RETHINKING COVARIANCE IN KNOWLEDGE EDITING: ESTIMATION, UPDATE GEOMETRY, AND MECHANISMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many locate-then-edit knowledge editing (KE) methods like ROME and AlphaEdit promise efficient factual updates by modifying a small set of model weights, and they rely on precomputed covariance statistics to preserve existing knowledge during parameter updates. However, the cost of estimating these statistics is *generally overlooked* and the mechanisms through which they influence editing *remain poorly understood*. In this paper, we present a systematic investigation connecting covariance estimation, update geometry, and editor-specific sensitivity. We find that extending calibration contexts increases estimation costs but yields no consistent gains in editing performance or general ability, revealing a mismatch between computational overhead and downstream benefits. We further examine how editing performance varies across calibration sequence length distributions. Beyond estimation, we show that covariance defines an anisotropic geometry whose effects on editing depend on how each method translates it into parameter updates. Mechanism analysis identifies update magnitude and directional structure as distinct factors in covariance sensitivity, with their relative importance varying across editors: MEMIT's sensitivity to calibration length is largely attributable to update magnitude, whereas AlphaEdit's behavior depends on the subspace it selects. Together, these findings show that covariance's role in knowledge editing is governed by the interaction between its geometry and the editor's update rule.

1 INTRODUCTION

Large language models (LLMs) store substantial factual and relational knowledge in their parameters, but this knowledge can become outdated, incorrect, or inconsistent with user requirements. Knowledge editing (KE) aims to update specific model behaviors without retraining the entire model, and is often advertised as a lightweight alternative to full finetuning or continual training. Among existing approaches, *locate-then-edit* methods have shown strong performance on standard knowledge-editing benchmarks and are widely used. Representative methods such as ROME [19], MEMIT [20], and AlphaEdit [8] modify a small set of model weights to write new knowledge into the model. These methods perform targeted parameter updates using closed-form solutions that incorporate precomputed statistics to limit interference with unrelated knowledge.

This precomputed statistic is the covariance, commonly stored as the `mom2` statistic. It is estimated from hidden representations collected on a calibration corpus, typically Wikipedia. In current locate-then-edit editing methods, such as ROME and MEMIT, covariance enters the update rule and influences how the editor balances updating the new association against preserving pre-existing knowledge representations. Although the exact use of covariance differs across editors, it directly affects the parameter update produced during editing. In practice, however, covariance construction is separated from the edit pipeline: the statistic is computed beforehand, cached, and then reused across editing requests. This separation hides a substantial preprocessing cost. Estimating covariance requires forwarding calibration sequences through the model and accumulating layer-wise second moments, which can incur substantial computation and GPU memory costs [11]. This preprocessing can be substantially more demanding than the lightweight edit step that follows, yet its cost is rarely considered when knowledge editing methods are usually described as computationally efficient.

Beyond this overlooked preprocessing cost, how covariance affects editing behavior is also not well understood. Existing locate-then-edit methods largely follow the same covariance-estimation

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.

procedure, while differing in how the resulting covariance enters their update rules. But the cost of obtaining covariance and the editing benefit it provides have rarely been studied together. It remains unclear whether the substantial preprocessing cost required to estimate covariance is justified by its downstream benefit to editing. Assessing whether this cost is worthwhile requires understanding how covariance affects editing performance in the first place, and how changes in covariance are reflected in the parameter update actually applied to the model. Because different editors incorporate covariance through different update rules, the same change to covariance can produce very different changes in the resulting parameter update. This leads to the central question of this work:

> *how does covariance affect knowledge editing, from its estimation cost to the parameter updates produced by different editors?*

In this paper, we present a systematic study of covariance in KE, progressing from the relation between estimation cost and editing performance, to how covariance changes affect the geometry of the realized parameter updates, and finally to the mechanisms that explain different sensitivities across methods. We reveal that *covariance does not affect editing merely because the estimation budget changes; it matters only when an editor transforms that change into realized update*.

We first investigate covariance estimation from a joint cost-performance perspective. We employ the maximum calibration length L as proxy, which controls both resource requirements and which calibration tokens enter the statistic. Results in §4.1 show that longer contexts increase computation and memory costs without consistently improving editing performance or general ability. Dataset analysis in §4.2 explains this mismatch through a corpus dominated by short samples, where increasing L adds relatively few tokens. To move beyond correlation, we conduct controlled comparisons in §4.3 that hold corpus identity fixed and match the sequence lengths to Wikipedia's length distribution at each tested L; the resulting changes in the response to L support a contribution of the calibration-length distribution. Experiments across corpora, models, and editors corroborate this finding: longer calibration contexts do not reliably improve performance beyond the default setting.

We next investigate how covariance shapes realized parameter updates in §5 . We intervene on covariance scale (§5.1), spectrum (§5.2), and directional structure (§5.3) in turn, and find that covariance does not constrain editing uniformly, but defines an anisotropic geometry in which the importance of a direction depends on both its variance and its representation basis. Changing relative eigenvalue weights changes performance even when the trace and the eigenvectors are held fixed, and eigenvalue permutation and covariance rotation reduce performance despite preserving the spectrum. In §5.4, alignment between realized and reference updates tracks performance more closely than inverse-key alignment. Returning to calibration-length comparisons in §5.5, we find that differences between MEMIT's cumulative updates concentrate in later edited layers at both editing checkpoints, whereas AlphaEdit maintains nearly identical update directions across calibration lengths.

Finally, we investigate the mechanisms behind method-dependent behaviors in §6. For MEMIT, uniform residual allocation improves performance and reduces calibration-length sensitivity. Norm matching under the original allocation rule also recovers performance, indicating that update magnitude accounts for a substantial part of sensitivity in §6.1. Cross-length differences concentrate in later edited layers, but swapping one edited down-projection weight at a time does not transfer the donor checkpoint's behavior in §6.2. For AlphaEdit, scale and fixed-projector controls link scale sensitivity to the selected subspace in §6.3. Together, these results show that covariance scale and directional structure affect editing through their interaction with each method's update rule, which determines how they shape the magnitude and direction of updates. Our main contributions are:

- *Estimation.* We characterize the cost-performance mismatch in covariance estimation, showing that longer contexts increase cost without reliably improving performance. Further experiments across corpora and models show that length distribution substantially contributes to this response.
- *Update geometry.* We show that how covariance shapes realized parameter updates, through controlled interventions on its scale, spectrum, and directional structure. We connect it to editing performance and reveal method-dependent patterns in direction, magnitude, and layer-wise drift.
- *Mechanisms.* We show how covariance sensitivity depends on the interaction between covariance geometry and the editor's update rule, and how these two factors shape the parameter updates.

