requiring the edit to surface in generation but not in any fixed position:

$$\text{LR} = \mathbb{E}_{\mathcal{D}_{\text{edit}}} \mathbb{1}\left[o_{\text{new}} \in \text{Gen}_K(f_W, p)\right]. \tag{3}$$

Edit Accuracy (EA) is the strictest: it requires o_{new} to be the model's actual top-1 prediction:

$$\text{EA} = \mathbb{E}_{\mathcal{D}_{\text{edit}}} \mathbb{1}\left[\arg\max_t P_{f_W}(t \mid p) = o_{\text{new}}\right]. \tag{4}$$

We adopt EA as the primary indicator of edit retention, since it most closely reflects what users observe, and we also report ES and LR as diagnostic auxiliaries.

4 Why Do Edits Degrade Under Fine-Tuning?

In this section, we first show that knowledge edits suffer substantial retention loss after downstream fine-tuning, and then investigate the mechanism underlying this failure from the perspective of layer-wise information readout. All experiments in this section use MEMIT (Meng et al., 2023), the first scalable knowledge-editing method and a widely adopted baseline in the field, to inject 100 edits from CounterFact (CF) (Meng et al., 2022) into GPT-J-6B. We then fine-tune the edited model on 1,000 examples from Zero-Shot Relation Extraction (ZsRE) (Levy et al., 2017) with MLP-only supervised fine-tuning, learning rate (lr) $= 6 \times 10^{-5}$ and 10 epochs, unless otherwise noted. CF and ZsRE are the two standard benchmarks in the knowledge-editing literature: CF provides counterfactual "subject-relation-object" triples, while ZsRE poses the same relations as open-ended QA. We defer full details of the datasets, model, and training protocol to Sec. 6.1.

4.1 Edit Retention Collapses After Fine-Tuning

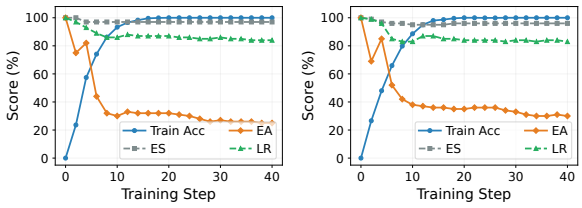
We track edit retention across fine-tuning steps using three metrics: **ES**, **EA**, and **LR**. LR fills the gap between ES and EA: stricter than a probability comparison, but tolerant of the edit being displaced from the first token by the fine-tuning task's output format.

Observation 1. ES masks the true extent of edit collapse. Fig. 2 reveals different levels of collapse across the three retention metrics, consistent for both MEMIT and AlphaEdit. ES holds steady at $\sim$96–97% throughout fine-tuning, under which edits appear to survive almost intact. LR, however, degrades to $\sim$83–84%, while EA collapses from 100% to $\sim$25–30%. The collapse of model editing is largely invisible under ES, but becomes apparent only under stricter probes such as LR and EA. This follows from what the metrics test: ES only requires the new object to be more probable than the original one, which can still hold after fine-tuning, whereas LR and EA require the new object to be the model's actual prediction. Thus, we use EA as the main observational metric in the rest of this section to study the robustness of edits.

4.2 The Edit Signal Survives in Parameter Space

A potential explanation for the collapse is that fine-tuning directly overwrites the edited MLP weights, erasing the injected knowledge (Cheng et al., 2025; Ji et al., 2025). This subsection and the following one show that this intuition is not supported by empirical evidence: the edit signal is largely preserved after fine-tuning; what breaks is the model's ability to *decode* it into the correct output.

To find out whether the edit signal is erased, we analyze how fine-tuning affects the edited MLP weights. Let $\Delta W^l = W^l_{\text{ft}} - W^l_{\text{ed}}$ denote the weight update injected by fine-tuning at the l-th layer of the edited model. Following the key–value memory view of transformer MLPs (Geva et al., 2021; Meng et al., 2023), we treat the MLP input k^l at the subject's last token as the *key* that retrieves the edited fact, since MEMIT writes the edit specifically at this location. For each edited prompt, we extract the key vector before and after fine-tuning, k^l_{ed} and k^l_{ft}, with $\Delta k^l = k^l_{\text{ft}} - k^l_{\text{ed}}$.



(a) MEMIT (b) AlphaEdit

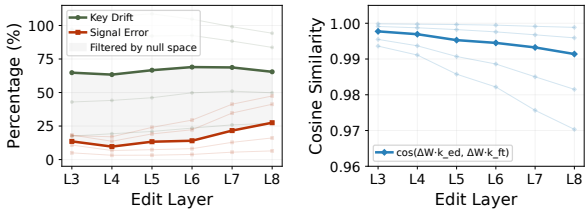
Figure 2: Edit retention under ZsRE downstream fine-tuning for (a) MEMIT and (b) AlphaEdit (100 CF edits, GPT-J-6B, lr $= 6 \times 10^{-5}$). A graded collapse emerges across three retention metrics under both editing methods.

We define two diagnostic quantities:

$$\textit{key drift} = \frac{\|\Delta k^l\|}{\|k^l_{\text{ed}}\|}, \textit{signal error} = \frac{\|\Delta W^l \cdot \Delta k^l\|}{\|\Delta W^l \cdot k^l_{\text{ed}}\|}. \tag{5}$$

Key drift measures how much the input representation to the edited MLP has shifted, and *signal error* measures the fraction of this shift that actually couples with the fine-tuning weight update ΔW^l to perturb the readout. If fine-tuning erased the edit, the signal error should align with key drift well.

Observation 2. The edit signal survives fine-tuning despite substantial key drift. Fig. 3a shows that the two quantities diverge sharply. At lr $= 6 \times 10^{-5}$, key drift averages $\sim$66% across edit layers L3–8, yet signal error averages only $\sim$17% ($\sim$14% at L3–6, rising to $\sim$28% at L8). The gap (shaded region) indicates that Δk^l lies predominantly in the null space of ΔW^l and produces a negligible change in the MLP readout. Furthermore, Fig. 3b shows the cosine similarity $\cos(\Delta W^l \cdot k^l_{\text{ed}}, \Delta W^l \cdot k^l_{\text{ft}})$ remains above 0.99 on every layer, confirming that the edit signal's *direction* is preserved almost exactly, with only a mild attenuation in magnitude. This pattern holds across all learning rates tested: *fine-tuning moves the keys, but ΔW^l is largely blind to this movement.* The edit, as a parameter-space object, is still there. Thus, it is hypothesized that what causes the EA collapse lies not in the edited layers themselves, but in how the model *decodes* this preserved signal into the final prediction. The next subsection localizes this decoding failure.

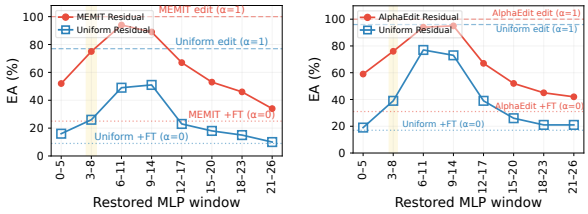


(a) Key drift vs. signal error (b) Signal direction similarity

Figure 3: Edit signal analysis on edit layers L3–8 (MEMIT, 100 edits, GPT-J-6B; bold: lr $= 6\times10^{-5}$, thin: other lr). (a) Key drift vs. signal error; the shaded gap is absorbed by the null space of ΔW^l. (b) $\cos(\Delta W^l k^l_{\text{ed}}, \Delta W^l k^l_{\text{ft}})$ per layer.

4.3 The Readout Path Is Disrupted, Not the Edit Signal

We now localize the layers that carry the failure. The tool is a sliding-window MLP restoration: for each 6-layer window $[l, l+5]$, we replace the post-fine-tuning MLP outputs in that window with their pre-fine-tuning (edited) counterparts at the relevant token positions, leaving all other layers untouched, then measure EA on the same 100 edits. This is a coarse-grained activation patching experiment at the MLP-block level (Vig et al., 2020; Meng et al., 2022). Each window asks the counterfactual "what if only these layers had not been fine-tuned?", and the resulting EA quantifies how much each window causally contributes to the EA collapse.



(a) MEMIT (b) AlphaEdit

Figure 4: EA (%) vs. restored 6-layer MLP window after ZsRE supervised fine-tuning (FT) under two editing methods: (a) MEMIT and (b) AlphaEdit, each compared against a uniform-residual baseline (100 edits). Dashed/dotted lines: only-edited (α=1) and edited+FT (α=0) baselines per setting; yellow shadow: edit layers.

To rule out the possibility that the across-window differences in restoration are an artifact of MEMIT/AlphaEdit's non-uniform residual allocation across the editing layers L3–L8 (Meng et al., 2023; Fang et al., 2025), we additionally include a uniform-residual baseline, in which the same total residual is distributed uniformly across the editing layers. Full details are provided in Appendix A.

Observation 3. Reverting downstream layers recovers edits more effectively than reverting the editing layers themselves. Fig. 4 shows the EA recovery curve across reversion windows for MEMIT and AlphaEdit. The key observation is that reverting the editing layers (L3–L8) is *not* optimal for edit recovery; instead, reverting the layers immediately downstream of the edit region achieves substantially better results. Specifically, the L3–L8 window recovers only 76% EA under both MEMIT and AlphaEdit, whereas reverting L9–L14 alone recovers 89% EA under MEMIT and 95% EA under AlphaEdit. Note that the same downstream peak also appears under the uniform-residual baseline (again peaking at L6–L11 and L9–L14). This rules out the explanation that L6–L14's importance is a side effect of MEMIT's non-uniform allocation of δ across the edit layers.

The same pattern holds on Llama-3-8B and Qwen2.5-7B (Appendix C.1). This empirically grounds the **readout circuitry** introduced in Fig. 1, the L6–L14 band spanning the deep editing (L6–L8) and adjacent post-editing (L9–L14) layers, and motivates our five-region partition in Fig. 1(C). In summary, fine-tuning leaves the edits in edit layers largely intact; it is in this downstream band that fine-tuning disrupts how the signal is decoded.

5 Layer-wise Model Interpolation

The findings from Sec. 4 jointly inform the method design. Observation 2 implies that the lost edit can be recovered by operating directly on the existing checkpoints W_{ed} and W_{ft}, rather than re-editing or retraining. Observation 3 implies that this recovery could be layer-wise, since the *readout circuitry* primarily drives edit recovery.

We accordingly propose a Gaussian-smoothed *Layer-wise Model Interpolation (LMI)*, a linear, layer-wise combination of W_{ed} and W_{ft} that concentrates edit weight on the readout circuitry, i.e., the deep editing and adjacent post-edit layers. For each layer $l \in \{0, \ldots, L-1\}$, LMI produces a merged weight by linearly interpolating between the edited and fine-tuned checkpoints with a layer-wise coefficient $\alpha_l \in [0, 1]$:

$$W^l_{\mathrm{LMI}} = \alpha_l \cdot W^l_{\mathrm{ed}} + (1 - \alpha_l) \cdot W^l_{\mathrm{ft}}. \tag{6}$$

The coefficient α_l is defined by a Gaussian kernel over the layer index:

$$\alpha_l = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot \exp\left(-\frac{(l-\mu)^2}{2\sigma^2}\right), \tag{7}$$

where μ is the centre of the kernel, σ its bandwidth, and $[\alpha_{\min}, \alpha_{\max}]$ the per-layer mixing range. Layers near μ receive $\alpha_l \approx \alpha_{\max}$ and retain primarily edited weights; layers far from μ receive $\alpha_l \approx \alpha_{\min}$ and retain more fine-tuned weights. The optimal μ is expected to lie at the boundary between the deep editing and adjacent post-editing layers, which carry primary responsibility for edit recovery; we will empirically verify this in Fig. 9.

6 Experiments

6.1 Experimental Setup

Our experiments measure how much edited knowledge survives downstream fine-tuning. Tab. 4 summarizes the setup.

LLMs, editing and fine-tuning methods. We conduct our main experiments on two LLMs from different model families, GPT-J-6B (Wang & Komatsuzaki, 2021) and Llama-3-8B (Grattafiori et al., 2024), and further test generalization on Qwen2.5-7B (Qwen et al., 2024). We evaluate two locate-then-edit methods, MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2025). Following the original works, we edit layers 3–8 for GPT-J and layers 4–8 for Llama-3; for Qwen2.5, we edit layers 4–8 following Wang et al. (2024b). For fine-tuning, we consider supervised fine-tuning (SFT) and LoRA (Hu et al., 2022). We first restrict fine-tuning to the MLP sublayers. Since locate-then-edit methods write their updates into MLP weights, MLP-only fine-tuning directly perturbs the edited parameters and thus provides the most targeted test of edit persistence. After validating our findings in this setting, we extend the analysis to attention-only and full-parameter fine-tuning.

Datasets, metrics and LMI hyperparameters. We edit with two standard benchmarks, CounterFact (CF) (Meng et al., 2022) and ZsRE (Levy et al., 2017), at four *edit scales*: 100, 1K, 10K, and the full dataset (≈21K). As defined in Sec. 3.2, we measure knowledge retention with ES, EA, and LR. Additionally, we report paraphrase score (PS), neighborhood score (NS), generation entropy (GE), and reference score (RS) (Meng et al., 2023) to assess generalization and locality of the surviving edits. We also report Train ACC (Tr.) and Test ACC (Te.) to quantify fine-tuning quality. To further assess the model's general capability, we additionally report 5-shot MMLU accuracy (Hendrycks et al., 2021) and zero-shot LAMBADA last-word accuracy (Paperno et al., 2016) for GPT-J. LMI fixes $[\alpha_{\min}, \alpha_{\max}]$=$[0.1, 0.9]$ and sets σ once per model from a coarse sweep; μ is then chosen per setting by maximising $\mathrm{EA} \times \mathrm{Tr.}$, the same rule used for α^* in UMI (App. B.7, E).

Baselines. We compare LMI against two recovery baselines. (i) *Restore Crucial Layers (RCL)*. We identify the *readout circuitry* layers and replace their post-finetune weights with the corresponding pre-finetune (post-edit) weights, leaving all other layers untouched. We select layers 6–14 for GPT-J-6B and layers 7–16 for Llama3-8B, matching the readout regions identified in our mechanistic analysis. (ii) *Uniform Model Interpolation (UMI).* Every layer is interpolated between the post-finetune and

Table 1: Edit accuracy and downstream retention across editing methods, before and after ZsRE fine-tuning. We compare three recovery methods: restore crucial layers (RCL), uniform model interpolation, and layer-wise model interpolation (LMI, ours). Full evaluation. EA: edit accuracy; Tr. and Te. denote ZsRE training and test accuracy. EA/Tr./Te. are reported as percentages (%). α^* is the selected uniform interpolation coefficient; μ^*/σ^* is the selected Gaussian center/width for LMI. For each row, the best value among RCL/Uniform/LMI is **bolded**, and the second-best is underlined.