Method	Objective	Constraint	Closed-form update
ROME [19]	$\min_{\hat{W}} \|\hat{W}K_0 - W_0K_0\|_F^2$	$\hat{W}k_e = v_e$	$\Delta = \frac{(v_e - W_0k_e)\, k_e^\top C_0^{-1}}{k_e^\top C_0^{-1} k_e}$
MEMIT [20]	$\min_{\hat{W}} \lambda\|\hat{W}K_0 - W_0K_0\|_F^2 + \|\hat{W}K_E - V_E\|_F^2$	Soft memorization; no hard constraint	$\Delta = (V_E - W_0K_E)\, K_E^\top S_\lambda^{-1}$
EMMET [10]	$\min_{\hat{W}} \|\hat{W}K_0 - W_0K_0\|_F^2$	$\hat{W}k_i^e = v_i^e, i = 1, \dots$	$\Delta = (V_E - W_0K_E)\, G_E^{-1} K_E^\top C_0^{-1}$
r-ROME [9]	Same as ROME	Same as ROME	Same as ROME, but using the homogeneous key k_e consistently
AlphaEdit [8]	$\min_{\Delta P} \|(W_0 + \Delta P)K_E - V_E\|_F^2 + \|\Delta P\|_F^2 + \|\Delta P K_p\|_F^2$	Preserve K_0: $(\Delta P)K_0 = 0$	$\Delta P = RK_E^\top P A^{-1}$

For all of methods, $\hat{W} = W_0 + \Delta$; AlphaEdit applies the effective update ΔP. K_0 denotes preserved keys, with covariance $C_0 = K_0K_0^\top$. (k_e, v_e) is a single edit and (K_E, V_E) denotes batched edits. For compactness, $S_\lambda = \lambda C_0 + K_EK_E^\top$, $G_E = K_E^\top C_0^{-1} K_E$, $A = K_pK_p^\top P + K_EK_E^\top P + I$, and $R = V_E - W_0K_E$.

Table 1: Optimization objectives, constraints, and closed-form updates of locate-then-edit methods.

2 Preliminaries

Knowledge Editing. Let f_θ be an LLM with parameters θ. A KE is specified by an input prompt x_e and a desired target response y_e. The goal is to make edited model $f_{\theta'}$ that produces the target response on the edit prompt or paraphrases, while preserving the model's behavior on unrelated inputs. This paper focuses on *locate-then-edit* methods, which modify a small set of feed-forward modules in the transformer. Following prior work, we view the edited projection as an associative memory: a linear map W maps a key representation k to a value representation v. For one edited layer, let $\widehat{W} = W_0 + \Delta$ denote the edited weight, where W_0 is the original weight and Δ is the edit update. For a batch of edit requests, we collect edit keys and target values as $K_E = [k_1^e, \dots, k_E^e]$ and $V_E = [v_1^e, \dots, v_E^e]$. We also collect preserved keys from a calibration corpus as $K_0 = [k_1^0, \dots, k_N^0]$ and $C_0 = K_0K_0^\top$. We refer to C_0 as the covariance or `mom2` statistic. See B for more details.

Covariance For Knowledge Editing. Locate-then-edit methods balance two goals: memorizing the new association and preserving the original model behavior. The preservation term penalizes changes to the output of the edited weight on preserved keys: $\|\widehat{W}K_0 - W_0K_0\|_F^2 = \|\Delta K_0\|_F^2 = \mathrm{tr}\big(\Delta K_0K_0^\top\Delta^\top\big) = \mathrm{tr}\big(\Delta C_0\Delta^\top\big)$. Thus, C_0 defines the metric under which weight updates are penalized. Directions with large variance in the preserved key distribution are costly to perturb, while low-variance directions are less constrained. In this sense, C_0 acts as a preservation geometry. A broad class of locate-then-edit methods can be viewed through the following preservation–memorization template: $\min_\Delta \mathcal{L}_{\text{edit}}(W_0+\Delta; K_E, V_E) + \lambda\ \mathrm{tr}\big(\Delta C_0\Delta^\top\big)$, where $\mathcal{L}_{\text{edit}}$ enforces the new association either as a soft loss or as a hard constraint. The role of C_0 differs across editing methods, but in all cases it controls how the update interacts with preserved model behavior. For example, the ROME-style single-edit update can be written as $\Delta(C) = r_e \frac{k_e^\top C^{-1}}{k_e^\top C^{-1} k_e}$, where $r_e = v_e - W_0k_e$. This expression shows that the update is controlled not by C directly, but by the inverse-preconditioned key $C^{-1}k_e$. For batched equality-constrained editing, the analogous object is $D_C = K_E^\top C^{-1} K_E$, the edit-key Gram matrix under the inverse covariance geometry. Table 1 summarizes the optimization objectives and closed-form updates under the unified notation.

3 Experimental Setups

Language Models. We conduct experiments on various LLMs, including Llama-3.1-8B-Instruct, Llama-3.1-8B-Base, Llama-3.2-3B-Instruct, and Qwen-2.5-7B-Instruct [32].

Editing Datasets and Methods. We employ CounterFact [19] and ZsRE [14] as editing datasets in all experiments. To comprehensively investigate the impact, we compared multiple located-then-edit method: ROME [19], MEMIT [20], AlphaEdit [8], EMMET [10], and R-ROME [9].

General Abilities Evaluation Benchmarks. To systematically evaluate the general abilities of edited language model, we employ the following widely adopted benchmarks: GSM8K [5], GPQA [24], BigBench-Hard (BBH) [28], CommonSense-QA (CSQA) [29], and MMLU [13].

Editing Settings. All edits are performed under *single sample sequential editing settings* without post-edit weight rollback for fair comparison. We evaluate both small and large edit regimes. We use the ES (Editing Success), GE (Generalization Efficacy), PS (Portability Success), and NS (Neighborhood Specificity) measure target-token accuracy on edit prompts, target-token accuracy on rephrased prompts, accuracy on portability queries, and prediction agreement on locality queries, respectively.

4 Covariance Estimation: Cost and Editing Performance

Covariance estimation incurs a preprocessing cost before any knowledge edit is made, and it is determined by the estimation pipeline rather than by the edits themselves. To connect the cost with editing performance, we examine the estimation pipeline through the maximum calibration length L, which we treat as the concrete estimation-budget parameter of the pipeline. We focus on L because it controls both sides of the trade-off: it sets the resources required to accumulate the second moment, and it determines which calibration positions enter the statistic. We then test whether increasing L improves editing, and examine whether the results depend on the calibration data and on the model.

4.1 Longer Covariance Estimation Context May Not Be Better

Setups. For each model, editor, and edit count, we compare covariance estimation context with maximum sequence length $L \in \{512, 1024, 2048\}$. We use the editing overall score as the edit metric and define general ability as the average of BBH, CSQA, GPQA, GSM8K, and MMLU; Table 19 reports both editing and general abilities of Llama-3-8B on Wikipedia across covariance lengths.