Model	Method @ Scale	Post-Edit	Post-Fine-tune			RCL			UMI				**LMI (Ours)**			
		EA	EA	Tr.	Te.	EA	Tr.	Te.	EA	Tr.	Te.	α^*	EA	Tr.	Te.	μ^*/σ^*
GPT-J-6B	MEMIT @ 10^2	100.0	25.0	100.0	21.0	97.0	96.2	19.0	**100.0**	**96.6**	**22.0**	0.5	**100.0**	94.6	21.0	6/5
	MEMIT @ 10^3	96.5	8.7	100.0	16.0	62.9	**91.3**	16.0	83.4	86.0	**22.0**	0.6	**90.0**	91.2	17.0	6/5
	MEMIT @ 10^4	95.5	11.6	100.0	17.0	57.4	**95.4**	16.0	81.8	86.1	**18.0**	0.6	**84.1**	91.7	16.0	6/5
	MEMIT @ Full	85.1	5.9	100.0	16.0	34.7	**92.1**	**20.0**	68.9	81.8	**20.0**	0.6	**73.0**	86.6	16.0	4/5
	AlphaEdit @ 10^2	100.0	31.0	100.0	20.0	96.0	95.8	19.0	97.0	**99.0**	**21.0**	0.4	**100.0**	97.7	17.0	6/4
	AlphaEdit @ 10^3	99.8	27.4	100.0	19.0	86.4	97.1	19.0	95.9	97.4	**23.0**	0.5	**96.9**	**98.0**	16.0	6/4
	AlphaEdit @ 10^4	92.7	6.0	100.0	15.0	31.2	90.9	**14.0**	**72.0**	81.8	9.0	0.6	58.7	**95.5**	13.0	6/4
	AlphaEdit @ Full	57.6	0.4	100.0	17.0	1.5	46.7	9.0	**41.1**	72.2	**12.0**	0.6	30.2	**78.9**	11.0	5/4
Llama3-8B	MEMIT @ 10^2	100.0	2.0	100.0	27.0	10.0	98.6	25.0	58.0	**99.0**	**43.0**	0.6	**83.0**	92.2	37.0	10/14
	MEMIT @ 10^3	93.3	3.1	100.0	22.0	9.0	98.7	27.0	54.3	**98.8**	39.0	0.6	**72.9**	90.4	**41.0**	10/14
	MEMIT @ 10^4	83.9	2.0	100.0	14.0	6.8	**98.2**	25.0	**56.7**	82.6	**33.0**	0.8	53.9	93.7	32.0	10/14
	MEMIT @ Full	79.9	1.5	100.0	15.0	4.6	**98.5**	21.0	**50.7**	83.1	**43.0**	0.8	50.5	94.6	37.0	7/14
	AlphaEdit @ 10^2	100.0	2.0	100.0	21.0	31.0	**98.9**	28.0	79.0	98.7	**40.0**	0.6	**95.0**	90.4	34.0	9/14
	AlphaEdit @ 10^3	99.9	4.0	100.0	17.0	16.4	**99.1**	35.0	65.4	98.8	**42.0**	0.6	**89.3**	94.2	34.0	9/14
	AlphaEdit @ 10^4	92.2	2.0	100.0	14.0	5.1	**99.8**	17.0	**68.6**	82.2	**34.0**	0.8	66.8	95.3	20.0	9/14
	AlphaEdit @ Full	85.4	2.0	100.0	17.0	4.9	98.2	18.0	33.1	**99.6**	**26.0**	0.6	**59.5**	95.2	20.0	7/14
Qwen 2.5-7B	MEMIT @ Full	81.6	1.8	100.0	13.0	4.9	**99.1**	12.0	48.4	81.7	**25.0**	0.7	**48.8**	88.3	17.0	13/10
	AlphaEdit @ Full	52.3	1.2	100.0	9.0	2.4	**99.5**	9.0	**49.6**	51.3	**12.0**	0.8	34.6	86.3	**12.0**	14/10

post-edit checkpoints with the same coefficient,

$$W_l^{(\alpha)} = (1-\alpha)\, W_l^{\mathrm{FT}} + \alpha\, W_l^{\mathrm{edit}}, \quad \forall\, l, \tag{8}$$

and the single coefficient $\alpha \in [0,1]$ is swept to find the sweet spot α^*. Note that both baselines can be viewed as degenerate versions of LMI: UMI uses a constant α_l across all layers (recovered as $\sigma \to \infty$ with $\alpha_{\min} = \alpha_{\max}$), while RCL uses a binary $\alpha_l \in \{0,1\}$ profile concentrated on the readout block.

We also include two widely used merging baselines that are agnostic to where the edits are stored. (iii) *Task Arithmetic (TA)* (Ilharco et al., 2023). With the edit task vector $\tau_{\mathrm{edit}} = W_{\mathrm{ed}} - W_0$ and the fine-tuning task vector $\tau_{\mathrm{ft}} = W_{\mathrm{ft}} - W_0$, both defined with respect to the pre-edit base model W_0, TA returns $W_0 + \lambda\,(\tau_{\mathrm{edit}} + \tau_{\mathrm{ft}})$. (iv) *TIES-Merging(TIES)* (Yadav et al., 2023) keeps the top-k fraction of entries of each task vector by magnitude, elects a sign per parameter, and averages only the entries that agree with the elected sign before adding the result to W_0. To make a fair comparison, we sweep $\lambda \in \{0.3, 0.5, \ldots, 1.3\}$ and $k \in \{0.1, 0.2, 0.3\}$ and report the best EA for every checkpoint.

6.2 Results Analysis

Fine-tuning substantially erases edits. Tab. 1 shows that, even a brief round of ZsRE SFT causes EA on CF to drop sharply. On GPT-J-6B with MEMIT, EA falls from 100.0% to 25.0% at the 10^2 scale and from 85.1% to 5.9% at the full scale; on Llama3-8B the collapse is even more severe, with EA dropping below 5% in most rows. AlphaEdit shows the same pattern. This damage is consistent across models, editors, and edit scales, confirming that knowledge edits are extremely fragile to subsequent fine-tuning.

LMI recovers most of the lost edits and outperforms all baselines. LMI restores CF edit accuracy far beyond the post-finetune checkpoint, e.g., from 5.9% to 73.0% (GPT-J, MEMIT @ Full) and from 2.0% to 83.0% (Llama3, MEMIT @ 10^2). Across the 18 settings in Tab. 1, LMI achieves the highest EA in 12 rows; the localized RCL baseline never wins, and UMI wins the remaining six rows, all at the 10^4 or Full scale, where the edited endpoint itself is weaker. Model-merging baselines do not close the gap either (Tab. 2): under the MLP-only SFT considered so far (MLP-only column in Tab. 2), LMI exceeds TA by 15–39 EA points on three model-editor pairs, and by 1 point on GPT-J with AlphaEdit. TIES never exceeds 16% EA because magnitude trimming discards most of the small, localized edit update. Effective recovery therefore requires both interpolation and a layer-wise allocation of the interpolation weight.

Table 2: EA (%) of model-merging baselines and LMI on Full-scale CF edits followed by ZsRE SFT under three parameter scopes.

Model	Editor	MLP-only			Attn-only			Full-param.		
		TA	TIES	LMI	TA	TIES	LMI	TA	TIES	LMI
GPT-J	MEMIT	57.9	16.0	**73.0**	68.2	14.9	**78.4**	24.4	4.0	**36.2**
	AlphaEdit	29.2	6.9	**30.2**	46.6	7.3	**47.3**	**11.8**	2.4	10.2
Llama-3	MEMIT	13.1	3.3	**50.5**	5.3	1.2	**58.7**	1.7	0.7	**15.0**
	AlphaEdit	20.2	3.6	**59.5**	7.5	1.8	**63.2**	1.7	0.5	**13.8**

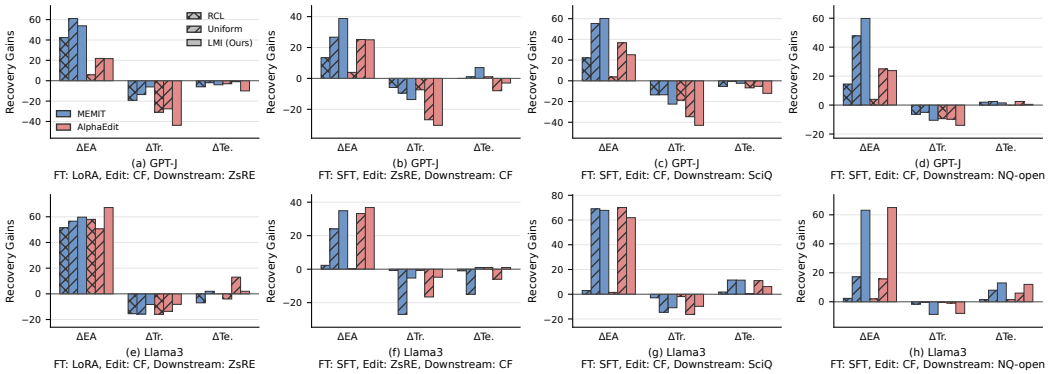


Figure 5: Recovery gains over post-finetune checkpoints across different fine-tuning methods, editing tasks, and downstream fine-tuning tasks, all these experiments are conducted on @full scale editing.

Beyond MLP-only fine-tuning. Tab. 2 also relaxes the MLP-only fine-tuning considered so far to attention-only and full-parameter SFT, so that fine-tuning is no longer confined to the edited weights. LMI remains the strongest method, attaining the highest EA in 7 of the 8 attention-only and full-parameter settings. Recovery under attention-only SFT (47.3–78.4%) is even higher than under MLP-only SFT, whereas full-parameter SFT, which updates every tensor carrying the edit, is the hardest regime (10.2–36.2%).

Downstream performance is preserved, and often improved. LMI's ZsRE test accuracy (Te.) is comparable to or higher than the post-finetune checkpoint in nearly every row of Tab. 1. On Llama3-8B with MEMIT @ 10^3, Te. rises from 22.0% post-finetune to 41.0% after LMI; with AlphaEdit @ 10^3 it rises from 17.0% to 34.0%. LMI does sacrifice some ZsRE training accuracy (Tr.), e.g., from 100.0% post-finetune to 91.2% on GPT-J with MEMIT @ 10^3, and to 90.4% on Llama3 with AlphaEdit @ 10^2, but this drop is small in most settings and is dwarfed by the corresponding gains in EA, which are tens of points in absolute terms. Fig. 6 makes this trade-off explicit by sweeping the Gaussian center μ: the configuration selected in Tab. 1 is non-dominated in EA and Tr, and increasing μ from 4 to 8 on GPT-J with MEMIT raises Tr. from 87% to 91% at the cost of 10 EA points, whereas the trade-off for AlphaEdit is much steeper.

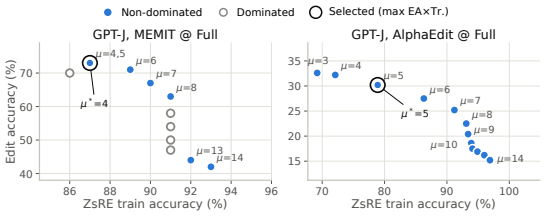


Figure 6: EA against ZsRE training accuracy for LMI on GPT-J (Full-scale CF edits), sweeping μ at fixed σ. Filled points are non-dominated within the sweep; the ring marks the configuration in Tab. 1.

Validity under different settings. We further test whether LMI generalizes beyond a single edit-finetune configuration. Fig. 5 compares LMI with UMI and RCL across four transfer settings: LoRA fine-tuning, editing in the ZsRE space, and two other downstream tasks, SciQ (Welbl et al., 2017) and NQ-open (Kwiatkowski et al., 2019; Lee et al., 2019). Each panel reports the gain over the post-finetune checkpoint in edit accuracy (ΔEA) and in downstream accuracy (ΔTr., ΔTe.). Across the 16 settings (four transfer settings, two models, two editors), LMI attains the largest ΔEA in 9 and is within 1.2 points of the best in 4 more. Its advantage is largest on Llama-3 (e.g., +63–65 EA points under NQ-open against +16–17 for UMI), where it also loses less training accuracy than UMI; on GPT-J the two methods are closer, with UMI ahead in a few settings. RCL recovers the least EA in almost every setting. The pattern therefore holds when the fine-tuning recipe, the edit dataset, or the downstream task is changed.

Generalization to Qwen2.5. We also apply the pipeline to Qwen2.5-7B with Full-scale MEMIT and AlphaEdit edits and ZsRE fine-tuning (Tab. 1). Fine-tuning again reduces EA to below 2% for both editors, and RCL recovers below 5%. LMI restores 48.8% EA (MEMIT) and 34.6% (AlphaEdit) while retaining 86–88% training accuracy, a better trade-off between edit retention and downstream accuracy on both editors than UMI, which reaches comparable or higher EA only at 81.7% and 51.3% training accuracy. Qwen2.5 shows that the readout band is a property of the model rather than of the edit layers: although editors write into layers 4–8, exactly as on Llama-3, the selected profile center is μ^*=13–14 for both editors, well downstream of the edited layers.

Recovery extends beyond edit accuracy. Beyond edit retention, we examine whether LMI restores the broader behavioral profile of the edited model. Fig. 7 reports paraphrase score (PS), neighborhood

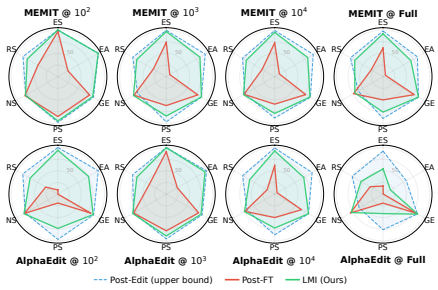


Figure 7: Behavioral profile on CounterFact (ES, EA, PS, NS, GE, RS) for GPT-J at four edit scales. LMI (green) tracks the post-edit profile (blue) on most axes, while post-finetune (red) contracts substantially.

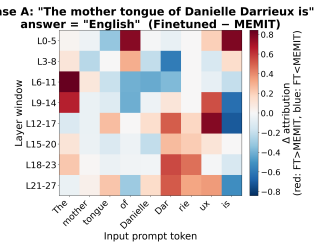


Figure 8: Layer-wise attribution difference for the answer token between the fine-tuned and MEMIT-edited models. Rows are sliding layer windows; columns are prompt tokens. Red/blue indicates that fine-tuning attributes more/less than MEMIT to a token.

score (NS), generation entropy (GE), and reference score (RS), alongside ES and EA. For both MEMIT and AlphaEdit on GPT-J, fine-tuning distorts this profile across all edit scales, worsening not only edit retention but also paraphrase generalization and reference-based scores. LMI closely tracks the post-edit profile on most axes, indicating that recovery is not confined to edit success but extends to generalization, locality, and generation quality.

We further measure general capability after Full-scale MEMIT edits followed by fine-tuning on 1,000 ZsRE facts (Tab. 3), using 5-shot MMLU (Hendrycks et al., 2021) on Llama-3-8B and LAMBADA (Paperno et al., 2016) on GPT-J. On Llama-3, fine-tuning lowers MMLU from 58.3% after editing to 38.5%; UMI restores it to 53.5% and LMI to 58.2%, within 0.1 points of the edited model. On GPT-J, fine-tuning lowers LAMBADA accuracy from 51.6% to 45.0%; UMI and LMI reach 59.5% and 58.2%, both above the two endpoints. In neither case does a merged model fall below the fine-tuned endpoint, so merging introduces no additional capability loss, and the remaining gap to the unedited base models (65.3% MMLU, 67.8% LAMBADA) stems from Full-scale editing itself.

Table 3: General capability (%) after Full-scale MEMIT edits and ZsRE SFT: 5-shot MMLU for Llama-3, LAMBADA last-word accuracy for GPT-J.

Checkpoint	Llama-3 MMLU	GPT-J LAMBADA
Base	65.3	67.8
Post-Edit	58.3	51.6
Post-Fine-tune	38.5	45.0
UMI	53.5	**59.5**
LMI (Ours)	**58.2**	58.2

Case study. Fig. 8 presents the attribution difference across MLP layers in GPT-J for the answer token between the supervised fine-tuned and the MEMIT-edited model, using Grad-ELLM (Huang & Chan, 2026). The vertical axis indexes the layer windows and the horizontal axis the prompt tokens, with red (blue) indicating that fine-tuning attributes more (less) to a given token than MEMIT. Within the edited layers (L3-8), attribution remains largely unchanged by fine-tuning, indicating that the stored edit signal is preserved. From the readout layers onward, starting at window L6-11, the prompt evidence on which the answer token relies shifts substantially, indicating that fine-tuning rewires the model's internal retrieval pathway for the edited fact at the readout layers.

Ablation study. Fig. 9 ablates μ, which sets the layers that receive the highest edit weight, with σ=5 fixed. The recovery of edit (EA) is strongly unimodal in μ, peaking at μ=6, whereas fine-tune retention (Tr.) stays essentially flat (∼91%) across the entire sweep. The optimum therefore reflects *where* edits are read out rather than a trade-off against forgetting.

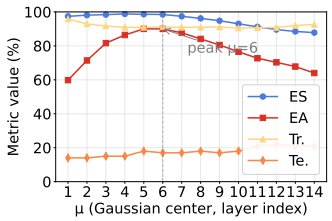


Figure 9: Ablation on the Gaussian center μ (σ=5 fixed): GPT-J, 1K CF edits with MEMIT, fine-tuned on 1K ZsRE.

7 Conclusion

This paper revisits the brittleness of model editing under downstream fine-tuning from a mechanistic perspective. Through layer-wise reversion, supported by null-space subspace analysis and Grad-ELLM attribution, we show that edited knowledge is preserved in the early editing layers, while fine-tuning primarily disrupts the later readout layers that decode it into the final prediction. Motivated by this finding, we propose Layer-wise Model Interpolation, which merges the pre- and post-fine-tuning weights of the critical readout layers under a layer-aware Gaussian kernel. Across four datasets, three language models, and two editing methods, LMI consistently achieves the best balance between edit retention and downstream utility.