Findings. In general, a longer calibration context does not provide a better covariance: shorter estimates are often competitive with the longest available estimate, and increasing L from 512 to 2048 provides no systematic benefit. We report GPU memory in Table 3. Specifically, the effect is editor-dependent: AlphaEdit is nearly invariant to length, and ROME and r-ROME are also length-invariant in many large-edit settings, although their editing scores are already low there. In contrast, EMMET and MEMIT are notably more sensitive, and in these regimes a longer covariance can even be worse. For example, for Qwen2.5-7B with MEMIT at 500 edits, moving from $L = 512$ to $L = 2048$ reduces editing by 10.21 points and general ability by 2.54 points. These results suggest that the calibration length should not be treated as a monotonic indicator of quality.

4.2 Why Longer Calibration Contexts Add Cost but Limited Signal

How Can We Analyze This Phenomenon? To check that this insensitivity is not an implementation defect, we directly compare covariance at different lengths. For layer l and two calibration lengths L_i and L_j, we compute the cosine distance $d_l(L_i, L_j)$. As shown in Figure 1, covariance with different lengths are highly similar within the same layer. We hypothesize this similarity largely comes from the calibration data distribution. The covariance statistic is computed on the `wikipedia/20200501.en` training split with `sample_size=100000`, `random_sample=1`, and sampler `FixedRandomSubsetSampler(seed=1, end=100000)`.

Dataset Level Explanation. As presented in Table 17, dataset is dominated by short samples: under the Llama tokenizer, 65.1% of samples are no longer than 512 tokens, and only 6.5% exceed 2048 tokens; Qwen's tokenizer shows the similar pattern. Since the covariance accumulates token-level second moments, increasing L mainly appends tail tokens from a small number of long documents, while most prefixes are shared across all lengths. Thus, a longer truncation changes the number of included positions where the calibration positions that enter the second moment, much less than the nominal maximum length suggests. This data-level explanation also clarifies the cost–performance mismatch. Although $L = 4096$ adds only about 13.7% additional included positions over $L = 2048$

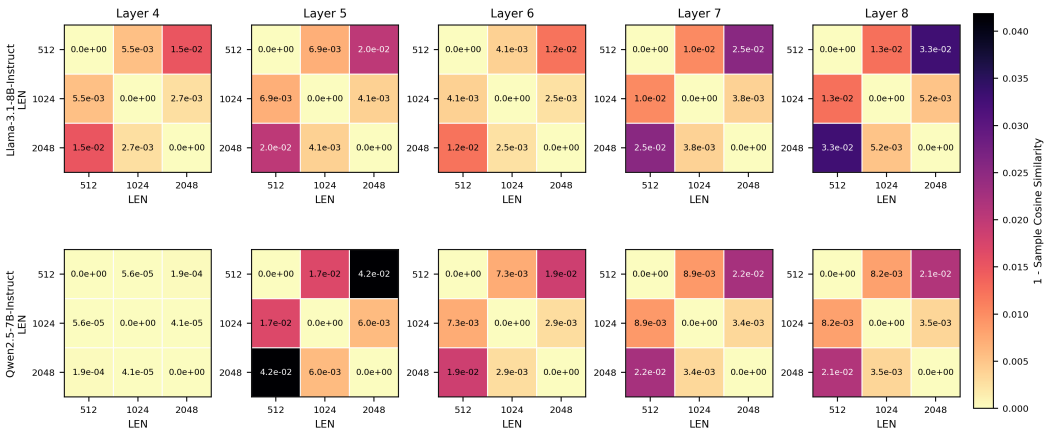

Figure 1: Within-layer cosine distance between covariance estimates computed with different sequence lengths. Small off-diagonal values indicate that different truncation lengths produce highly similar covariance matrices within the same layer.

Calibration set	Median	q_{75}	≤ 512 (%)	> 2048 (%)
C4-native	249	530	73.94	0.00
C4-match	314	719	65.37	0.00
PG19-native	2048	2048	0.00	0.00
PG19-match	314	719	65.37	6.46

Table 2: Lengths of the 100,000 materialized calibration sequences under the Llama tokenizer, before truncation to the tested L. Native sequences and C4-match are capped at 2048; PG19-match retains longer sequences. The two matched sets have identical per-sequence lengths after truncation at every tested $L \leq 2048$. These are sequence lengths, not source-document lengths.

for Llama and 14.2% for Qwen, it can still create much larger memory pressure. This analysis suggests that the response to L may depend on the effective distribution of calibration positions rather than on the nominal maximum length alone. We therefore examine this hypothesis in the following.

4.3 What Determines the Response to Calibration Length?

Section 4.2 suggests that the response to L may depend on the distribution of calibration positions rather than on the nominal maximum length alone. This observation does not by itself separate the effects of corpus content, sequence-length distribution, and sequence construction. We therefore use three comparisons with different purposes: within-corpus native–matched comparisons to test the contribution of length distribution, fixed-L construction comparisons to test whether L specifies the covariance statistic, and additional models to assess the scope of the effect.

Within-corpus length-distribution controls. We first compare two calibration constructions for C4 and PG19. In the *native* setting, passages follow the corpus-specific sampling procedure and length distribution. In the *matched* setting, passages are sampled from the same corpus but their lengths are matched to the corresponding Wikipedia reference distribution at each tested L. Thus, the native–matched comparison changes the calibration-length distribution while holding the corpus identity fixed. This is a control for the contribution of length distribution, although it does not isolate length distribution from every other aspect of the sampling. See Table 2 for length distribution.

The response to L changes substantially between the native and matched constructions in both corpora, but not in a common direction. For AlphaEdit on C4, the native editing scores at $L \in \{512, 1024, 2048\}$ are 5.98, 51.29, and 56.80, whereas the matched scores are 5.98, 62.57, and 29.56. On PG19, the corresponding scores are 11.40, 54.92, and 10.74 in the native setting and 33.50, 9.19, and 12.60 in the matched setting. Neither produces a monotonic improvement with L.

Our Findings. As can be seen from the Table 14, the PG-19 native dataset is predominantly composed of long documents, whereas PG19 match, C4-native, C4-match, and the default Wikipedia all consist of relatively short documents. When L varies, performance on the PG-19 native dataset changes markedly, while other datasets remains largely stable. This experiment demonstrates two points: (1) the data distribution of the dataset is indeed the reason why the model's downstream performance is insensitive to changes in L; and (2) our findings can generalize beyond a specific dataset.

Sequence construction at fixed nominal length. We next vary how calibration sequences are constructed while keeping the nominal maximum length fixed. The resulting in Table 15 changes in editing performance show that L is not a sufficient description of the calibration statistic. However, this comparison changes several properties jointly, including token order, document boundaries, and padding. It therefore identifies sequence construction as an additional source of covariance variation. We leave the details in §C.3 for reference.