AI USE STATEMENT

In this work, we used generative AI tools (large language model assistants) for writing assistance and for code generation. For writing, the tools were used to polish grammar and wording, shorten paragraphs, and check consistency between the text and the reported numbers; the research questions, method, experimental design, and the claims made in the paper were conceived and written by the authors. For code, the tools assisted in writing experiment launch scripts, result-collection and table-generation scripts, and plotting scripts; the core editing, fine-tuning, and interpolation implementations were written by the authors. We have not used generative AI tools for research ideation, for generating or fabricating experimental data, or for literature search and citation generation. The remaining disclosure categories of the ICLR 2027 AI Policy for Authors are not applicable to this work.

We have reviewed all AI-assisted work. All AI-generated or AI-edited text was read and revised by the authors, and every numerical claim was checked against the underlying result files. AI-assisted code was reviewed by at least two authors, tested on small inputs before use, and, where it produced tables or figures, its output was cross-checked against independently logged results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All models, editing methods, and datasets used in this work are publicly available, and every step of the pipeline is deterministic given the released configurations. Appendix B documents the experimental setup in full: the language models and checkpoints (App. B.1), the editing hyperparameters for MEMIT and AlphaEdit (App. B.2), the fine-tuning recipes for SFT, LoRA, attention-only and full-parameter training (App. B.3), the datasets and train/test splits (App. B.4), the evaluation protocol and metric definitions (App. B.5), the exact merging procedures for RCL, UMI, LMI, Task Arithmetic and TIES-Merging (App. B.6), and the compute environment (App. B.8). The hyperparameter selection procedure for α and (μ, σ) is specified in App. B.7 and App. E, with the complete sweep results in App. C. The Grad-ELLM attribution and the SVD null-space analysis are described in App. D and App. F. Our source code, including the editing and fine-tuning scripts, the layer-wise interpolation implementation, the evaluation code, and the scripts that regenerate every table and figure from the logged results, is provided as anonymised supplementary material and will be released publicly upon acceptance.

REFERENCES

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Siyuan Cheng et al. Can fine-tuning erase your edits? on the fragile coexistence of knowledge editing and adaptation. *arXiv preprint arXiv:2511.05852*, 2025.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

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

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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. *arXiv preprint arXiv:2401.04700*, 2024.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics: (NAACL) 2024*, 2024.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Mo Li, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics(ACL Findings)*.

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 (ICLR)*, 2022.

Xin Huang and Antoni B. Chan. Grad-ellm: Gradient-based explanations for decoder-only llms, 2026. URL `https://arxiv.org/abs/2601.03089`.

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 (ICLR)*, 2023.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

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

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AnyEdit: Edit any knowledge encoded in language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

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.

Hanyu Lai, Xiao Liu, Junjie Gao, Jiale Cheng, Zehan Qi, Yifan Xu, Shuntian Yao, Dan Zhang, Jinhua Du, Zhenyu Hou, Xin Lv, Minlie Huang, Yuxiao Dong, and Jie Tang. A survey of post-training scaling in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics(ACL)*, 2025.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

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

Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. In *Findings of the association for computational linguistics (EMNLP Findings)*, 2024a.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. PMET: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024b.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.

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

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 (ICLR)*, 2023.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2022b.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

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. *arXiv preprint arXiv:2412.15115*, 2024.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems (NeurIPS)*, 2017.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2020.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 billion parameter autoregressive language model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021.

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

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024b.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (W-NUT)*, 2017.

Fufang Wen and Shichang Zhang. On the retention of edited knowledge in fine-tuned language models. *arXiv preprint arXiv:2507.14198*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Qi Yang, Jiawei Wang, Yifan Yang, Jian Liu, and Jingyuan Wang. Should we really edit language models? on the evaluation of edited language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super Mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

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. *arXiv preprint arXiv:2401.01286*, 2024.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Jun Zhao, Yongzhuo Yang, Xiang Hu, Jingqi Tong, Yi Lu, Wei Wu, Tao Gui, Qi Zhang, and Xuanjing Huang. Understanding parametric and contextual knowledge reconciliation within large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

- **Limitations**
- **A. Uniform Residual Baseline**
- **B. Experimental Setup**
  - B.1 Language Models
  - B.2 Editing Methods
  - B.3 Downstream Fine-tuning
  - B.4 Datasets and Train/Test Splits
  - B.5 Evaluation Protocol
  - B.6 Recovery Procedures
  - B.7 Hyperparameter Selection
  - B.8 Compute Environment
- **C. Detailed Experiments Results**
  - C.1 Sliding-window on Llama-3
  - C.2 Uniform Interpolation
  - C.3 Layer-wise Model Interpolation
- **D. Grad-ELLM Attribution: Method and Additional Case Studies**
  - D.1 Attribution Formula
  - D.2 Sliding-Window Aggregation and Difference Protocol
  - D.3 Additional Case Studies
- **E. Hyperparameter Search Procedure**
  - E.1 UMI: α Sweep
  - E.2 LMI Stage 1: Coarse 2D (μ, σ) Sweep
  - E.3 LMI Stage 2: Fine 1D μ Sweep at Fixed σ
  - E.4 Why a Product Objective?
- **F. SVD Verification of the Null-Space Finding**
  - F.1 Setup
  - F.2 Findings

LIMITATIONS

Scope of editors and edit types. Our analysis covers two locate-then-edit editors (MEMIT, AlphaEdit) that write into MLP sublayers of decoder-only Transformers. We do not study memory-based editors (e.g., GRACE, WISE), meta-learning editors (e.g., MEND, MALMEN), or fine-tuning-based editors (e.g., FT-L, FT-M). The last two families disperse the edit over the same parameters that downstream fine-tuning perturbs and reach lower edit success at scale, so the retention question is less well posed for them; whether the readout-circuitry account extends to them is open. All edits are QA-style factual associations, and we consider a single edit-then-finetune round; behavioural edits and multi-round pipelines (edit-finetune-edit, continual editing) are left for future work.

Fine-tuning regimes. We evaluate MLP-only SFT and LoRA as the main regimes and extend to attention-only and full-parameter SFT. Recovery under full-parameter SFT is only partial (10.2–36.2% EA against 47–78% under attention-only SFT). All fine-tuning corpora have at most $\sim$12K examples and at most 10 epochs; continued pretraining on large corpora and instruction tuning are not covered.

Model coverage and extreme-scale editing. Experiments span three decoder-only models in the 6–8B range (GPT-J-6B, Llama-3-8B, Qwen2.5-7B). Transfer to substantially larger models, encoder–decoder architectures, or mixture-of-experts models is untested. At the Full scale ($\sim$21K edits), mass editing itself already degrades general capability; LMI recovers what fine-tuning destroyed, but cannot restore capability that the edited endpoint no longer contains, so its benefit is bounded by the quality of the edited model.

Profile selection. The Gaussian profile (μ, σ) is chosen per model and editor by a small sweep, and the best profile is architecture-dependent: GPT-J prefers a narrow early band, whereas Llama-3 and Qwen2.5 require a broad, later-centred profile. The sweep is scored on the evaluated edits themselves (App. B.7); a principled way to set the profile without any sweep, or to select it on a held-out subset of edits, remains open.

A Uniform Residual Baseline

When applying multi-layer edits, MEMIT and AlphaEdit distribute the target residual increment δ non-uniformly across the edit layers L3–8: later layers carry a larger share, induced by the recursive formula that splits the remaining residual among the remaining layers. This non-uniform allocation could itself introduce a structural bias within the edit layers, making certain downstream layers *appear* disproportionately important when in fact the effect merely reflects the allocation scheme rather than any property of the readout. To rule out this explanation, we introduce a **uniform-residual baseline**: holding the total δ fixed, we instead allocate it *uniformly* across L3–8 ($\delta/6$ per layer), keeping the rest of the MEMIT/AlphaEdit pipeline unchanged, and then run the same sliding-window restoration experiment on top of this variant. If the downstream-window peak persists under uniform allocation, we can rule out the explanation that the phenomenon is an artifact of how the edit signal is distributed within the edit layers.

B Experimental Setup

Table 4: Experimental setup. Full specifications are given in §B.

Component	Configuration
Models and editing	
Models	GPT-J-6B (Wang & Komatsuzaki, 2021) , Llama-3-8B (Grattafiori et al., 2024) , Qwen2.5-7B (Qwen et al., 2024)
Editing Methods	MEMIT (Meng et al., 2023), AlphaEdit (Fang et al., 2025)
Edited Weights	MLP down-projection; layers 3–8 (GPT-J), layers 4–8 (Llama-3, Qwen2.5)
Edit Datasets	CounterFact (Meng et al., 2022) (main), ZsRE (Levy et al., 2017) (transfer setting)
Edit Scales	100, 1K, 10K, Full (21,919 for CF; 19,009 for ZsRE)
Edit Metrics	ES, LR, EA, evaluated on all edited facts at each scale
Downstream fine-tuning	
Fine-tune Methods	SFT and LoRA (Hu et al., 2022) (r=16, α_{LoRA}=32, merged after training)
Parameter Scopes	(i) MLP only: the MLP sublayers that hold the edits (main setting; all models, scales, SFT and LoRA). (ii) Attention only: query, key, value and output projections. (iii) Full-parameter: all weights.
Fine-tune Datasets	ZsRE subset (1,000), SciQ (Welbl et al., 2017) (11,679), NQ-open (Kwiatkowski et al., 2019; Lee et al., 2019) (1,000), CF subset (1,000; for ZsRE edits)
Optimization	AdamW, `bfloat16`; learning rate 6×10^{-5} (GPT-J SFT), 1×10^{-4} (Llama-3 and Qwen2.5 SFT), 8×10^{-4} (LoRA); 10 epochs with effective batch size 256 (NQ-open and scopes (ii) and (iii): 1, 3, and 10 epochs, effective batch size 64)
Fine-tune Metrics	Train ACC (training examples), Test ACC (held-out examples: 100 for ZsRE, 200 for NQ-open); greedy decoding, exact match
Recovery and evaluation	
Recovery Methods	RCL, UMI (α), LMI (Gaussian profile, μ, σ); selected by $\max \text{EA}\times\text{Tr}$.
General Capability	MMLU (Hendrycks et al., 2021) (Llama-3, 5-shot), LAMBADA (Paperno et al., 2016) (GPT-J)
Hardware	Single NVIDIA H200 GPU per run

This appendix gives the full specification used to produce every number reported in the main text. We list the language models (§B.1), the two locate-then-edit editors (§B.2), the two downstream fine-tuning regimes (§B.3), the datasets and the train/eval splits used to compute every metric (§B.4), the evaluation protocol (§B.5), the three recovery procedures (§B.6), the hyperparameter selection rule (§B.7), and the compute environment (§B.8). All configuration values below are exactly the ones used in our released YAML/scripts.

B.1 LANGUAGE MODELS

We use three open-weight decoder-only LLMs, all at the pretrained (non-instruction-tuned) checkpoint. GPT-J-6B(Wang & Komatsuzaki, 2021) has L=28 Transformer blocks, hidden size 4096, 16 attention heads with rotary embeddings on 64 dimensions, a 50,400-token vocabulary, and a standard two-layer GELU FFN (`fc_in`→`fc_out`) with intermediate size 16384; its attention and MLP sublayers are applied in parallel within each block, and edits write into `transformer.h.{l}.mlp.fc_out`. Llama-3-8B (Grattafiori et al., 2024) has L=32 blocks, hidden size 4096, 32 query heads with grouped-query attention (8 key/value heads), a 128,256-token vocabulary, and a gated SwiGLU MLP with intermediate size 14336, whose down-projection `model.layers.{l}.mlp.down_proj` plays the role of the second linear layer that edits write into. Qwen2.5-7B (Qwen et al., 2024) has L=28 blocks, hidden size 3584, 28 query heads with grouped-query attention (4 key/value heads), a 152,064-token vocabulary, and the same gated SwiGLU MLP structure with intermediate size 18944; edits again target `model.layers.{l}.mlp.down_proj`. Following the original works, the edit layers are $\{3, \ldots, 8\}$ for GPT-J and $\{4, \ldots, 8\}$ for Llama-3 and Qwen2.5.

B.2 EDITING METHODS

We evaluate two locate-then-edit editors that write into the MLP *output* matrix at the subject's last-token position.

MEMIT (Meng et al., 2023). Edit layers are $\{3, 4, 5, 6, 7, 8\}$ for GPT-J and $\{4, 5, 6, 7, 8\}$ for Llama-3 and Qwen2.5, matching the original works and the EasyEdit defaults (Wang et al., 2024b). The optimizer used to find the per-edit target value v^* runs for 25 gradient steps with learning rate 0.5 on GPT-J and Qwen2.5 and 0.1 on Llama-3, weight decay 0.5, a KL anchor on the unedited distribution with weight 0.0625, and a clamp-norm factor of 0.75. The second-moment matrix used by the closed-form key-projection step is estimated from 10^5 Wikipedia samples and stored in `float32`; the mixing weight λ for this matrix is 15,000 for all three models. Edits rewrite `transformer.h.{l}.mlp.fc_out` on GPT-J and `model.layers.{l}.mlp.down_proj` on Llama-3 and Qwen2.5, with the loss anchored at the final block (layer 27 for GPT-J and Qwen2.5, layer 31 for Llama-3).

AlphaEdit (Fang et al., 2025). AlphaEdit shares the same edit layers, KL weight, clamp factor, v^* optimizer, second-moment matrix, and rewrite modules as MEMIT. It adds a null-space projector P pre-computed once per model from the second-moment statistics with a null-space threshold of 2×10^{-2}, plus an explicit Tikhonov term L_2=10 on GPT-J and Qwen2.5 and L_2=1 on Llama-3. For Qwen2.5 we depart from the EasyEdit default configuration (clamp-norm factor 4, weight decay 10^{-3}, L_2=1): its residual stream is roughly an order of magnitude larger in norm than Llama-3's, and under the default clamp the per-batch updates at the Full scale grow to several times the norm of the original weights, so editing itself fails. We therefore use the GPT-J values (0.75 / 0.5 / 10) for both editors on Qwen2.5, which keeps the update-to-weight norm ratio in the same range as on GPT-J and Llama-3.

Edit scales. We edit at four scales: 10^2, 10^3, 10^4, and the full CF split ($21k$ edits). Qwen2.5 is edited at the Full scale only. At each scale we pre-cache an edited checkpoint $f_{W_{\text{ed}}}$ that is then re-used by all downstream-fine-tuning and recovery experiments, so $f_{W_{\text{ed}}}$ is identical across rows that share a model×method×scale cell.

B.3 DOWNSTREAM FINE-TUNING

Unless stated otherwise, the trainable parameters are restricted to the MLP sublayers (the two FFN matrices on GPT-J, and `gate_proj`/`up_proj`/`down_proj` on Llama-3 and Qwen2.5). This is the same parameter set that MEMIT/AlphaEdit modify, so fine-tuning directly perturbs the edited weights; attention, embeddings, layernorms and the LM head are frozen. This MLP-only scope is used for every row of Tab. 1 and Fig. 5, for both SFT and LoRA. The two wider scopes of Tab. 2 are described at the end of this section.

Supervised fine-tuning (SFT). We use AdamW in `bfloat16`, per-device batch size 32 with gradient accumulation 8 (effective batch 256), maximum sequence length 256, gradient clipping at 1.0, and 10 epochs. The learning rates are 6×10^{-5} on GPT-J and 1×10^{-4} on Llama-3, held constant throughout training; we observed in pilot sweeps that further tuning the learning rate has a small effect on the qualitative collapse pattern reported in §4.

LoRA fine-tuning. We attach LoRA adapters (Hu et al., 2022) with rank r=16, scaling α_{LoRA}=32, and dropout 0.05 on the same MLP target modules. LoRA fine-tuning uses batch size 32, 10 epochs, and a learning rate 8×10^{-4} for both models, with the rest of the optimization configuration matching SFT. To form a single weight-space checkpoint that the recovery procedures of §B.6 can interpolate, we merge the LoRA adapters back into the base MLP matrices before evaluation.

Attention-only and full-parameter SFT. For Tab. 2 we relax the parameter scope on the Full-scale checkpoints. *Attention-only* trains the query, key, value and output projections of every block (`q_proj/k_proj/v_proj/out_proj` on GPT-J, `q_proj/k_proj/v_proj/o_proj` on Llama-3) and freezes the MLPs, so the edited weights are never touched directly; *full-parameter* trains all weights. Both use the SFT optimiser and learning rates above with per-device batch size 16 and gradient accumulation 4 (effective batch 64), checkpointed at epochs 1, 3 and 10; the epoch-10 checkpoints are the ones reported. Recovery is applied with the same (μ, σ) profile as the MLP-only setting (Tab. 5), without re-tuning.