Dependence on model and editor. Finally, we repeat the length comparison on Llama-3.2-3B-Instruct and Llama-3.1-8B Base, in addition to the Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct with the same setting in Table 16. The response remains the similar pattern. For example, AlphaEdit is nearly flat across lengths on the 3B model, with scores of 64.14, 62.93, and 63.32, whereas corrected EMMET on the 8B base model decreases from 37.46 at $L = 512$ to 32.42 at $L = 2048$. These results extend the scope of the observation that a longer calibration context does not reliably improve editing, but they do not imply a universal response across models or editors.

Taken together, these comparisons show that the nominal maximum length L is not a sufficient proxy for the useful information and computing overhead in covariance estimation. This motivates the next section, which moves to how an editor transforms those differences into realized parameter updates.

5 Update Geometry: From Covariance to Realized Updates

Section 4 shows that more costly covariance estimation does not consistently improve editing. We therefore ask which properties of the covariance actually shape the updates applied to the model. Writing the reference covariance as $C_\star = U\Lambda U^\top$ separates three properties: its global scale, eigenvalue spectrum, and directional structure. We intervene on each property in turn and trace the resulting changes in the realized updates. Total results across different model, methods, and subsection are summarized in Table 21, 22, and 23. We leave detailed results and discussion in §D.

5.1 Scale: Overall Covariance Magnitude

We begin with the global scale of the covariance, that is, the magnitude of C rather than its directional structure. Rescaling the reference covariance as $C_s = sC_\star$ multiplies all eigenvalues by the same positive factor s, and therefore preserves both their ratios and the eigenvectors. To test whether changing the overall covariance strength alone is sufficient to alter editing performance, we vary s from 10^{-3} to 10^3 while holding the edit requests and all other settings fixed.

Results. Scale affects the editors differently as shown in Table 21, 22 , and 23. ROME remains close to its reference performance for $s = 10$–1000, whereas scales below one produce nonfinite updates. MEMIT exhibits a non-monotonic response: increasing the scale initially improves both writing and locality, with its score rising from 0.1569 to 0.5941 at $s = 10$, before larger scales begin to suppress writing. EMMET shows the opposite trend, as increasing the scale reduces both its update norm and editing performance; at $s = 10$, its score decreases from 0.7523 to 0.5619. AlphaEdit remains relatively stable over moderate scales but produces no update at $s = 1000$ because its projector selects no directions. Overall, covariance scale affects editors through different parts of update computations.

5.2 Spectrum: Relative Directional Weights

We next ask whether changing these weights matters when the trace is held fixed. These weights form the spectrum, that is, the eigenvalues in $C_\star = U\Lambda U^\top$, where the columns of U are the corresponding eigenvectors. Unlike analysis in [33], We construct $C_\gamma = \alpha_\gamma U\Lambda^\gamma U^\top$, where α_γ matches the reference trace, so that the relative weighting is varied while the overall magnitude is held fixed. Values $\gamma < 1$ flatten the spectrum, $\gamma > 1$ sharpen it, and $\gamma = 0$ gives equal weight to every direction. We test $\gamma \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5\}$, using the reference covariance for $\gamma = 1$.

Results. Flattening the spectrum substantially degrades the inverse-based editors in the main experiment, as in Table 21. At $\gamma = 0$, ROME falls from 0.761 to 0.0979, EMMET from 0.7523 to 0.1591, and MEMIT to zero. Moderate sharpening is less harmful and can improve MEMIT: at $\gamma = 1.25$, its score increases to 0.3628, while ROME and EMMET remain close to their references. AlphaEdit is comparatively stable, with scores between 0.7216 and 0.7596. Thus, editing depends not only on the total covariance magnitude but also on how that magnitude is distributed across directions.

5.3 Directional Structure: Anisotropy and Eigen-Direction Alignment

A fixed set of eigenvalues can assign different weights to the same representation directions. Directional structure refers to this assignment as well as to the orientation of the eigenvectors. We therefore ask whether the reference spectrum remains useful once its association with representation directions is altered, and compare the reference against three alternatives: a trace-matched isotropic matrix, the coordinate-diagonal matrix $\mathrm{diag}(C_\star)$, and an eigenvalue-permuted covariance $C_\pi = U\Lambda_\pi U^\top$, in which Λ_π is a permutation of Λ. These interventions respectively remove directional preferences, and change the association between eigenvalues and eigenvectors.

Permutation Results. As is illustrated in Table 21, the full covariance outperforms all three alternatives for every editor. The diagonal covariance partially preserves performance, reaching 0.4621 for ROME and 0.5688 for EMMET, but remains below their reference scores of 0.7610 and 0.7523. Eigenvalue permutation is considerably more damaging, reducing the two scores to 0.0697 and 0.1647. MEMIT is the most sensitive to these interventions, whereas AlphaEdit remains relatively stable. These results show that coordinate-wise variances retain some useful information, but the assignment of covariance weights to directions is also important.

Rotation Results. We additionally rotate the covariance, $C_\theta = R_\theta C_\star R_\theta^\top$, to change its eigenvector orientation while preserving the spectrum. Empirical results reveal that editing performance decreases sharply under rotation: at 30°, Llama Base ROME falls from 0.7380 to 0.0226, while Qwen's results on MEMIT fall from 0.7397 to 0.0077. Together with the permutation results, this shows that the reference covariance is useful not only because of its eigenvalues, but also because those eigenvalues are aligned with particular representation directions. Overall, covariance does not impose a uniform constraint over the editing space. Instead, it defines an anisotropic geometry where the importance of an update direction depends on both the associated variance and representation basis.

5.4 From Covariance Changes to Realized Updates

The interventions above change different properties of the same covariance, yet the responses vary considerably across editors. We therefore trace these changes through the update computation and ask whether editing performance is more closely tracked by intermediate inverse-key quantities or by the final weight change. We measure the 4 metric: *inverse-key alignment* $\cos(C^{-1}k, C_\star^{-1}k)$, where k denotes the edit key, and the *scalar leverage* $h_C(k) = k^\top C^{-1}k$, alongside the *update norm* and the *update alignment* $a_\Delta(C) = \cos(\Delta(C), \Delta_\star)$ defined above.

Results. Update alignment tracks editing performance more closely than the inverse-key diagnostics. Its Spearman correlation with score is 0.897 across all 51 non-reference conditions and 0.966 across the spectrum and directional conditions, compared with 0.340 and 0.587 for inverse-key alignment. This pattern is shared across the four editors, although the dominant update property differs by method. MEMIT is affected more strongly by update magnitude. Its update norm has a correlation of -0.784 with score, compared with 0.506 for update alignment. For example, at $s = 2$ on Llama Base, the update norm decreases to 0.5665 of the reference while the score increases from 0.1172 to 0.5746. Thus, covariance differences are best understood through the final updates they produce, with update direction and magnitude contributing differently across editors.