B.4 Datasets and Train/Test Splits

Edit datasets. COUNTERFACT (CF) (Meng et al., 2022) contains $\sim$ 21,919 counterfactual subject-relation-object triples; at scale S we edit the first S records and report edit-retention metrics on exactly those S records (no edit is ever evaluated on records it was not applied to). ZsRE (Levy et al., 2017) contains the same relations posed as open-ended QA; we use it both as an edit dataset (for the ZsRE-edit→CF-FT transfer experiment) and as a downstream fine-tuning dataset.

Downstream fine-tuning datasets. For the main ZsRE-SFT/LoRA experiments, the training set is the first 1,000 ZsRE examples and the held-out test set is a disjoint 100-example slice; this matches the protocol used in §4 of the main text. For the SciQ experiments (Welbl et al., 2017), we use the full `allenai/sciq` training split (11,679 examples; scripts pass `train_size=12000` and SciQ caps it to the available size), with 1,000 examples held out as a validation slice and the standard 1,000-example SciQ validation split for test reporting. For the NQ-open experiments (Kwiatkowski et al., 2019; Lee et al., 2019), we use the first 1,000 questions of the `nq_open` training split for fine-tuning and the first 200 questions of its validation split as the held-out test set. For the ZsRE-edit→CF-FT transfer experiment, we fine-tune on the first 1,000 CF records (using `target_true` as the supervision target so that fine-tuning pushes the model toward the *un-edited* answer).

Disjointness. The fine-tuning training sets never overlap with the edit set: CF and ZsRE share relations but use disjoint records; SciQ and NQ-open are from an entirely different domain.

B.5 Evaluation Protocol

Edit-retention metrics. For every reported number we evaluate edit retention on *all* edited cases at that scale ($S \in \{10^2, 10^3, 10^4, \approx 2.1\times10^4\}$); we never sub-sample the edit set. ES, LR, EA are defined exactly as in §3.2 of the main text. LR uses greedy decoding for K=10 tokens; EA is the strict top-1 argmax probe; ES is the two-token probability comparison.

Downstream metrics. Train accuracy (Tr.) is computed on the same 1,000 records used for fine-tuning, and test accuracy (Te.) on the held-out slice described in §B.4. For SFT/LoRA on ZsRE/SciQ, we use greedy decoding with a maximum of 10 new tokens and exact-match scoring against the gold answer. The eval batch size is 1 on SFT (to match the edit-eval pipeline) and 64 on LoRA.

Generalization metrics (Fig. 7). Paraphrase score (PS), neighborhood score (NS), generation entropy (GE), and reference score (RS) follow the definitions of Meng et al. (2023) and are computed on the same edited records as ES/EA, using the official MEMIT evaluation harness.

B.6 Recovery Procedures

All three recovery procedures take the two cached checkpoints $\{W_{\text{ed}}, W_{\text{ft}}\}$ and produce a single recovered checkpoint W^* via per-parameter weight interpolation. No additional training or re-editing is performed.

Restore Crucial Layers (RCL). We overwrite all parameters of the blocks in the band with the post-edit MLP weights inside a fixed layer band and leave every other layer untouched. The band is layers $[6, 14]$ on GPT-J and $[7, 16]$ on Llama-3, matching the readout regions identified in §4.3. RCL is a binary $\alpha_\ell \in \{0, 1\}$ profile.

Uniform Model Interpolation (UMI). A single coefficient $\alpha \in [0, 1]$ is applied to all parameters: $W_\ell^{(\alpha)} = \alpha W_\ell^{\text{ed}} + (1-\alpha) W_\ell^{\text{ft}}$. We sweep $\alpha \in \{0.1, 0.2, \ldots, 0.9\}$ together with the endpoints $\{0.0, 1.0\}$ and report the sweet-spot α^* chosen by the rule in §B.7.

Layer-wise Model Interpolation (LMI, ours). The per-layer coefficient is a Gaussian kernel

$$\alpha_\ell \;=\; \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \, \exp\!\Big(-\tfrac{(\ell-\mu)^2}{2\sigma^2}\Big), \tag{9}$$

with $\alpha_{\min}$=0.1, $\alpha_{\max}$=0.9 throughout. The center μ is swept over integer layer indices and the bandwidth σ is taken from a small per-setting grid (Tab. 5). We did *not* sweep μ and σ jointly: σ is fixed up-front based on whether the setting concentrates the readout circuitry tightly (GPT-J, ZsRE-SFT) or broadly (Llama-3, SciQ-FT), and μ is then swept over layer indices.

Table 5: Per-setting bandwidth σ used for LMI, together with the search range of μ. The selected μ^* per row is reported in Tab. 10–13. Settings marked "fixed" run no search: they reuse the (μ^*, σ) selected for the same model and editor under ZsRE SFT at the 10^3 scale.

Downstream task	Model×Method	σ	μ range
ZsRE SFT	GPT-J / MEMIT	5	$\{0, \ldots, 14\}$
ZsRE SFT	GPT-J / AlphaEdit	4	$\{0, \ldots, 14\}$
ZsRE SFT	Llama-3 / MEMIT	14	$\{4, \ldots, 14\}$
ZsRE SFT	Llama-3 / AlphaEdit	14	$\{4, \ldots, 14\}$
ZsRE SFT	Qwen2.5 / MEMIT	10	$\{4, \ldots, 18\}$
ZsRE SFT	Qwen2.5 / AlphaEdit	10	$\{4, \ldots, 18\}$
ZsRE LoRA	GPT-J / MEMIT	3	$\{0, \ldots, 14\}$
ZsRE LoRA	GPT-J / AlphaEdit	3	$\{0, \ldots, 14\}$
ZsRE LoRA	Llama-3 / MEMIT	6	$\{4, \ldots, 16\}$
ZsRE LoRA	Llama-3 / AlphaEdit	6	$\{4, \ldots, 16\}$
SciQ SFT	GPT-J / MEMIT	6	$\{0, \ldots, 14\}$
SciQ SFT	GPT-J / AlphaEdit	6	$\{0, \ldots, 14\}$
SciQ SFT	Llama-3 / MEMIT	20	$\{4, \ldots, 16\}$
SciQ SFT	Llama-3 / AlphaEdit	14	$\{4, \ldots, 16\}$
CF SFT (ZsRE edits)	GPT-J / MEMIT	6 (5 at Full)	$\{3, \ldots, 14\}$
CF SFT (ZsRE edits)	GPT-J / AlphaEdit	6 (5 at Full)	$\{3, \ldots, 14\}$
CF SFT (ZsRE edits)	Llama-3 / MEMIT	14	$\{4, \ldots, 14\}$
CF SFT (ZsRE edits)	Llama-3 / AlphaEdit	14	$\{4, \ldots, 14\}$
NQ-open SFT; ZsRE SFT with attention-only or full-parameter updates	GPT-J / MEMIT	5	fixed, μ=6
	GPT-J / AlphaEdit	4	fixed, μ=6
	Llama-3 / MEMIT	14	fixed, μ=10
	Llama-3 / AlphaEdit	14	fixed, μ=9

B.7 Hyperparameter Selection

For UMI and LMI, we pick the sweet spot by maximising the product $\text{EA} \times \text{Tr.}$ over the swept grid; this is the same criterion used to populate the α^* and μ^*/σ^* columns of Tab. 6–13. The product form rewards configurations that simultaneously recover edits and retain downstream task accuracy, and is

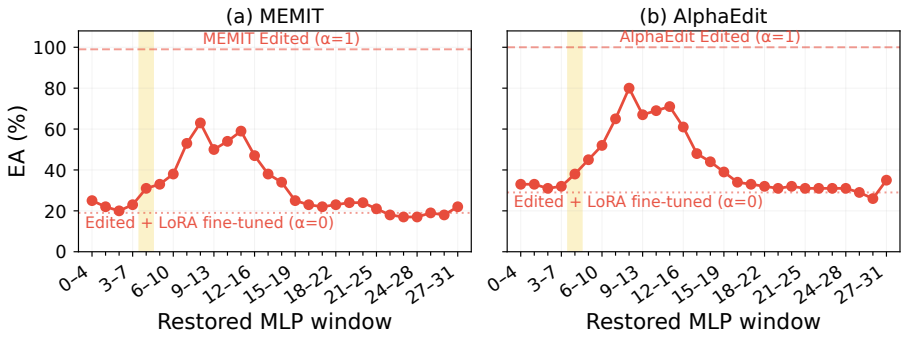


Figure 10: Sliding-window restoration on Llama-3-8B after 100 CF edits and LoRA fine-tuning on ZsRE, with (a) MEMIT and (b) AlphaEdit. Each point restores the MLP weights of five consecutive layers from the edited model into the fine-tuned model. The shaded strip marks the window that coincides with the edit layers (L4–8). Dashed and dotted lines give the EA of the edited model (α=1) and of the fine-tuned model (α=0).

well-defined because $\mathrm{EA}, \mathrm{Tr.} \in [0, 1]$. Because the benchmark protocol does not define a separate edit-validation split for CF, we follow prior work and report numbers at this selected sweet spot on the same evaluation set; the corresponding Te. column, which is computed on a held-out slice and therefore not part of the selection criterion, provides an unbiased view of downstream quality. For the recovery range $[\alpha_{\min}, \alpha_{\max}]$ inside LMI we re-use the default $[0.1, 0.9]$ without further tuning. RCL has no continuous hyperparameter; its layer band is fixed by the mechanistic findings of §4.3.

B.8 Compute Environment

All experiments run on a single GPU per job (NVIDIA H200 NVL and A100-40GB depending on availability) with 8 CPU cores and 88 GB host RAM on a Slurm-managed cluster.

The deployment-time cost of our method is dominated by a single weight merge plus one pass of edit/downstream evaluation. On an H200, one LMI merge at a given (μ, σ), including loading W_{ed} and W_{ft}, computing the per-layer Gaussian-weighted convex combination, and writing the merged checkpoint, takes ≈ 30 s; the full-set evaluation that produces the ES/LR/EA/Tr./Te. row reported in Tab. 10–13 takes $\approx$ 12–15 min, dominated by the strict-EA argmax probe over the entire edit set. UMI has identical merge cost. RCL is cheaper at merge time ($\approx$ 10 s, since it is a binary layer-band overwrite) and shares the same evaluation cost. At deployment, recovering edits with LMI therefore adds well under half an hour of single-GPU compute on top of an already-fine-tuned model, and produces a standard `transformers` checkpoint that can be served without any further runtime overhead.

C Detailed Experiments Results

This appendix reports the full evaluation underlying the main-text recovery comparison. We give two parallel batteries of results: one for the Uniform Model Interpolation (UMI) baseline (§C.2, Tab. 6–9) and one for Layer-wise Model Interpolation (§C.3, Tab. 10–13). Each battery sweeps the same four transfer settings: ZsRE supervised fine-tuning, ZsRE LoRA fine-tuning, ZsRE-space edits followed by CF fine-tuning, and SciQ full-dataset fine-tuning. All numbers are reported on the full evaluation set described in §B.5; the sweet-spot α^* for UMI and μ^*/σ^* for LMI are selected by maximising $\mathrm{EA} \times \mathrm{Tr.}$ as defined in §B.7.

C.1 Sliding-window restoration on Llama-3 and Qwen2.5.

Llama3. Fig. 10 repeats the sliding-window analysis of Fig. 4 on Llama-3-8B, after 100 CF edits and LoRA fine-tuning on ZsRE. Each point restores the MLP weights of five consecutive layers from the edited model into the fine-tuned model and leaves all other layers untouched. Restoring the edit layers themselves (window 4–8) recovers little: EA rises from 19% to 31% for MEMIT and from

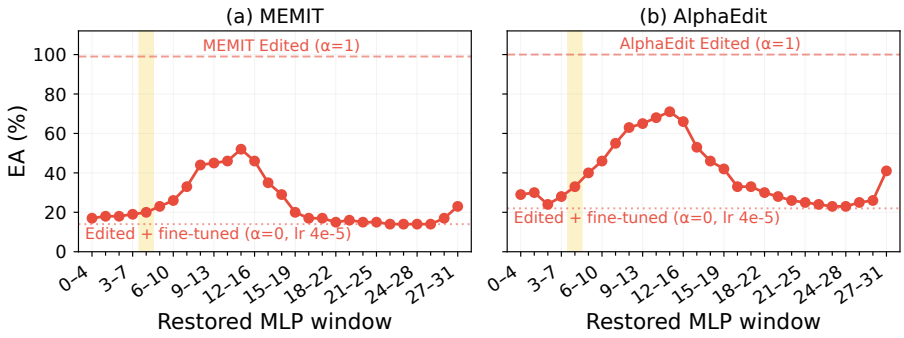


Figure 11: Sliding-window restoration on Llama-3-8B @ 10^2 under ZsRE MLP-only SFT at a milder learning rate (4×10^{-5}; the main-text setting uses 10^{-4}). Each point restores one 5-layer window of post-edit MLP weights into the fine-tuned model and reports EA on the 100 edited facts; the dashed and dotted lines mark the post-edit (α=1) and post-FT (α=0) endpoints, and the shaded strip marks the edit-layer window (L4–8). (a) MEMIT, (b) AlphaEdit. Both editors peak at L11–15 (EA 52 and 71, from post-FT 14 and 22), downstream of the edit layers, while ZsRE Tr. stays at 100 for every window.

29% to 38% for AlphaEdit. The largest recovery comes from the window immediately downstream, 7–12, which reaches 63% (MEMIT) and 80% (AlphaEdit); EA stays clearly above the fine-tuned baseline up to window 12–16 (47% and 61%) and returns to it from window 16–20 onwards. ZsRE training accuracy remains at about 99% for every window, so the recovered edits do not come at the cost of the fine-tuned task. The elevated windows together span layers 7–16, the band we restore in RCL for Llama-3, and the profile mirrors the one observed for GPT-J: the layers that matter most for reading out an edit lie just after the layers in which the edit is written.

Fig. 11 repeats the analysis with MLP-only SFT on ZsRE. We use a learning rate of 4×10^{-5} here (ZsRE training accuracy still reaches 100%). The band reappears at the same location as under LoRA. Restoring the edit layers themselves (window 4–8) gives 20% (MEMIT) and 33% (AlphaEdit) from post-FT values of 14% and 22%; the peak sits at window 11–15 with 52% and 71%, EA remains elevated through window 12–16 (46% and 66%), and it returns to the baseline from window 16–20 onwards. ZsRE training accuracy is 100% for every window. The readout band on Llama-3 is therefore not an artefact of LoRA: it lies in the same layers under both fine-tuning regimes, and what LoRA changes is only how little collateral damage there is outside the band.

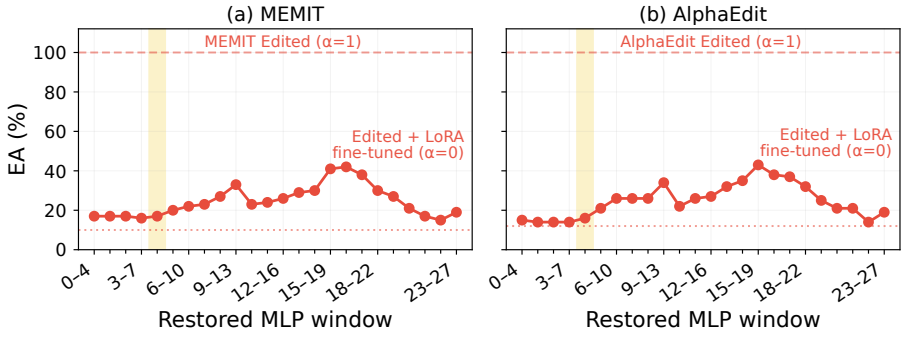


Figure 12: Sliding-window MLP restoration on Qwen2.5-7B @ 10^2 after ZsRE LoRA fine-tuning (5-layer windows, step 2). (a) MEMIT, (b) AlphaEdit. Dashed and dotted lines mark the post-edit (α=1) and post-FT (α=0) endpoints; the shaded strip is the edit-layer window 4–8.

Qwen2.5. Figures 12 and 13 repeat the analysis on Qwen2.5-7B (28 layers, edit layers 4–8) under LoRA and SFT (4×10^{-5}) fine-tuning. Restoring the edit layers again recovers little (window 4–8: 17%/16% under LoRA, 37%/41% under SFT, from post-FT values of 10%/12% and 24%/26% for MEMIT/AlphaEdit). Unlike GPT-J and Llama-3, the recovery is bimodal: one band around

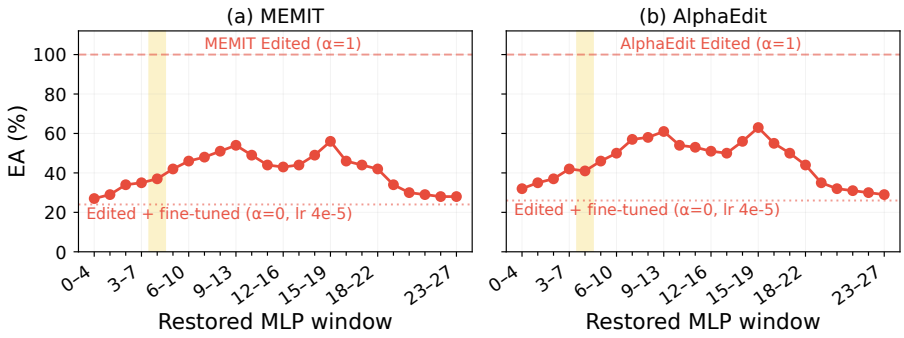


Figure 13: Sliding-window MLP restoration on Qwen2.5-7B @ 10^2 after ZsRE SFT (5-layer windows, step 2). (a) MEMIT, (b) AlphaEdit. Dashed and dotted lines mark the post-edit (α=1) and post-FT (α=0) endpoints; the shaded strip is the edit-layer window 4–8.

windows 8–12 (51%/58% under SFT) and a second one around windows 14–20 (49%/56% under SFT, 42%/38% under LoRA, where it is the dominant peak). EA returns to the baseline from window 20–24 onwards, and ZsRE training accuracy stays at 96–100% for every window up to 20–24. Both bands lie well downstream of the edit layers, and the LMI profile selected for Qwen2.5 (μ^*=13–14, σ=10) is centered between them, covering both.

C.2 Uniform Interpolation

UMI applies a single scalar α to every MLP layer, so it ablates LMI's layer-wise prior while retaining the convex-combination form of the recovery. The four tables below report the post-edit checkpoint (α=1), the post-fine-tune checkpoint (α=0), and the UMI sweet-spot α^* for each model×method×scale cell.

ZsRE supervised fine-tuning (Tab. 6). The selected α^* falls in a narrow band (0.4–0.8 across all 16 rows) and increases mildly with edit scale, indicating that more aggressive edit re-injection is needed when the fine-tuned damage is larger. UMI recovers a substantial fraction of the post-edit EA: on GPT-J with MEMIT @ 10^2 EA is restored from 25.0 to 100.0, and on Llama-3 with MEMIT @ 10^2 from 2.0 to 58.0; however, recovery decays at larger scales: e.g., AlphaEdit @ Full GPT-J only reaches EA 18.3 (post-edit 57.6). Train accuracy degrades modestly from the post-FT value of ≈ 100 to $\approx$ 80–99, while held-out Te. is generally preserved or slightly improved.

ZsRE LoRA fine-tuning (Tab. 7). LoRA's MLP-only adapters perturb a smaller subspace of the MLP weights than full SFT, and the post-FT EA collapse is correspondingly less severe (e.g., 48.0 for GPT-J MEMIT @ 10^2 vs. 25.0 under SFT). The sweet-spot α^* concentrates tightly at 0.4–0.5 on GPT-J and at 0.6 on Llama-3 across nearly every row, suggesting that an LR-invariant scalar exists per model family under LoRA. UMI almost fully restores EA in the small-scale regime (≥ 91 for $S \leq 10^3$ on GPT-J), and downstream Tr. drops only by $\sim$ 1–15 points relative to the post-FT value.

ZsRE-space edits, CF fine-tuning (Tab. 8). This transfer setting inverts the edit/fine-tune roles: edits are applied in the ZsRE relation space and fine-tuning then injects knowledge from CF, the dataset usually used as the source of edits. Because the FT task is itself a knowledge update that conflicts with the edits, recovery is intrinsically harder than in the SFT/LoRA settings above, and UMI partial-recovers EA but never fully closes the gap (e.g., GPT-J AlphaEdit @ Full: post-edit 34.6, post-FT 0.1, UMI 25.3). The sweet-spot α^* is again narrow (0.5–0.6 on GPT-J, 0.8 on Llama-3), and Llama-3 consistently needs more aggressive edit re-injection (higher α) because its post-FT EA is essentially zero.

SciQ fine-tuning (Tab. 9). SciQ is the most domain-orthogonal downstream task we consider: it is multiple-choice science QA, with no relation overlap with CF. Post-FT EA collapses sharply on both models (e.g., Llama-3 MEMIT @ 10^2: EA 0.0 after SciQ FT). UMI recovers most of the edit accuracy at small/medium scales ($\geq$ 89 EA for $S \leq 10^3$ on Llama-3 MEMIT, $\geq$ 96 on GPT-J) while

Table 6: Edit success and locality across editing methods, before and after ZsRE fine-tuning, and at the sweet-spot interpolation point. Full evaluation. ES: edit success; LR: loose-retain rate (#retained / #edits); EA: edit accuracy; Tr. and Te. denote ZsRE training and test accuracy from `analyze_model_interpolation`. ES/LR/EA/Tr./Te. are reported as percentages (%); α is the raw interpolation coefficient.

Model	Method @ Scale	Post-Edit					Post-Finetune					UMI (α^*)					
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	α^*
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.0	0.0	97.0	84.0	25.0	100.0	21.0	100.0	100.0	100.0	96.6	22.0	0.5
	MEMIT @ 10^3	99.4	96.6	96.5	0.0	0.0	73.5	49.3	8.7	100.0	16.0	96.8	89.8	83.4	86.0	22.0	0.6
	MEMIT @ 10^4	99.0	96.7	95.5	0.0	0.0	72.4	45.9	11.6	100.0	17.0	95.0	88.1	81.8	86.1	18.0	0.6
	MEMIT @ Full	97.8	88.0	85.1	0.0	0.0	61.3	28.6	5.9	100.0	16.0	91.8	76.6	68.9	81.8	20.0	0.6
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.0	0.0	96.0	84.0	31.0	100.0	20.0	100.0	100.0	97.0	99.0	21.0	0.4
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.0	0.0	91.1	70.7	27.4	100.0	19.0	99.5	98.6	95.9	97.4	23.0	0.5
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	61.6	30.9	6.0	100.0	15.0	92.6	81.7	72.0	81.8	9.0	0.6
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	17.3	3.4	0.4	100.0	17.0	54.1	28.4	18.3	77.3	15.0	0.5
Llama3-8B	MEMIT @ 10^2	100.0	100.0	100.0	3.8	8.0	40.0	20.0	2.0	100.0	27.0	87.0	78.0	58.0	99.0	43.0	0.6
	MEMIT @ 10^3	96.8	94.2	93.3	3.5	7.0	44.1	23.2	3.1	100.0	22.0	81.0	71.3	54.3	98.8	39.0	0.6
	MEMIT @ 10^4	90.8	86.4	83.9	3.7	4.0	28.6	9.4	2.0	100.0	14.0	78.8	66.5	56.7	82.6	33.0	0.8
	MEMIT @ Full	93.7	82.3	79.9	1.9	4.0	27.0	8.7	1.5	100.0	15.0	78.8	60.4	50.7	83.1	43.0	0.8
	AlphaEdit @ 10^2	100.0	100.0	100.0	3.3	5.0	56.0	33.0	2.0	100.0	21.0	94.0	90.0	79.0	98.7	40.0	0.6
	AlphaEdit @ 10^3	100.0	99.9	99.9	2.7	5.0	50.5	23.4	4.0	100.0	17.0	89.3	79.7	65.4	98.8	42.0	0.6
	AlphaEdit @ 10^4	96.9	94.1	92.2	1.9	2.0	32.4	10.3	2.0	100.0	14.0	88.3	78.2	68.6	82.2	34.0	0.8
	AlphaEdit @ Full	95.4	88.6	85.4	0.5	0.0	30.9	10.7	2.0	100.0	17.0	66.1	47.4	33.1	99.6	26.0	0.6

Table 7: Edit success and locality across editing methods, before and after ZsRE LoRA fine-tuning (lr=8×10^{-4}), and at the sweet-spot interpolation point. Full evaluation. ES: edit success; LR: loose-retain rate (#retained / #edits); EA: edit accuracy; Tr. and Te. denote ZsRE training and test accuracy, respectively. α^*: sweet-spot interpolation coefficient (selected by max EA $\times$ Tr.). ES/LR/EA/Tr./Te. are reported as percentages (%); α^* is the raw interpolation coefficient.

Model	Method @ Scale	Post-Edit					Post-Finetune					UMI					
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	α^*
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.0	0.0	99.0	95.0	48.0	100.0	17.0	100.0	100.0	100.0	95.0	21.0	0.5
	MEMIT @ 10^3	99.4	96.6	96.5	0.0	0.0	89.9	71.8	24.7	100.0	21.0	98.8	95.5	91.1	89.3	21.0	0.5
	MEMIT @ 10^4	99.0	96.7	95.5	0.0	0.0	76.7	53.3	15.1	100.0	19.0	95.1	88.8	82.5	88.9	16.0	0.5
	MEMIT @ Full	97.8	88.0	85.1	0.0	0.0	65.8	38.4	8.3	99.9	20.0	91.8	78.4	69.4	86.3	18.0	0.5
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.0	0.0	100.0	96.0	56.0	100.0	19.0	100.0	100.0	98.0	97.6	21.0	0.4
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.0	0.0	96.4	88.9	41.2	99.7	23.0	99.7	99.1	94.8	97.5	22.0	0.4
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	65.6	38.2	9.6	100.0	13.0	92.5	82.3	73.0	85.8	13.0	0.5
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	41.0	11.7	1.7	99.7	18.0	65.5	38.3	23.6	72.1	17.0	0.5
Llama3-8B	MEMIT @ 10^2	100.0	100.0	100.0	3.8	8.0	84.0	74.0	15.0	99.7	26.0	98.0	99.0	83.0	91.1	44.0	0.6
	MEMIT @ 10^3	96.8	94.2	93.3	4.0	7.0	77.7	66.1	11.9	99.7	29.0	94.3	89.0	75.8	91.0	47.0	0.6
	MEMIT @ 10^4	90.8	86.4	83.9	4.0	4.0	72.3	59.0	16.8	99.8	41.0	87.8	81.4	74.5	84.2	49.0	0.6
	MEMIT @ Full	93.7	82.3	79.9	1.9	4.0	68.7	47.6	10.9	99.8	40.0	89.0	76.7	67.5	84.1	42.0	0.6
	AlphaEdit @ 10^2	100.0	100.0	100.0	3.3	5.0	94.0	82.0	16.0	99.7	32.0	100.0	99.0	82.0	91.4	56.0	0.6
	AlphaEdit @ 10^3	100.0	99.9	99.9	2.7	5.0	88.4	79.0	23.1	99.1	23.0	99.2	97.9	91.9	91.2	39.0	0.6
	AlphaEdit @ 10^4	96.9	94.1	92.2	1.9	2.0	77.7	59.8	16.3	99.6	29.0	93.9	88.7	81.5	88.2	37.0	0.6
	AlphaEdit @ Full	95.4	88.6	85.4	0.5	0.0	65.9	45.7	5.0	100.0	15.0	89.9	78.1	55.6	86.4	28.0	0.6

Table 8: After **ZsRE-space** edits, MLP-only finetune on **CF** (same lr as CF-only FT: GPT-J 6×10^{-5}, Llama-3-8B 10^{-4}), with linear interpolation between edited and CF-FT checkpoints. ES/LR/EA/CF metrics in %; LR = loose-retained fraction $\times100$. $\mathrm{CF_{tr}}$ / $\mathrm{CF_{te}}$ from `train_acc` / `test_acc` (missing test shown as —). α^* maximizes EA $\times$ $\mathrm{CF_{tr}}$ over recorded α.

Model	Method @ Scale	Post-ZsRE edit (α=1)					Post-CF FT (α=0)					Layer-wise Model Interp. (α^*)					
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	α^*
GPT-J-6B	MEMIT @ 10^2	91.0	72.0	75.0	16.0	22.0	76.0	20.0	16.0	95.0	53.0	85.0	57.0	53.0	90.0	52.0	0.5
	MEMIT @ 10^3	88.5	82.4	85.0	15.9	21.0	69.4	17.5	9.7	95.4	47.0	85.8	65.7	66.3	88.3	46.0	0.6
	MEMIT @ 10^4	92.8	71.8	74.0	14.4	19.0	71.4	12.0	6.1	95.1	48.0	83.0	42.2	38.8	87.8	47.0	0.5
	MEMIT @ Full	92.8	66.7	70.8	11.9	17.0	68.7	7.4	2.6	95.1	44.0	81.2	35.4	29.3	85.6	45.0	0.5
	AlphaEdit @ 10^2	95.0	94.0	90.0	16.4	21.0	83.0	26.0	14.0	95.0	50.0	92.0	74.0	71.0	79.2	50.0	0.6
	AlphaEdit @ 10^3	97.4	93.6	91.6	14.0	20.0	83.3	26.8	17.4	95.2	47.0	96.1	80.9	80.7	79.6	47.0	0.6
	AlphaEdit @ 10^4	94.3	58.9	60.3	6.9	9.0	65.2	9.3	1.0	94.4	41.0	84.2	43.7	37.1	73.8	43.0	0.6
	AlphaEdit @ Full	87.5	31.7	34.6	2.0	2.0	61.6	3.3	0.1	94.3	44.0	80.5	29.7	25.3	67.5	36.0	0.6
Llama-3-8B	MEMIT @ 10^2	71.0	44.0	46.0	27.0	30.0	53.0	9.0	0.0	98.0	65.0	72.0	30.0	32.0	69.0	52.0	0.8
	MEMIT @ 10^3	58.0	46.7	58.8	27.0	32.0	42.1	4.0	0.7	96.4	65.0	53.8	32.5	39.8	74.3	52.0	0.8
	MEMIT @ 10^4	71.0	41.7	43.5	27.6	27.0	48.3	4.1	0.1	97.5	65.0	62.0	28.4	24.1	70.7	50.0	0.8
	MEMIT @ Full	71.0	41.7	43.0	28.0	27.0	48.0	4.1	0.0	98.0	65.0	62.0	28.4	24.0	71.0	50.0	0.8
	AlphaEdit @ 10^2	97.0	89.0	84.0	27.0	31.0	60.0	9.0	0.0	98.0	64.0	87.0	69.0	66.0	72.0	52.0	0.8
	AlphaEdit @ 10^3	91.8	77.1	78.2	27.4	28.0	54.1	4.5	1.2	96.8	65.0	82.4	57.3	54.9	70.0	51.0	0.8
	AlphaEdit @ 10^4	84.2	59.8	61.8	24.3	28.0	48.0	6.0	1.3	96.7	63.0	70.8	39.5	35.9	75.1	53.0	0.8
	AlphaEdit @ Full	84.2	52.5	58.2	20.1	21.0	49.3	4.4	0.2	96.5	60.0	72.5	38.6	33.4	79.9	54.0	0.8

preserving SciQ Tr. within $\sim$ 5–15 points of the post-FT value. The selected α^* is somewhat lower on GPT-J (0.4–0.6) than on Llama-3 (0.6–0.8), mirroring the SFT/LoRA pattern.

Table 9: SciQ full-dataset fine-tuning after CF edits (GPT-J: lr=6×10^{-5}; Llama-3-8B: lr=10^{-4}), with linear interpolation between edited and SciQ-FT checkpoints. ES/LR/EA/Tr./Te. are reported in %; LR = loose-retained fraction $\times100$. Tr. and Te. denote SciQ training and test accuracy, respectively. Sweet spot α^* maximizes EA $\times$ Tr.