5.5 Layer-wise Distribution of Update Differences

The global update diagnostics summarize changes across all edited layers but do not reveal where these differences occur. Returning to the calibration-length comparisons in Section 4, we examine whether the differences are spread across layers or concentrated in a few of them. We compare cumulative updates after 500 and 1,000 sequential edits to assess whether their layer-wise distribution persists as editing continues. Let Δ_l^L be the cumulative update at layer l under calibration length L,

with the edit count held fixed. We measure its squared difference from the $L = 2048$ reference as $d_l = \|\Delta_l^L - \Delta_l^{2048}\|_F^2$, and report the layer's share $m_l = d_l / \sum_j d_j$. This separates the location of cross-length differences from the distribution of update magnitude itself.

Results. Results are presented in Table 20. MEMIT's cross-length differences are concentrated in the later edited layers. For $L = 512$ versus $L = 2048$, layers 7–8 account for 91.3% of the difference after 500 edits and 90.9% after 1,000 edits, while layers 4–5 contribute less than 1%. The same pattern appears for $L = 1024$, where layers 7–8 account for 85.5% and 81.4% at the two checkpoints. In contrast, ROME and AlphaEdit produce nearly identical update directions across lengths.

6 Mechanisms of Editor-Specific Covariance Sensitivity

Section 5 shows that covariance changes affect editors through their realized updates. We now investigate the three editor-specific behaviors via mechanistic analysis: MEMIT's allocation of residual updates across layers, higher layer drift in MEMIT, and AlphaEdit's covariance-derived projection geometry. We leave more results and discussion in Appendix E for reference.

6.1 Update Allocation and Magnitude in MEMIT

Hypothesis and Experiments. MEMIT unevenly distributes the residual correction across multiple edited layers, hence we postulate its length sensitivity may arise either from the *allocation rule* or from the *magnitude of the resulting update*. To separate the two, we compare the original *remaining* rule, which divides each residual by the number of remaining edited layers in MEMIT paper, with a *uniform* rule that uses the total number of edited layers, and we add a control that keeps the remaining rule but rescales its proposed layer updates to match the per-request norm of the uniform reference.

Results. As shown in Table 28 and 29, and the remaining rule is both lower-performing and more sensitive to the calibration length: its editing performance decreases from 44.36 at length 512 to 28.72 at length 2048. Uniform allocation narrows this gap to 2.72 points, but it also lowers the final update norm from approximately 87–96 to 33, so it changes the allocation and the update size at the same time. In contrast, matching the update norm while keeping the remaining allocation recovers both performance and length stability, with editing is 72.14, 70.63, and 70.01 at lengths 512, 1024, and 2048. This indicates that update magnitude accounts for a substantial part of length sensitivity.

6.2 MEMIT: Layer-Wise Differences

Hypothesis and experiments. Section 5.5 locates much of MEMIT's cross-length update mismatch in the later edited layers like [16]; here we ask whether these layers also carry its effect on editing. We summarize the archived layer-wise mismatch and exchange one edited down-projection weight between the checkpoints built with lengths 512 and 2048, in both directions for layers 6–8.

Results. As shown in Table 25, the mismatch is concentrated for length 512 relative to length 2048: layer 8 accounts for 66.26% of the squared update mismatch at 500 edits and layers 7–8 jointly for 91.31%, and the shares persist at 1000 edits, at 64.58% and 90.89%. The concentration is somewhat weaker for length 1024, where layers 7–8 account for 85.46% and 81.37%. As presented in Table 30, the swaps change the editing scores across layers and directions, but the donor checkpoint's behavior does not transfer, so they localize where the mismatch sits rather than separate it from the other differences between the two checkpoints.

6.3 AlphaEdit: Projection Geometry, Direction, and Magnitude

Experiments. AlphaEdit uses covariance to construct a projection subspace. We first rescale $C_s = sC_\star$ and compare the original absolute-threshold projector with a fixed-rank projector and a projector frozen at the baseline estimate. We then test whether differences between projectors can be explained by update norm alone, and examine projectors constructed at shared covariance-energy targets. These experiments use Llama-3.1-8B-Instruct and 100 sequential CounterFact edits.

Scale controls. With the original projector construction, ES remains at 100% for scales from 0.1 to 10. At scale 100, the projector retains far fewer directions and ES falls to 19%; at scale 1000, it

selects no directions and applies no update. Fixing the rank removes this scale variation but yields 43% rewrite accuracy. Freezing the baseline projector preserves 100% rewrite accuracy and editing performance of 75.61 across scales. These results connect scale sensitivity to the selected directions.

Norm-matched projectors. We next match, at each request, the joint proposed-update norm across the five edited layers to the absolute-projector reference. The random projectors' mean final update norm falls from 14.2944 to 12.8779, close to the reference 12.8486. Their mean locality score improves from 49.29 to 51.30, but remains below the reference value of 61.23; mean S_4 is nearly unchanged, from 72.82 to 72.85. See Table 34 for full results. Matching the total per-request update norm therefore does not remove the locality difference between projectors.

Common-energy projectors. Finally, we construct same-rank projectors at shared covariance-energy targets. For rank r in dimension d, the target is $E_{\text{target}} = E_{\min} + f\left(\frac{r}{d}\operatorname{tr}(C) - E_{\min}\right)$, where $E_{\min}$ is the minimum energy attainable at rank r and $f \in \{0.25, 0.5\}$. Within each target, the three seeds have the same energy but different projectors. Rewrite accuracy remains at 100% across these conditions, while locality decreases as the target energy increases, as is shown in Table 35.

7 Related Work

Taxonomy of Knowledge Editing Methods. Existing KE methods can be broadly categorized into three paradigms based on their knowledge update mechanism: *(1) Meta-Learning Approaches.* These methods, such as MEND [21] and KE-MAML, utilize hyper-networks to predict weight updates efficiently. They are designed for rapid adaptation but often struggle with the stability required for large-scale sequential editing. *(2) Locate-then-Edit Approaches* rely on identifying specific mechanisms (e.g., Key-Value memories in Feed-Forward Networks) responsible for factual recall. Methods like KN [6], ROME [19], and MEMIT [20] directly modify these identified parameters. Recent advancements like PMET [15] enhance precision by targeting attention heads. *(3) External Memory & Contextual Methods.* Approaches like SERAC [22] and IKE [34] bypass parameter modification by storing edits in an external memory or using in-context learning. While they avoid direct structural modification, they incur inference latency and are often not classified as strict KE.