Model	Method	Post-Edit (α=1)					Post-SciQ FT (α=0)					UMI (α^*)					
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	α^*
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.4	0.2	96.0	76.0	54.0	94.8	47.0	100.0	97.0	96.0	91.1	48.3	0.4
	MEMIT @ 10^3	99.4	96.6	96.5	0.5	0.2	97.1	65.4	42.7	94.6	47.4	99.7	94.1	91.2	85.2	48.1	0.4
	MEMIT @ 10^4	99.0	96.7	95.5	0.2	0.6	86.1	42.2	24.3	94.6	47.2	97.0	88.4	83.2	82.1	46.4	0.5
	MEMIT @ Full	97.8	88.0	85.1	0.2	0.3	79.9	26.7	12.5	94.5	47.0	95.3	75.1	67.9	81.0	46.3	0.5
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.4	0.2	99.0	84.0	65.0	94.8	46.1	100.0	99.0	99.0	90.9	48.1	0.4
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.4	0.2	97.9	74.3	51.3	94.5	48.1	99.9	98.3	96.5	86.1	48.7	0.4
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	73.5	21.8	8.2	94.4	46.5	94.6	76.3	67.3	81.0	44.8	0.5
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	52.2	6.1	1.3	94.3	44.7	85.4	48.4	38.0	59.6	39.4	0.6
Llama-3-8B	MEMIT @ 10^2	100.0	100.0	100.0	16.6	18.9	34.0	9.0	0.0	94.5	45.6	99.0	99.0	99.0	77.0	57.1	0.8
	MEMIT @ 10^3	96.8	94.2	93.3	15.8	17.5	38.4	9.2	1.6	94.7	49.3	93.5	88.8	85.7	76.7	58.1	0.8
	MEMIT @ 10^4	90.8	86.4	83.9	14.2	15.7	28.7	4.8	1.3	94.5	48.0	83.9	74.9	68.3	80.2	57.7	0.8
	MEMIT @ Full	93.7	82.3	79.9	8.2	8.9	33.6	5.2	0.8	94.4	46.2	89.5	75.8	69.8	79.8	57.7	0.8
	AlphaEdit @ 10^2	100.0	100.0	100.0	16.6	19.4	66.0	22.0	5.0	94.6	47.0	96.0	97.0	94.0	93.4	57.2	0.6
	AlphaEdit @ 10^3	100.0	99.9	99.9	17.9	19.8	60.2	20.9	4.8	94.6	50.0	97.0	91.5	85.9	90.9	58.1	0.6
	AlphaEdit @ 10^4	96.9	94.1	92.2	14.8	16.4	33.4	4.2	0.8	94.5	47.1	93.1	87.1	81.5	78.7	57.5	0.8
	AlphaEdit @ Full	95.4	88.6	85.4	1.0	1.2	29.5	3.5	0.6	94.5	45.5	91.5	79.4	70.7	78.2	56.5	0.8

C.3 Layer-wise Model Interpolation

We now report the same four transfer settings with LMI, the Gaussian-kernel layer-wise interpolation introduced in §5. Per-row μ^*/σ^* are the sweet-spot center/width selected by maximising EA $\times$ Tr. over the grid in Tab. 5. Reading the μ^* column across the four tables provides empirical evidence for the readout-circuitry localization claim of §4.3: the selected center concentrates on the deep editing and adjacent post-edit layers, and shifts predictably with the model architecture and the downstream fine-tuning recipe.

ZsRE supervised fine-tuning (Tab. 10). On GPT-J, the selected μ^* is concentrated at layer 6 for all rows except AlphaEdit @ Full, exactly on the boundary between deep editing layers (L3–8) and adjacent post-edit layers (L9–14) identified in §4.3 as the readout circuitry. On Llama-3, μ^* shifts down-stream to layers 9–11, consistent with the readout band of L7–16 identified by the sliding-window analysis on Llama-3. Compared to UMI, LMI gains 5–25 EA points on most rows (e.g., GPT-J MEMIT @ Full: 73.0 vs. UMI 68.9; Llama-3 MEMIT @ 10^2: 83.0 vs. UMI 58.0), while sacrificing at most ~ 5 Tr. points relative to UMI.

Table 10: Edit success and locality across editing methods, before and after ZsRE fine-tuning, and at the layerwise Gaussian interpolation sweet spot. Full evaluation. ES: edit success; LR: loose-retain rate (#retained / #edits); EA: edit accuracy; Tr. and Te. denote ZsRE training and test accuracy from `analyze_model_interpolation`. ES/LR/EA/Tr./Te. are reported as percentages (%); μ^*/σ^* denotes the selected Gaussian interpolation center and width.

Model	Method @ Scale	Post-Edit					Post-Finetune					Layerwise Model Interp.					μ^*/σ^*
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.0	0.0	97.0	84.0	25.0	100.0	21.0	100.0	100.0	100.0	94.6	21.0	6/5
	MEMIT @ 10^3	99.4	96.6	96.5	0.0	0.0	73.5	49.3	8.7	100.0	16.0	98.5	93.8	90.0	91.2	17.0	6/5
	MEMIT @ 10^4	99.0	96.7	95.5	0.0	0.0	72.4	45.9	11.6	100.0	17.0	95.6	90.1	84.1	91.7	16.0	6/5
	MEMIT @ Full	97.8	88.0	85.1	0.0	0.0	61.3	28.6	5.9	100.0	16.0	94.5	78.9	73.0	86.6	16.0	4/5
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.0	0.0	96.0	84.0	31.0	100.0	20.0	100.0	100.0	100.0	97.7	17.0	6/4
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.0	0.0	91.1	70.7	27.4	100.0	19.0	99.2	98.9	96.9	98.0	16.0	6/4
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	61.6	30.9	6.0	100.0	15.0	89.3	71.7	58.7	95.5	13.0	6/4
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	17.3	3.4	0.4	100.0	17.0	89.8	68.4	40.2	41.7	6.0	3/4
Llama3-8B	MEMIT @ 10^2	100.0	100.0	100.0	3.8	8.0	40.0	20.0	2.0	100.0	27.0	97.0	92.0	83.0	92.2	37.0	10/14
	MEMIT @ 10^3	96.8	94.2	93.3	3.5	7.0	44.1	23.2	3.1	100.0	22.0	92.0	81.3	72.9	90.4	41.0	10/14
	MEMIT @ 10^4	90.8	86.4	83.9	3.7	4.0	28.6	9.4	2.0	100.0	14.0	80.8	77.4	53.9	93.7	32.0	10/14
	MEMIT @ Full	93.7	82.3	79.9	1.9	4.0	27.0	8.7	1.5	100.0	15.0	82.7	67.2	50.5	94.6	37.0	7/14
	AlphaEdit @ 10^2	100.0	100.0	100.0	3.3	5.0	56.0	33.0	2.0	100.0	21.0	100.0	98.0	95.0	90.4	34.0	9/14
	AlphaEdit @ 10^3	100.0	99.9	99.9	2.7	5.0	50.5	23.4	4.0	100.0	17.0	98.7	93.4	89.3	94.2	34.0	9/14
	AlphaEdit @ 10^4	96.9	94.1	92.2	1.9	2.0	32.4	10.3	2.0	100.0	14.0	89.1	75.2	66.8	95.3	20.0	9/14
	AlphaEdit @ Full	95.4	88.6	85.4	0.5	0.0	30.9	10.7	2.0	100.0	17.0	88.9	72.8	59.5	95.2	20.0	7/14

ZsRE LoRA fine-tuning (Tab. 11). LoRA produces a more spatially concentrated MLP perturbation than full SFT, which is reflected in a noticeably tighter bandwidth $\sigma^*=3$ for GPT-J (vs. $\sigma\in\{4,5\}$ under SFT) and a higher $\mu^*=9$ for most rows. In other words, LoRA's damage is narrowly localized in the adjacent post-edit band, and LMI's kernel automatically tightens to match. The recovery gain over UMI is correspondingly larger (e.g., AlphaEdit @ 10^3 on GPT-J: LMI EA 98.0 vs. UMI 94.8; @ 10^4: 54.4 vs. 73.0, the only row where UMI surpasses LMI in EA, though both still nearly recover the post-edit ceiling). The cost in Tr. is consistently within ~ 5 points of the post-FT value across all 16 rows.

ZsRE-space edits, CF fine-tuning (Tab. 12). This is the hardest setting of the four, since fine-tuning on CF directly injects competing knowledge into the parametric memory. On GPT-J, $\mu^*=5$–6 for most rows, sitting at the lower edge of the readout band; the bandwidth $\sigma^*=5$–6 remains tight. On Llama-3, $\mu^*=7$–8 with broader $\sigma^*=14$, reflecting Llama-3's wider readout circuitry. LMI's gain over UMI is more modest in this setting (typical $\Delta\text{EA} \approx 2$–8 points) but consistent in direction, and the downstream CF Tr. recovers to 63–92% (vs. UMI's 63–80%), indicating that LMI sacrifices less downstream knowledge while injecting the edit.

SciQ fine-tuning (Tab. 13). SciQ pulls a broader range of MLP layers than relation-style fine-tuning, since answering a science multiple-choice question recruits more general semantic processing rather than a narrow relation-specific pathway. The selected bandwidth σ^* correspondingly widens to 14–20 on Llama-3, the largest values in the entire grid, while GPT-J retains $\sigma^*=5$–6. LMI achieves near-ceiling edit recovery at small/medium scales (e.g., Llama-3 MEMIT @ 10^2: EA 97.0 vs. post-edit 100.0 and post-FT 0.0), and maintains SciQ Tr. within $\sim$ 10–20 points of the post-FT value across all rows.

D Grad-ELLM Attribution: Method and Additional Case Studies

The case study reported in §6.2 of the main text (Fig. 8) is the attention-based interpretability analysis referenced in our abstract. It builds on Grad-ELLM (Huang & Chan, 2026), a gradient $\times$ attention

Table 11: Edit success and locality across editing methods, before and after ZsRE LoRA fine-tuning (lr=8e-4), and at the layerwise Gaussian interpolation sweet-spot. Full evaluation. ES: edit success; LR: loose-retain rate (#retained / #edits); EA: edit accuracy; Tr. and Te. denote ZsRE training and test accuracy, respectively. μ^*/σ^*: per-row sweet-spot of the layerwise Gaussian interpolation kernel (selected by max EA $\times$ Tr.). ES/LR/EA/Tr./Te. are reported as percentages (%).

Model	Method @ Scale	Post-Edit					Post-Finetune					Layerwise Model Interp.					μ^*/σ^*
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.0	0.0	99.0	95.0	48.0	100.0	17.0	100.0	100.0	100.0	97.7	22.0	9/3
	MEMIT @ 10^3	99.4	96.6	96.5	0.0	0.0	89.9	71.8	24.7	100.0	21.0	98.2	94.8	89.6	97.0	20.0	9/3
	MEMIT @ 10^4	99.0	96.7	95.5	0.0	0.0	76.7	53.3	15.1	100.0	19.0	91.6	87.1	76.3	97.5	20.0	9/3
	MEMIT @ Full	97.8	88.0	85.1	0.0	0.0	65.8	38.4	8.3	99.9	20.0	89.6	77.6	62.2	93.8	16.0	7/3
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.0	0.0	100.0	96.0	56.0	100.0	19.0	100.0	100.0	100.0	97.4	19.0	9/3
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.0	0.0	96.4	88.9	41.2	99.7	23.0	99.5	98.8	98.0	97.4	24.0	9/3
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	65.6	38.2	9.6	100.0	13.0	83.2	76.9	54.4	94.1	10.0	9/3
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	41.0	11.7	1.7	99.7	18.0	81.2	62.1	23.4	55.9	8.0	3/3
Llama3-8B	MEMIT @ 10^2	100.0	100.0	100.0	3.8	8.0	84.0	74.0	15.0	99.7	26.0	99.0	99.0	99.0	95.2	34.0	11/6
	MEMIT @ 10^3	96.8	94.2	93.3	4.0	7.0	77.7	66.1	11.9	99.7	29.0	94.8	91.0	87.5	94.9	28.0	11/6
	MEMIT @ 10^4	90.8	86.4	83.9	4.0	4.0	72.3	59.0	16.8	99.8	41.0	88.5	83.5	77.4	91.2	45.0	11/6
	MEMIT @ Full	93.7	82.3	79.9	1.9	4.0	68.7	47.6	10.9	99.8	40.0	87.9	79.5	70.6	91.5	40.0	12/6
	AlphaEdit @ 10^2	100.0	100.0	100.0	3.3	5.0	94.0	82.0	16.0	99.7	32.0	100.0	100.0	98.0	95.1	34.0	11/6
	AlphaEdit @ 10^3	100.0	99.9	99.9	2.7	5.0	88.4	79.0	23.1	99.1	23.0	99.7	99.4	98.4	94.0	32.0	11/6
	AlphaEdit @ 10^4	96.9	94.1	92.2	1.9	2.0	77.7	59.8	16.3	99.6	29.0	93.7	88.7	85.2	94.9	26.0	11/6
	AlphaEdit @ Full	95.4	88.6	85.4	0.5	0.0	65.9	45.7	5.0	100.0	15.0	89.8	79.9	72.2	91.8	17.0	12/6

Table 12: After **ZsRE-space** edits, MLP-only fine-tuning on **CF** (same lr as CF-only FT: GPT-J 6×10^{-5}, Llama-3-8B 10^{-4}), with layerwise Gaussian interpolation between edited and CF-FT checkpoints. ES/LR/EA/CF metrics in %; LR = loose-retained fraction $\times100$. $\mathrm{CF_{tr}}$ / $\mathrm{CF_{te}}$ from `train_acc` / `test_acc`. μ^*/σ^* maximizes EA $\times$ $\mathrm{CF_{tr}}$.

Model	Method @ Scale	Post-ZsRE edit (α=1)					Post-CF FT (α=0)					Layer-wise Model Interp. (μ^*/σ^*)					μ^*/σ^*
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	
GPT-J-6B	MEMIT @ 10^2	91.0	72.0	75.0	16.0	22.0	76.0	20.0	16.0	95.0	53.0	87.8	57.0	60.2	82.6	54.0	6/6
	MEMIT @ 10^3	88.5	82.4	85.0	15.9	21.0	69.4	17.5	9.7	95.4	47.0	92.5	65.7	65.9	89.4	52.0	6/6
	MEMIT @ 10^4	92.8	71.8	74.0	14.4	19.0	71.4	12.0	6.1	95.1	48.0	87.8	42.2	54.7	81.9	51.0	6/6
	MEMIT @ Full	92.8	66.7	70.8	11.9	17.0	68.7	7.4	2.6	95.1	44.0	86.0	35.4	41.4	81.4	51.0	5/5
	AlphaEdit @ 10^2	95.0	94.0	90.0	16.4	21.0	83.0	26.0	14.0	95.0	50.0	93.9	74.0	75.5	82.2	53.0	5/6
	AlphaEdit @ 10^3	97.4	93.6	91.6	14.0	20.0	83.3	26.8	17.4	95.2	47.0	96.6	80.9	82.1	82.9	52.0	5/6
	AlphaEdit @ 10^4	94.3	58.9	60.3	6.9	9.0	65.2	9.3	1.0	94.4	41.0	88.9	43.7	45.3	75.2	42.0	5/6
	AlphaEdit @ Full	87.5	31.7	34.6	2.0	2.0	61.6	3.3	0.1	94.3	44.0	85.1	29.7	25.1	63.9	41.0	4/5
Llama-3-8B	MEMIT @ 10^2	71.0	44.0	46.0	27.0	30.0	53.0	9.0	0.0	98.0	65.0	71.6	30.0	35.8	88.1	62.0	8/14
	MEMIT @ 10^3	58.0	46.7	58.8	27.0	32.0	42.1	4.0	0.7	96.4	65.0	66.7	32.5	33.6	91.9	66.0	8/14
	MEMIT @ 10^4	71.0	41.7	43.5	27.6	27.0	48.3	4.1	0.1	97.5	65.0	64.3	28.4	29.0	90.2	61.0	8/14
	MEMIT @ Full	71.0	41.7	43.0	28.0	27.0	48.0	4.1	0.0	98.0	65.0	69.4	28.4	34.9	92.7	66.0	7/14
	AlphaEdit @ 10^2	97.0	89.0	84.0	27.0	31.0	60.0	9.0	0.0	98.0	64.0	90.1	69.0	67.9	92.5	64.0	7/14
	AlphaEdit @ 10^3	91.8	77.1	78.2	27.4	28.0	54.1	4.5	1.2	96.8	65.0	85.4	57.3	58.9	92.2	64.0	7/14
	AlphaEdit @ 10^4	84.2	59.8	61.8	24.3	28.0	48.0	6.0	1.3	96.7	63.0	75.0	39.5	39.7	91.8	66.0	7/14
	AlphaEdit @ Full	84.2	52.5	58.2	20.1	21.0	49.3	4.4	0.2	96.5	60.0	74.0	38.6	37.0	91.7	61.0	8/14

Table 13: SciQ full-dataset fine-tuning after CF edits (GPT-J: lr=6×10^{-5}; Llama-3-8B: lr=10^{-4}), with layerwise Gaussian interpolation between edited and SciQ-FT checkpoints. ES/LR/EA/Tr./Te. are reported in %; LR = loose-retained fraction $\times100$. Tr. and Te. denote SciQ training and test accuracy, respectively. Sweet spot μ^*/σ^* maximizes EA $\times$ Tr.

Model	Method	Post-Edit (α=1)					Post-SciQ FT (α=0)					Layer-wise Model Interp.					
		ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	ES	LR	EA	Tr.	Te.	μ^*/σ^*
GPT-J-6B	MEMIT @ 10^2	100.0	100.0	100.0	0.4	0.2	96.0	76.0	54.0	94.8	47.0	100.0	100.0	100.0	76.3	47.3	5/6
	MEMIT @ 10^3	99.4	96.6	96.5	0.5	0.2	97.1	65.4	42.7	94.6	47.4	99.5	97.4	95.3	71.3	46.6	5/6
	MEMIT @ 10^4	99.0	96.7	95.5	0.2	0.6	86.1	42.2	24.3	94.6	47.2	98.6	94.6	89.0	73.9	44.9	5/6
	MEMIT @ Full	97.8	88.0	85.1	0.2	0.3	79.9	26.7	12.5	94.5	47.0	97.4	89.0	72.7	72.0	44.6	3/6
	AlphaEdit @ 10^2	100.0	100.0	100.0	0.4	0.2	99.0	84.0	65.0	94.8	46.1	100.0	100.0	100.0	76.3	46.5	5/6
	AlphaEdit @ 10^3	100.0	99.8	99.8	0.4	0.2	97.9	74.3	51.3	94.5	48.1	100.0	99.2	98.8	71.9	46.9	5/6
	AlphaEdit @ 10^4	99.0	95.4	92.7	0.0	0.0	73.5	21.8	8.2	94.4	46.5	98.5	87.3	79.0	65.5	40.8	5/6
	AlphaEdit @ Full	92.3	66.3	57.6	0.0	0.0	52.2	6.1	1.3	94.3	44.7	83.8	59.5	26.5	51.3	32.6	8/6
Llama-3-8B	MEMIT @ 10^2	100.0	100.0	100.0	16.6	18.9	34.0	9.0	0.0	94.5	45.6	99.0	99.0	97.0	83.7	56.0	5/20
	MEMIT @ 10^3	96.8	94.2	93.3	15.8	17.5	38.4	9.2	1.6	94.7	49.3	94.8	89.9	83.6	82.1	57.5	5/20
	MEMIT @ 10^4	90.8	86.4	83.9	14.2	15.7	28.7	4.8	1.3	94.5	48.0	87.5	79.4	66.9	85.0	57.5	5/20
	MEMIT @ Full	93.7	82.3	79.9	8.2	8.9	33.6	5.2	0.8	94.4	46.2	91.8	77.1	68.6	83.5	57.6	6/20
	AlphaEdit @ 10^2	100.0	100.0	100.0	16.6	19.4	66.0	22.0	5.0	94.6	47.0	100.0	100.0	100.0	89.4	56.6	5/14
	AlphaEdit @ 10^3	100.0	99.9	99.9	17.9	19.8	60.2	20.9	4.8	94.6	50.0	99.4	92.2	93.1	87.4	56.9	5/14
	AlphaEdit @ 10^4	96.9	94.1	92.2	14.8	16.4	33.4	4.2	0.8	94.5	47.1	94.7	89.5	73.2	88.2	55.0	5/14
	AlphaEdit @ Full	95.4	88.6	85.4	1.0	1.2	29.5	3.5	0.6	94.5	45.5	93.0	81.8	62.5	84.7	51.7	8/14

attribution method designed for decoder-only LLMs. This appendix specifies the method as we use it (§D.1), describes the sliding-window aggregation and difference protocol that produces the heatmaps (§D.2), and presents two additional case studies that complement the single case shown in main-text Fig. 8.

D.1 Attribution formula

Given a prompt with T tokens and an output token t^* that we wish to attribute (we use the first generated answer token), Grad-ELLM computes, at every Transformer layer l and every source-token position j, a scalar attribution

$$e_{l,j} \;=\; \mathrm{ReLU}\left(\sum_d \big(\nabla_l\big)_d \cdot \big(v_{l,j}\big)_d \cdot \widetilde{\cos}(q_{l,T},\, k_{l,j})\right), \tag{10}$$

where $q_{l,T}, k_{l,j}, v_{l,j} \in \mathbb{R}^{d_{\text{head}}}$ are the query of the target-position token, and the key/value of the source-position token j, respectively, all taken from layer l's self-attention; $\nabla_l = \partial \log p(t^*)/\partial a_{l,T}$ is the gradient of the target-token log-probability with respect to the attention block output at the target position; and $\widetilde{\cos}$ is a per-layer min–max normalisation of the cosine similarity over source positions. The intuition is that $\widetilde{\cos}$ identifies which source tokens the target attends to, $v_{l,j}$ supplies the value vector that those tokens contribute, and the gradient ∇_l measures how much that contribution influences the target prediction. For grouped-query attention models (Llama-3), keys and values are first repeated to match the query head count before computing Eq. 10.

D.2 Sliding-window aggregation and difference protocol

To match the layer-wise granularity used by the sliding-window MLP-restoration analysis in §4.3, we aggregate the per-layer attributions of Eq. 10 into 6-layer windows with stride 3. For GPT-J (L=28) this yields eight windows: $\mathcal{W} = \{[0,5],[3,8],[6,11],[9,14],[12,17],[15,20],[18,23],[21,27]\}$. The window-level attribution is the per-layer sum $E_{w,j} = \sum_{l\in w} e_{l,j}$, which we then L_1-normalise across source tokens within each window so the rows of the resulting heatmap are comparable across models.

To produce a between-model difference map, we compute $E_{w,j}$ for each of the original, MEMIT-edited, and post-FT checkpoints under the same prompt and target-token choice, then take the pairwise difference $\Delta^{A-B}_{w,j} = E^A_{w,j} - E^B_{w,j}$. Positive (red) cells indicate source tokens that model A relies on more than model B; negative (blue) cells indicate the opposite. Main-text Fig. 8 displays the

fine-tuned − MEMIT difference for one case; Fig. 14 and 15 below give the full 3-model absolute and 3-pair difference heatmaps for two additional cases.

D.3 Additional case studies

Case: Mother tongue of a French actress (Fig. 14). The prompt is "The mother tongue of Danielle Darrieux is"; the unedited model answers "French", and both the MEMIT-edited and the post-FT model answer "English" (the edit target). Reading the top row of Fig. 14, the original model and the MEMIT-edited model both place the bulk of their attribution mass on the subject-end token "ux" (the last sub-word of "Darrieux") from L9 onwards, with MEMIT additionally adding a strong L12–17 peak on "is". The post-FT model, by contrast, distributes attribution much more broadly across the prompt and substantially reduces its reliance on the subject-end token at the deep editing and adjacent post-edit layers. The fine-tuned − MEMIT difference panel (bottom right) makes this rewiring explicit: the L12–17 window contains a strong negative (blue) cell on "is" and positive (red) cells on the subject tokens "Dar/rie/ux", showing that fine-tuning has redirected the retrieval pathway away from the relation-anchor token even though the predicted answer is unchanged.

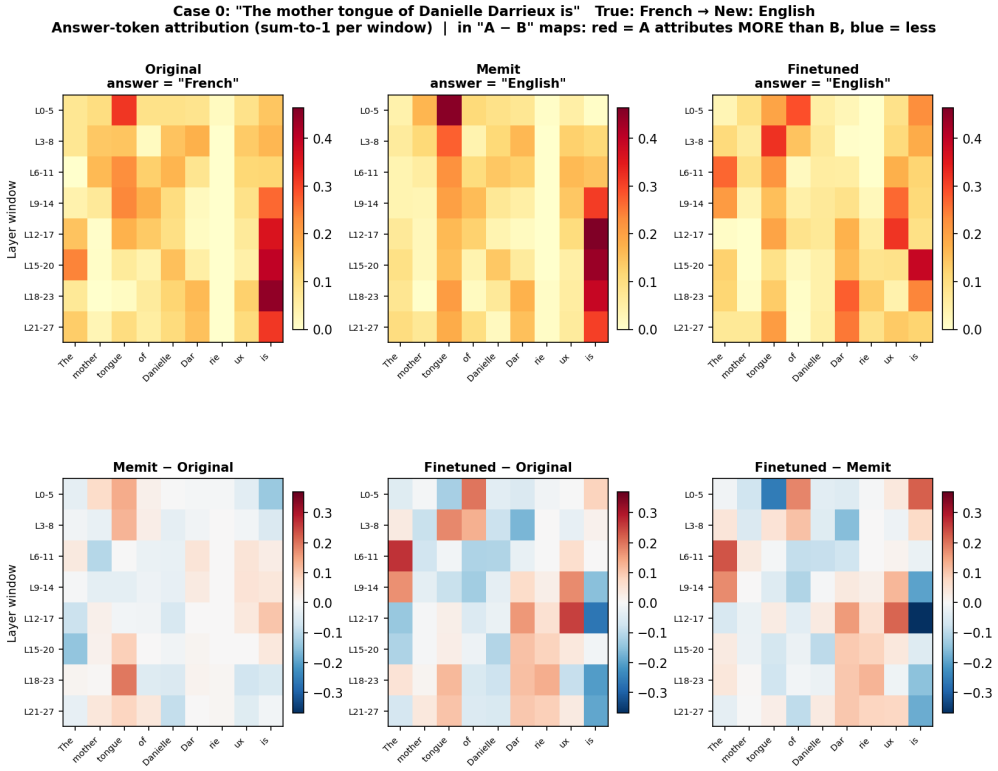


Figure 14: Grad-ELLM attribution for case "The mother tongue of Danielle Darrieux is" (True: *French* → New: *English*). Top row: per-window attribution maps for the unedited (Original), MEMIT-edited (Memit), and post-FT (Finetuned) models, L_1-normalised across source tokens within each window. Bottom row: pairwise differences. Red = A attributes more than B; blue = less. Rows are sliding layer windows; columns are prompt tokens.

Case: Location of a Spanish university (Fig. 15). The prompt is "Autonomous University of Madrid, which is located in"; the unedited model answers "Spain", and both edited variants answer "Sweden". The MEMIT-edited model develops a sharp, single-token attribution onto the subject token "Madrid" at L12–17 (top-middle panel; visible as the dark-red cell). The fine-tuned model preserves an overall focus on "Madrid" but its attribution is shallower and more diffuse, with the L9–14 cell on "Madrid" weakened markedly (fine-tuned − MEMIT diff, bottom right: strong blue cell at L9–14 on "Madrid"). This is the same qualitative pattern as case "The mother tongue of Danielle Darrieux is"

(True: *French* → New: *English*): fine-tuning leaves the prediction intact but rewires which prompt tokens the model attends to at the deep editing and adjacent post-edit layers, exactly the band our §4.3 analysis identifies as the readout circuitry.

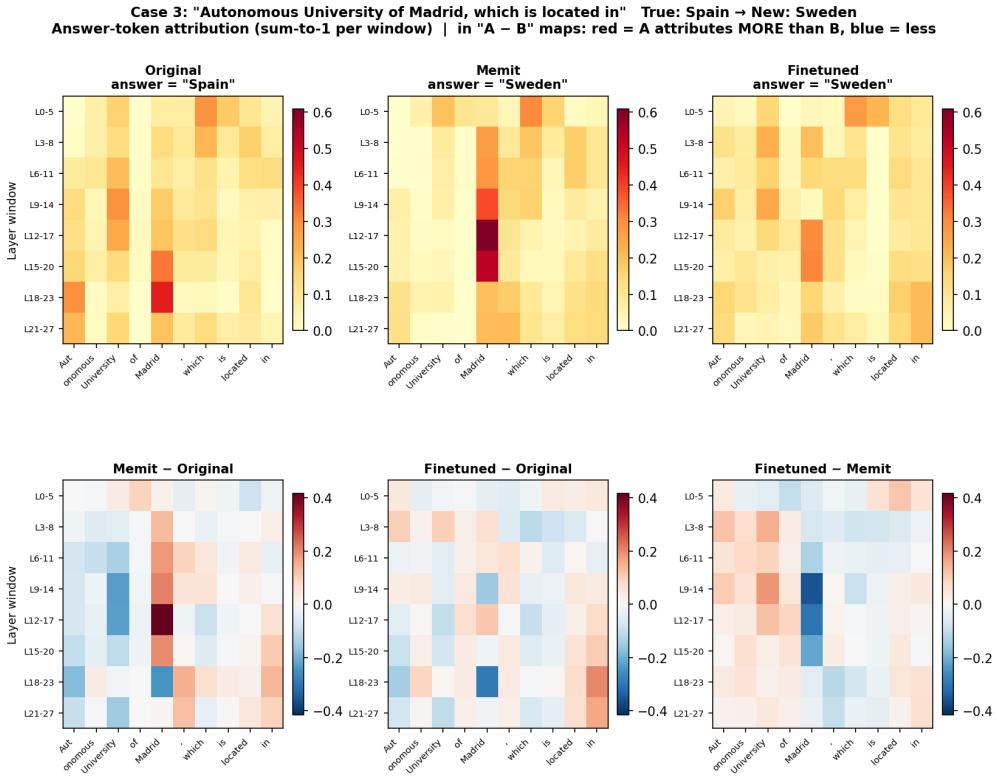


Figure 15: Grad-ELLM attribution for case "Autonomous University of Madrid, which is located in" (True: *Spain* → New: *Sweden*). Layout follows Fig. 14. The deep-editing and adjacent post-edit layers (L9–17) display the largest between-model differences, consistent with the readout-circuitry localisation identified in §4.3.

Takeaway. Across both additional cases, the pattern observed in main-text Fig. 8 holds: fine-tuning leaves the answer token unchanged but *rewires which prompt tokens the model attends to at the deep editing and adjacent post-edit layers*, whereas the early editing layers (L3–8) are largely unchanged. This is the attention-based confirmation of the readout-circuitry claim we make in the abstract and in §4.3, and it complements the MLP-restoration and null-space analyses by establishing the same conclusion at the attention level.

E Hyperparameter Search Procedure

This appendix documents how the continuous hyperparameters of the two recovery procedures of §B.6 were selected. Both UMI and LMI use the same selection criterion (the product $\mathrm{EA} \times \mathrm{Tr.}$ defined in §B.7), but differ in dimensionality: UMI has a single scalar α, whereas LMI has a (μ, σ) pair. For LMI we adopt a two-stage protocol: a one-time coarse 2D sweep that identifies the appropriate bandwidth σ for each architecture family and downstream regime, followed by a per-row fine 1D μ sweep at the fixed σ. The two-stage design lets us amortise the 2D cost across many rows of the main result tables while still capturing the per-row variation in μ^*.

E.1 UMI: α sweep

For every model×method×scale×downstream cell, we sweep α over the eleven values $\{0.0, 0.1, \ldots, 0.9, 1.0\}$, where the endpoints recover the post-FT and post-edit checkpoints, re-

spectively. We then pick $\alpha^* = \arg\max_\alpha \mathrm{EA}(\alpha) \times \mathrm{Tr.}(\alpha)$ and report the row corresponding to α^* in Tab. 6–9. The α^* values concentrate in a narrow band (0.4–0.8 across all 64 rows), and shift mildly with edit scale and downstream task as reported in §C.2.

E.2 LMI Stage 1: Coarse 2D (μ, σ) sweep

The first stage of LMI hyperparameter search establishes the operational range of σ. We perform a 2D sweep on a single representative setting (GPT-J + MEMIT @ 10^2, ZsRE-SFT) over the grid $\mu \in \{5.5, 7.0, 8.5, 10.0, 11.5\}$ and $\sigma \in \{2, 3, 4, 6\}$, plus the UMI anchors at $\alpha \in \{0.3, \ldots, 0.7\}$ for cross-method calibration. Each (μ, σ) point produces one merge plus one full-set evaluation, for a total cost of 20 evaluations $\approx$ 5 GPU-h on H200.