LLM Evaluation. Evaluating the effectiveness of LLMs [1; 32] involves a diverse array of tests, where models are assessed across various tasks, showcasing their capabilities. Among the benchmarks employed, Bigbench [28], MMLU [13], and HELM [17] are notable. These benchmarks utilize a range of evaluation metrics, such as BLEURT [26] and others requiring meticulous annotation to ensure data quality for downstream applications [23; 18; 30]. Typically, these evaluations emphasize accuracy across multiple choices, which serves as the primary metric [4; 3]. Foundation models, often tested across broad linguistic tasks—both generative and multiple-choice—are analyzed to ensure rigorous assessment standards [25; 13; 31; 27]. These methods, however, may focus on the accuracy within confined options and might not effectively capture the nuances of more open-ended, practical applications where models generate free-form text. Moreover, assessing the safety of LLMs is essential. Safety concentrate on three key areas: truthfulness [2], toxicity [12], and bias [7].

8 Conclusion

In this paper, we investigate covariance in locate-then-edit KE through its estimation cost, update geometry, and interaction with editing methods. We find that longer calibration contexts in covariance estimation increase resource requirements without consistently improving editing performance or general ability. From an update geometric perspective, covariance shapes the magnitude and direction of parameter updates, with consequences that depend on the editor's update rule. Further mechanistic analysis explains the importance of update magnitude for MEMIT and projection subspace selection for AlphaEdit, demonstrating distinct sources of method-dependent covariance sensitivity behaviors.

Practical Implications Although primarily analytical, our study offers practical implications for both covariance estimation and editor design. Shorter calibration contexts can make covariance estimation feasible on workstation GPUs: in general setup, a context length of 4096 exceeds the memory capacity of NVIDIA RTX A6000, whereas shorter contexts remain viable without largely sacrificing performance. Our findings also motivate covariance-aware editor that jointly account for covariance geometry and update rules to better balance knowledge preservation and factual updates.

AI USE STATEMENT

In this work, we used generative AI tools for polishing sentence and checking grammar. We have not used generative AI tools for any other purpose. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[2] Su Lin Blodgett, Gilsinia Lopez, Alexandra Olteanu, Robert Sim, and Hanna Wallach. Stereotyping norwegian salmon: An inventory of pitfalls in fairness benchmark datasets. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1004–1015, 2021.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *ACL*, 2022.

[7] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. Bold: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 862–872, 2021.

[8] Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Xiang Wang, Xiangnan He, and Tat seng Chua. Alphaedit: Null-space constrained knowledge editing for language models, 2024. URL https://arxiv.org/abs/2410.02355.

[9] Akshat Gupta, Sidharth Baskaran, and Gopala Anumanchipalli. Rebuilding rome: Resolving model collapse during sequential model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 21738–21744, 2024.

[10] Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. A unified framework for model editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15403–15418, 2024.

[11] Akshat Gupta, Maochuan Lu, Thomas Hartvigsen, and Gopala Anumanchipalli. Efficient knowledge editing via minimal precomputation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 829–840, 2025.

[12] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 3309–3326, 2022.

[13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[14] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *CoNLL 2017*, pp. 333, 2017.

[15] 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*, volume 38, pp. 18564–18572, 2024.

[16] Xiaopeng Li, Shangwen Wang, Shasha Li, Shezheng Song, Bin Ji, Ma Jun, and Jie Yu. Rethinking residual distribution in locate-then-edit model editing. *Advances in Neural Information Processing Systems*, 38:78348–78374, 2026.

[17] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2022.

[18] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

[19] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual knowledge in GPT. In *NeurIPS*, 2022.

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

[21] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *ICLR*, 2022. URL https://openreview.net/forum?id=0DcZxeWfOPt.

[22] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15817–15831. PMLR, 2022. URL https://proceedings.mlr.press/v162/mitchell22a.html.

[23] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[24] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[25] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.

[26] Thibault Sellam, Dipanjan Das, and Ankur Parikh. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 7881–7892, 2020.

[27] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

[28] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[29] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[30] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

[31] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[32] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[33] Chi Zhang, Mengqi Zhang, Xiaotian Ye, Runxi Cheng, Zisheng Zhou, Ying Zhou, Pengjie Ren, and Zhumin Chen. Spectral characterization and mitigation of sequential knowledge editing collapse. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30009–30032, 2026.

[34] Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? *CoRR*, abs/2305.12740, 2023. doi: 10.48550/arXiv.2305.12740. URL https://doi.org/10.48550/arXiv.2305.12740.

APPENDIX

A Extended Background **14**

A.1 Editing Dataset Details 14

A.2 Editing Methods Details 14

B Detailed Preliminaries **15**

B.1 Knowledge Editing Methods For Large Language Models 15

B.2 Editing Evaluation Metrics 15

B.3 Methods Details 16

C Detailed Results and Discussion for Covariance Estimation **17**

C.1 Estimation Pipeline of Covariance 18

C.2 Details For Why Increasing Context Length Provide Limited Benefit. 19

C.3 Effects of Calibration Corpus Construction 19

C.4 General Ability For Unedited Models 21

C.5 General Ability For Experiments 21

C.6 Main Results 27

C.7 Analysis Table 36

C.8 Extended Discussions 36

D Detailed Results For Update Geometry **38**

D.1 Scale 38

D.2 Spectrum 39

D.3 Directional Structure 39

D.4 From Covariance Changes to Realized Updates 40

D.5 Layer-wise Distribution of Covariance-Induced Update Differences 40

D.6 Main Results 41

E More Details for Mechanic Analysis **46**

E.1 MEMIT: Update Allocation and the Role of Update Magnitude 46

E.2 Update Alignment and Functional Localization of Layer-wise Drift 47

E.3 AlphaEdit: Projection Construction Determines Scale Sensitivity 48

E.4 Main Results 49

F Implementation and Reproduction Details **52**

F.1 Code Base 52

F.2 Models 52

F.3 Hyperparameter Settings 52

F.4 Running Devices 52

A Extended Background

A.1 Editing Dataset Details

ZsRE [14] is a question-answering (QA) dataset that uses questions generated through back-translation as equivalent neighbors. Following previous work, natural questions are used as out-of-scope data to evaluate locality. Each sample in ZsRE includes a subject string and answers as the editing targets to assess editing success, along with the rephrased question for generalization evaluation and the locality question for evaluating specificity.

Counterfact [19] is a more challenging dataset that contrasts counterfactuals with factual statements, initially scoring lower for Counterfact. It constructs out-of-scope data by replacing the subject entity with approximate entities sharing the same predicate. The Counterfact dataset has similar metrics to ZsRE for evaluating efficacy, generalization, and specificity. Additionally, Counterfact includes multiple generation prompts with the same meaning as the original prompt to test the quality of the generated text, specifically focusing on fluency and consistency.

A.2 Editing Methods Details

Here we introduce the all of editing methods employed in this study. **For the hyperparameter settings of the baseline methods, we used the code provided in the Easyedit framework for reproduction.**

- **MEND** [21] is a method for efficiently editing large pre-trained models using a single input-output pair. MEND utilizes small auxiliary networks to make fast, localized changes to the model without full retraining. By applying a low-rank decomposition to the gradient from standard fine-tuning, MEND enables efficient and tractable parameter adjustments. This approach allows for post-hoc edits in large models while avoiding the overfitting common in traditional fine-tuning methods.