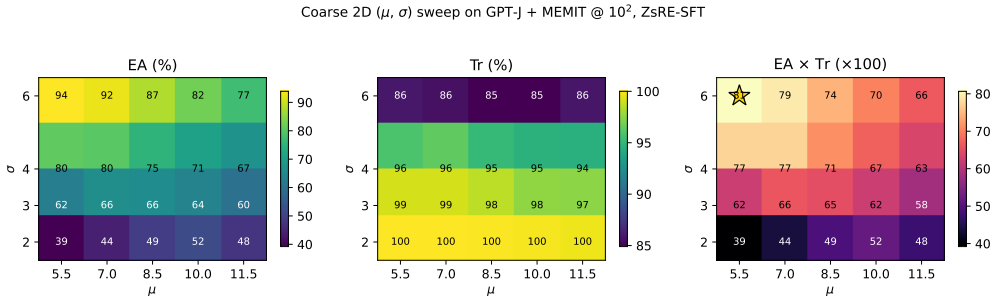


Figure 16: LMI Stage 1: coarse 2D (μ, σ) sweep on GPT-J + MEMIT @ 10^2 followed by ZsRE-SFT. Cells show EA (left), downstream Tr. (middle), and the selection criterion EA $\times$ Tr. (right, multiplied by 100 for display). The gold star marks the joint argmax at $(\mu^*, \sigma^*) = (5.5, 6)$. The sweep cleanly separates the two failure modes: small σ leaves the readout under-restored (EA < 50), whereas large σ over-restores and damages downstream Tr.

Findings from Stage 1 (Fig. 16). The two diagonals are monotonic in opposite directions: EA increases with σ (a broader Gaussian re-injects more edit weight into the readout circuitry), while Tr. decreases with σ (a broader Gaussian sacrifices more of the fine-tuned MLP capacity). The selection criterion peaks at $(\mu^*, \sigma^*) = (5.5, 6)$ with EA $\times$ Tr. $= 80.7$, on the boundary between the deep editing layers (L3–8) and the adjacent post-edit layers (L9–14) identified in §4.3 as the readout circuitry. The optimal σ^* sits in the upper end of the swept range, suggesting that σ should be on the order of the readout-band half-width (about 5 layers for GPT-J).

Translating Stage 1 to per-setting σ. We do not re-run the 2D sweep for every cell in the main result tables. Instead, Stage 1 establishes the rule of thumb $\sigma \approx$ readout-band half-width, which we then apply across all 12 model$\times$method$\times$downstream combinations using the readout bands localised in §4.3 and Appendix A. The resulting per-setting σ values are listed in Tab. 5: σ=4–6 for GPT-J (readout half-width $\sim$ 4 layers, since the readout band is L6–14), σ=3 for GPT-J under LoRA (narrower since LoRA's perturbation is itself narrower), σ=14–20 for Llama-3 (readout half-width $\sim$ 5–6 layers, but the SciQ setting calls for a broader bandwidth that recruits more of the MLP stack).

E.3 LMI Stage 2: Fine 1D μ sweep at fixed σ

With σ fixed by Stage 1, we sweep μ over integer layer indices on the range listed in Tab. 5. For GPT-J this is $\mu \in \{0, 1, \ldots, 14\}$; for Llama-3 we restrict the lower end to $\mu \geq 4$ to keep the kernel centred inside or above the edit layers L4–8. Per-row μ^* is again chosen by $\arg\max_\mu$ EA $\times$ Tr. and reported in Tab. 10–13. The cost is at most 15 evaluations per row, i.e. $\approx$ 3–4 GPU-h on H200, comparable to the UMI search budget of $\approx$ 2–3 GPU-h per row.

Findings from Stage 2 (Fig. 17). Three structural patterns emerge across the ten swept settings. First, the EA curve is unimodal (or monotonically saturating) in μ for every panel; this validates the assumption that the readout circuitry is spatially compact and that locating its centre with a single scalar μ is well-posed. Second, μ^* shifts downstream under LoRA fine-tuning (e.g., GPT-J μ^*=7–9

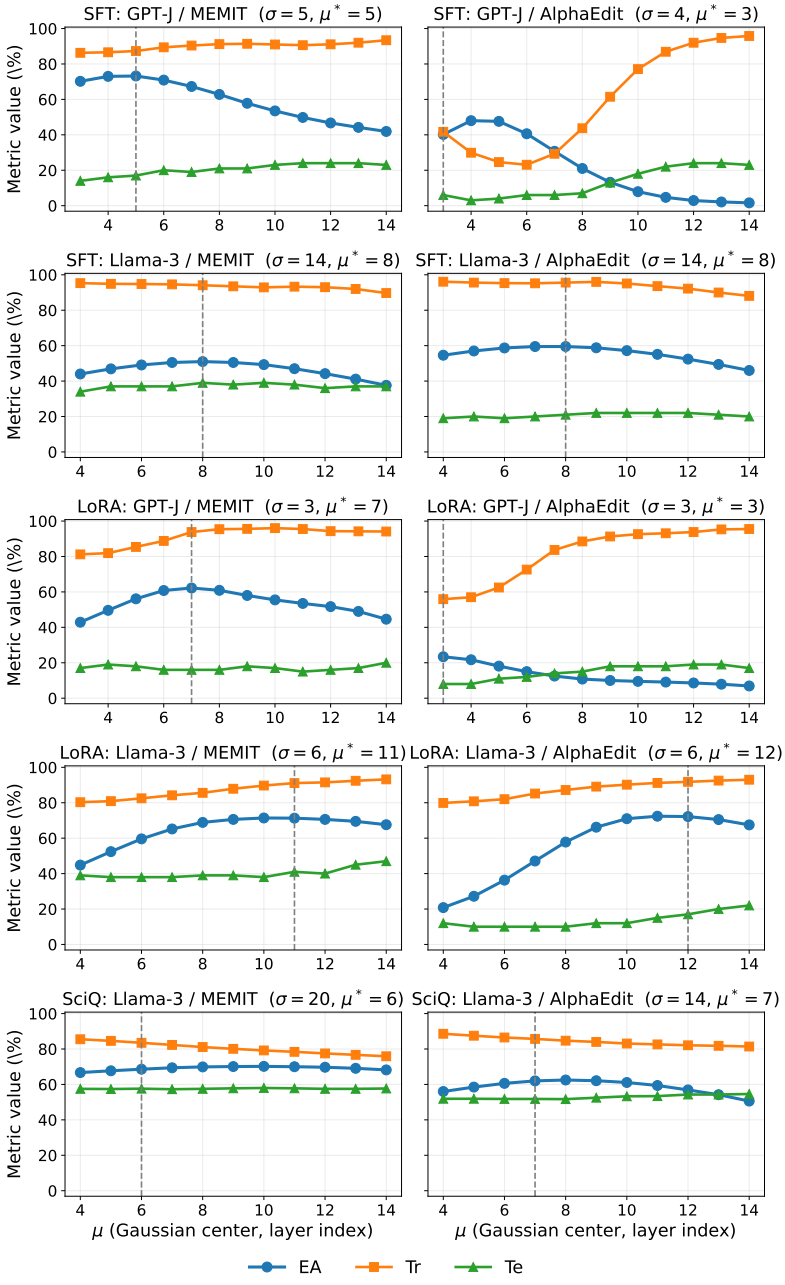


Figure 17: LMI Stage 2: fine 1D μ sweep at the per-setting fixed σ, parsed from the Slurm logs of the μ-sweep scripts. Each panel shows EA (blue), Tr. (orange), and held-out Te. (green) at the @Full edit scale; the grey dashed line marks the μ^* selected by $\arg\max$ EA $\times$ Tr.. The μ curves are predominantly unimodal in μ, and the selected μ^* concentrates inside the readout band identified in §4.3 (GPT-J $\mu^* \in \{3, \ldots, 7\}$; Llama-3 $\mu^* \in \{6, \ldots, 12\}$).

under LoRA vs. μ^*=3–6 under SFT), consistent with LoRA's MLP-only perturbation being more concentrated in the adjacent post-edit layers. Third, μ^* shifts downstream on Llama-3 relative to GPT-J (e.g., μ^*=10–12 for Llama-3 SFT vs. 3–6 for GPT-J SFT), reflecting Llama-3's wider 32-layer stack and the correspondingly larger absolute index of its readout band.

E.4 Why a product objective?

We use $\mathrm{EA} \times \mathrm{Tr.}$ rather than, e.g., $\mathrm{EA} + \mathrm{Tr.}$ or EA alone, because the product penalises configurations that excel on one axis at the cost of collapsing the other. Inspecting Fig. 16 makes this concrete: the EA-only argmax is at $(\mu, \sigma) = (5.5, 6)$ with EA 93.9 but Tr. 85.9, while the Tr.-only argmax is anywhere on the σ=2 row with Tr. $\geq$ 99.6 but EA below 52. The product objective $\mathrm{EA} \times \mathrm{Tr.}$ recovers the same $(5.5, 6)$ as EA-only because Tr. is already close to 86% there, but on harder settings it pushes the selection toward configurations that retain more downstream capacity. The additive $\mathrm{EA} + \mathrm{Tr.}$ behaves similarly but is more sensitive to the absolute scale of the two metrics; we did not see qualitatively different sweet-spots under the additive form in pilot runs and therefore report the product version throughout.

F SVD Verification of the Null-Space Finding

The null-space analysis in §4.2 measures how much of fine-tuning's update ΔW couples with the edit signal via a single direction ($\Delta W \cdot k_{\text{ed}}$), and concludes that the coupling is small. A natural concern is that this direction-specific probe could understate the disruption if fine-tuning moved the weights in some other principal direction of the edit. This appendix verifies the null-space finding from a complementary, basis-free angle: we decompose $\Delta W_{\text{edit}} = W_{\text{ed}} - W_0$ with an SVD and check how strongly the fine-tuning update $\Delta W_{\text{ft}} = W_{\text{ft}} - W_{\text{ed}}$ projects onto each of its top singular directions.

F.1 Setup

For each edit layer $\ell \in \{3, \ldots, 8\}$ on GPT-J, we form the difference matrices $\Delta W_{\text{edit}}^{(\ell)}$ and $\Delta W_{\text{ft}}^{(\ell)}$ at the MLP output matrix `transformer.h.{ℓ}.mlp.fc_out.weight`, compute the SVD $\Delta W_{\text{edit}}^{(\ell)} = U \Sigma V^\top$, and then report three quantities: (i) the global cosine similarity $\cos(\Delta W_{\text{edit}}^{(\ell)}, \Delta W_{\text{ft}}^{(\ell)})$ as a flat vectorised similarity; (ii) the *edit survival ratio*, the projection of $\Delta W_{\text{edit}}^{(\ell)} + \Delta W_{\text{ft}}^{(\ell)}$ along the edit direction normalised by $\|\Delta W_{\text{edit}}^{(\ell)}\|$, which is 1 if fine-tuning is exactly orthogonal to the edit; and (iii) for each of the top-10 singular directions u_k, the normalised projection $|\langle \Delta W_{\text{ft}}^{(\ell)}, u_k \rangle| / \sigma_k$. The setting is the MEMIT-edited GPT-J checkpoint used as the running example in §4.2, fine-tuned with MLP-only SFT on ZsRE at lr$= 6 \times 10^{-5}$; results are nearly identical at the other learning rates we tested (last column of Tab. 14).

F.2 Findings

ΔW_{edit} is not low-rank, but ΔW_{ft} still misses every one of its principal directions. Fig. 18(a) shows that the cumulative energy of ΔW_{edit} in its top-10 singular directions saturates at only 24–32%, with the top-1 direction capturing 6–12%. The edit is therefore spread across many directions rather than concentrated in a few, and any direction-specific probe could in principle miss most of it. Despite this, Fig. 18(b) shows that $|\langle \Delta W_{\text{ft}}, u_k \rangle| / \sigma_k$ stays below 10^{-3} for every top-10 direction at every layer, two orders of magnitude smaller than the 1% reference line that would indicate a non-trivial coupling. Fine-tuning is therefore not just orthogonal to the edit along a single hand-picked direction, but orthogonal to its entire top-10 principal subspace.

Edit survival ratio > 1: fine-tuning slightly reinforces, rather than erases, the edit. The middle column of Tab. 14 shows that the survival ratio sits in 1.014–1.017 across all six layers and in 1.015–1.016 when averaged over all six learning rates we tested. A value of exactly 1 would mean ΔW_{ft} is perfectly orthogonal to ΔW_{edit}; the small positive deviation indicates that fine-tuning's update, when projected along the edit direction, points slightly *toward* the edit rather than against it. This makes the EA collapse documented in §4.1 even more striking: the parameter-space edit not only survives, but it is also mildly amplified, while the strict-EA probe collapses from 100% to under 30%. The

Table 14: SVD analysis of ΔW_{edit} and ΔW_{ft} at each GPT-J edit layer (MEMIT @ ~ 20 edits, ZsRE-SFT lr=6×10^{-5}). `cos`: global cosine similarity between the two flattened update matrices. `Surv.`: edit survival ratio (Sec. F.1); values close to 1 indicate that fine-tuning is essentially orthogonal to the edit. `Top-1 %`: cumulative energy of the largest singular value of ΔW_{edit}. `Top-10 %`: cumulative energy of the top-10 singular values. `Max FT proj.`: maximum of $|\langle \Delta W_{\text{ft}}, u_k \rangle|/\sigma_k$ over $k = 1, \ldots, 10$.

Layer	cos	Surv.	Top-1 %	Top-10 %	Max FT proj.
L3	-0.0005	1.0151	6.9%	29.3%	7×10^{-4}
L4	-0.0009	1.0153	6.1%	24.8%	9×10^{-4}
L5	-0.0004	1.0142	6.3%	24.3%	5×10^{-4}
L6	-0.0003	1.0167	6.3%	24.7%	4×10^{-4}
L7	-0.0007	1.0165	5.6%	24.7%	8×10^{-4}
L8	-0.0005	1.0172	12.1%	32.4%	4×10^{-4}
Robustness across learning rates (layer-averaged):					
lr= 2×10^{-5}	-0.0005	1.0156	—	—	—
lr= 4×10^{-5}	-0.0005	1.0159	—	—	—
lr= 6×10^{-5}	-0.0005	1.0158	—	—	—
lr= 8×10^{-5}	-0.0004	1.0157	—	—	—
lr= 1×10^{-4}	-0.0005	1.0158	—	—	—
lr= 2×10^{-4}	-0.0004	1.0157	—	—	—

disruption therefore cannot be a weight-space erasure of the edit; it has to occur further downstream, which is exactly what the readout-circuitry account of §4.3 supplies.

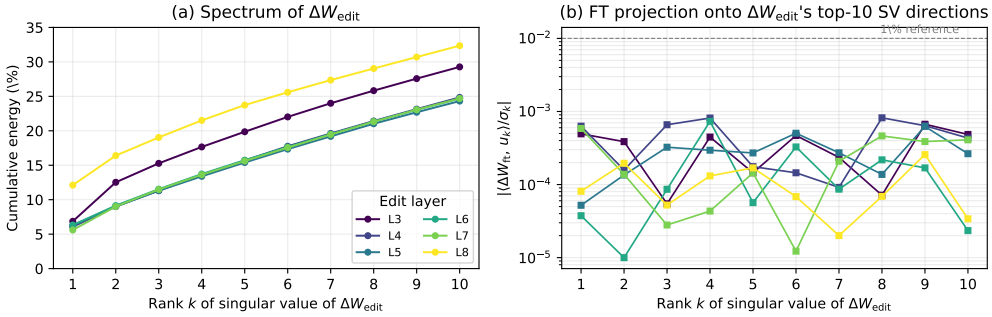


Figure 18: SVD verification of the null-space finding on GPT-J + MEMIT, ZsRE-SFT lr= 6×10^{-5}. **(a)** Cumulative energy in the top-10 singular values of $\Delta W_{\text{edit}}^{(\ell)}$, one curve per edit layer ℓ. The spectrum is broad: the top direction captures only 6–12% and the top-10 subspace captures 24–32%. **(b)** $|\langle \Delta W_{\text{ft}}^{(\ell)}, u_k \rangle|/\sigma_k$ on log scale, for each of those top-10 directions. The grey dashed line marks 1%. All values lie below 10^{-3}, confirming that fine-tuning's update has negligible coupling to every principal direction of the edit.

Connection to §4.2. The null-space probe of §4.2 (Fig. 3a) measures a single quantity (the projection of $\Delta W \cdot \Delta k$ along the edit-key direction) and finds it small. The SVD-based probe in this appendix is basis-free in ΔW_{edit} and finds the same conclusion in a strong form: the coupling is small along *every* top singular direction, the edit's energy is preserved in aggregate (survival > 1), and the result is invariant across all six learning rates we tested. Together, the two analyses give a stronger statement than either alone: fine-tuning's MLP update does not lie in the column span of the edit at all, in any basis we can reasonably define.