- **ROME** [19] is a method for updating specific factual associations in LLMs. By identifying key neuron activations in middle-layer feed-forward modules that influence factual predictions, ROME modifies feed-forward weights to edit these associations directly. ROME demonstrates that mid-layer feed-forward modules play a crucial role in storing and recalling factual knowledge, making direct model manipulation a viable editing technique.

- **MEMIT** [20] is a scalable multi-layer update algorithm designed for efficiently inserting new factual memories into transformer-based language models. Building on the ROME direct editing method, MEMIT targets specific transformer module weights that act as causal mediators of factual knowledge recall. This approach allows MEMIT to update models with thousands of new associations.

- **PMET** [15] is a knowledge editing method, which simultaneously optimizes transformer hidden states of MHSA and FFN, utilizing solely the optimized transformer hidden states of FFN as the target knowledge representations for updating FFN weights, enabling precise updates.

- **AlphaEdit** [8] is a knowledge editing method that aims to update specific knowledge while preserving existing knowledge. Unlike traditional methods that struggle with balancing knowledge retention and modification, AlphaEdit introduces a null-space-constrained approach. It projects the perturbations onto the null space of preserved knowledge before applying updates, ensuring that previously stored knowledge remains intact. This technique prevents model forgetting and overfitting in sequential editing scenarios. AlphaEdit significantly improves editing performance (by 36.4% on average) across various LLMs, such as LLaMA3 and GPT-2 XL. It seamlessly integrates into existing editing frameworks like MEMIT and ROME, making it a simple yet highly effective solution for updating LLMs without disrupting their overall knowledge structure.

B Detailed Preliminaries

B.1 Knowledge Editing Methods For Large Language Models

Large language models store a substantial amount of factual knowledge in their parameters. Given an edit request, knowledge editing aims to modify a model so that it reflects a new target fact while preserving unrelated behavior. Formally, let f_θ be an autoregressive language model with parameters θ. An edit request can be written as

$$e = (x_e, y_e), \tag{1}$$

where x_e is an edit prompt and y_e is the desired target response. The goal is to obtain an edited model $f_{\theta'}$ such that

$$f_{\theta'}(x_e) \approx y_e, \tag{2}$$

while keeping the model's behavior close to the original model f_θ on unrelated inputs.

A successful edit should satisfy several desiderata. First, it should have high *efficacy*: the edited model should produce the new target answer on the edit prompt. Second, it should generalize to semantically equivalent prompts, often referred to as *paraphrase generalization*. Third, it should preserve unrelated knowledge, often measured by *neighborhood specificity* or *locality*. Finally, the edited model should maintain its general language modeling ability and downstream task performance.

This work focuses on locate-then-edit methods, which first identify a small set of model components that mediate a factual association, and then directly modify the corresponding parameters. In transformer language models, many such methods edit the feed-forward network at one or more selected layers. Following prior work, we view the edited feed-forward projection as an associative memory: a linear map W maps a key vector k to a value vector v. For a single edited layer, we write the original weight as

$$W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{key}}}, \tag{3}$$

and the edited weight as

$$\widehat{W} = W_0 + \Delta, \tag{4}$$

where Δ is the weight update induced by editing.

For a set of edit requests, let

$$K_E = [k_1^e, \ldots, k_E^e] \in \mathbb{R}^{d_{\text{key}} \times E} \tag{5}$$

denote the edit keys, and let

$$V_E = [v_1^e, \ldots, v_E^e] \in \mathbb{R}^{d_{\text{out}} \times E} \tag{6}$$

denote the target values that the edited model should associate with these keys. Similarly, let

$$K_0 = [k_1^0, \ldots, k_N^0] \in \mathbb{R}^{d_{\text{key}} \times N} \tag{7}$$

denote a set of preserved keys collected from a calibration corpus. These keys represent directions on which the edited model should behave similarly to the original model.

The central object studied in this paper is the second-moment statistic

$$C_0 = K_0 K_0^\top. \tag{8}$$

In implementations, this quantity is often estimated as an uncentered second moment over hidden keys from a calibration corpus. We refer to it as the covariance or `mom2` statistic throughout the paper. Constant normalization factors, such as $1/N$, are omitted because they can be absorbed into method-specific hyperparameters or cancel out in the closed-form updates of equality-constrained methods.

When multiple layers are edited, all quantities above can be indexed by layer l, for example W_l, $K_{0,l}$, and $C_{0,l}$. For clarity, we omit the layer index unless necessary.

B.2 Editing Evaluation Metrics

Current evaluation primarily focuses on whether the model can effectively recall edited knowledge, whether the edits cause side effects, and whether these edits can generalize within the editing domain (paraphrase). The post-edit model M_θ should satisfy the following three properties: **reliability**, **generalization**, and **specificity**.

Reliability Previous works like ROME[19] define a reliable edit when the post-edit model M_θ^e gives the target answer for the knowledge tuple $t = (s, r, o)$ to be edited. If we denote prompt $p(s, r)$ that embed s and r as x, o as y. The reliability is measured as the average accuracy on the edit case:

$$\mathbb{E}_{x'_\mathrm{e}, y'_\mathrm{e} \sim \{(x_\mathrm{e}, y_\mathrm{e})\}} \mathbb{1} \left\{ \mathrm{argmax}_y f_{\theta_e} \left(y \mid x'_\mathrm{e} \right) = y'_\mathrm{e} \right\} \tag{9}$$

Generalization The post-edit model f_{θ_e} should also edit the equivalent neighbour $N\left(x_\mathrm{e}, y_\mathrm{e}\right)$ (e.g. rephrased sentences). It is evaluated by the average accuracy of the model f_{θ_e} on examples drawn uniformly from the equivalence neighborhood:

$$\mathbb{E}_{x'_\mathrm{e}, y'_\mathrm{e} \sim N(x_\mathrm{e}, y_\mathrm{e})} \mathbb{1} \left\{ \mathrm{argmax}_y f_{\theta_e} \left(y \mid x'_\mathrm{e} \right) = y'_\mathrm{e} \right\} \tag{10}$$

Specificity also noted as **Locality** in some work. Editing should be implemented locally, which means the post-edit model f_{θ_e} should not change the output of the irrelevant examples in the out-of-scope $O(x_e, y_e)$. Hence, the locality is evaluated by the rate at which the post-edit model f_{θ_e}'s predictions are unchanged as the pre-edit f_θ model:

$$\mathbb{E}_{x'_\mathrm{e}, y'_\mathrm{e} \sim O(x_\mathrm{e}, y_\mathrm{e})} \mathbb{1} \left\{ f_{\theta_e} \left(y \mid x'_\mathrm{e} \right) = f_\theta \left(y \mid x'_\mathrm{e} \right) \right\} \tag{11}$$

B.3 Methods Details

Locate-then-edit methods can be understood as balancing two goals: *memorization* of the new edit and *preservation* of the original model behavior. The preservation objective penalizes changes to the output of the edited weight on preserved keys:

$$\|\widehat{W} K_0 - W_0 K_0\|_F^2 = \|\Delta K_0\|_F^2. \tag{12}$$

Using $C_0 = K_0 K_0^\top$, this can be rewritten as

$$\|\Delta K_0\|_F^2 = \mathrm{tr}(\Delta C_0 \Delta^\top). \tag{13}$$

Thus, C_0 defines the metric under which weight updates are penalized. Directions with large variance in the preservation keys are penalized more strongly, while directions weakly represented in the preservation corpus are penalized less. In this sense, C_0 acts as a *preservation geometry* for knowledge editing.

Different editing methods instantiate memorization and preservation in different ways. Some impose the new association as a hard constraint, while others use a soft least-squares objective. Below, we summarize the optimization objectives of ROME, r-ROME, MEMIT, and EMMET under a common notation.

ROME. ROME performs a single factual edit by finding the minimum-change update that preserves the behavior of W_0 on K_0, subject to exactly writing a new key-value association:

$$\min_{\widehat{W}} \|\widehat{W} K_0 - W_0 K_0\|_F^2 \quad \text{s.t.} \quad \widehat{W} k_e = v_e. \tag{14}$$

Equivalently, in terms of $\Delta = \widehat{W} - W_0$,

$$\min_{\Delta} \|\Delta K_0\|_F^2 \quad \text{s.t.} \quad \Delta k_e = v_e - W_0 k_e. \tag{15}$$

Let

$$r_e = v_e - W_0 k_e \tag{16}$$

be the residual that must be written into the layer. The closed-form update is

$$\Delta_{\mathrm{ROME}} = r_e \frac{k_e^\top C_0^{-1}}{k_e^\top C_0^{-1} k_e}. \tag{17}$$

This expression shows that ROME does not use C_0 merely as a scalar penalty. Instead, the update direction is shaped by the inverse covariance geometry $C_0^{-1} k_e$.

r-ROME. r-ROME preserves the same optimization objective and closed-form update as ROME, but corrects the inconsistent use of the edit key in the original implementation. In our notation, r-ROME uses the same key k_e consistently in both the constraint and the closed-form update:

$$\Delta_{\mathrm{rROME}} = r_e \frac{k_e^\top C_0^{-1}}{k_e^\top C_0^{-1} k_e}. \tag{18}$$

Therefore, in this paper, r-ROME is treated as the implementation-corrected version of the ROME objective. This distinction is important because our goal is to study the effect of the covariance statistic itself, rather than artifacts caused by inconsistent key construction.

MEMIT. MEMIT extends single-edit updates to mass editing by using a soft memorization objective. Given a batch of edit keys K_E and target values V_E, MEMIT solves

$$\min_{\widehat{W}} \lambda \|\widehat{W} K_0 - W_0 K_0\|_F^2 + \|\widehat{W} K_E - V_E\|_F^2, \tag{19}$$

where $\lambda > 0$ controls the strength of preservation relative to memorization. In terms of Δ, this becomes

$$\min_{\Delta} \lambda \|\Delta K_0\|_F^2 + \|(W_0 + \Delta) K_E - V_E\|_F^2. \tag{20}$$

Let

$$R = V_E - W_0 K_E \tag{21}$$

be the residual matrix for the edit batch. The closed-form update is

$$\Delta_{\mathrm{MEMIT}} = R K_E^\top \left(\lambda C_0 + K_E K_E^\top\right)^{-1}. \tag{22}$$

Unlike equality-constrained methods such as ROME and EMMET, the global scale of C_0 does not necessarily cancel in MEMIT, because it changes the effective trade-off between preservation and memorization through the term λC_0.

EMMET. EMMET can be viewed as a batched equality-constrained counterpart to ROME. It preserves the original model behavior on K_0, while requiring all edit associations in the batch to be written exactly:

$$\min_{\widehat{W}} \ \|\widehat{W} K_0 - W_0 K_0\|_F^2 \quad \text{s.t.} \quad \widehat{W} K_E = V_E. \tag{23}$$

Equivalently,

$$\min_{\Delta} \ \|\Delta K_0\|_F^2 \quad \text{s.t.} \quad \Delta K_E = R. \tag{24}$$

The closed-form update is

$$\Delta_{\mathrm{EMMET}} = R \left(K_E^\top C_0^{-1} K_E\right)^{-1} K_E^\top C_0^{-1}. \tag{25}$$

The matrix

$$D_C = K_E^\top C_0^{-1} K_E \tag{26}$$

is the edit-key Gram matrix under the inverse covariance geometry. Its conditioning affects the stability of the batched update, making it a useful diagnostic for understanding edit interference.

C Detailed Results and Discussion for Covariance Estimation

The preservation covariance summarizes second-order statistics of pre-existing keys and shapes the weight update so that new knowledge can be written while limiting changes to preserved representations. Its construction, however, requires forwarding calibration sequences through the model and accumulating second moments for the relevant layers. This preprocessing can be substantially more demanding than the subsequent closed-form editing step, especially for long sequences and high-dimensional feed-forward representations.This raises the first question of our study: *does the additional cost of covariance estimation translate into better editing performance?* We first characterize the estimation pipeline and its memory requirements. Maximum calibration length L is a natural and important parameter for studying this cost–performance trade-off because it affects both the resource requirements of covariance construction and the set of calibration representations that can enter the

Model / representation	$L = 512$	1024	2048	4096
Analytical Overhead				
Llama-3.1-8B, $d = 14{,}336$	1.70	1.86	2.19	2.84
Qwen2.5-7B, $d = 18{,}944$	2.89	3.11	3.54	4.41
Real Memory				
Llama-3.1-8B, $d = 14{,}336$	18.6GB	24.3GB	37.6GB	OOM
Qwen2.5-7B, $d = 18{,}944$	16.7GB	23.1GB	33.5GB	OOM

Table 3: Analytical covariance-specific float32 working set (GiB) and real memory, recomputed from Eq. equation 28. These are not measured GPU peaks and exclude the model forward pass.

statistic. These two effects need not scale together. A larger L increases the sequence and feature sizes that the estimation pipeline must support, while the amount of additional calibration content depends on the underlying document-length distribution. We then test whether longer calibration contexts improve editing, examine how much additional calibration data those contexts actually contribute, and investigate the roles of corpus choice, sequence construction, and model setting. Throughout, we distinguish the configured maximum length from the effective data entering the statistic.

C.1 Estimation Pipeline of Covariance

We examine the uncentered second-moment statistic used by the locate-then-edit implementations in our study. For an edited layer ℓ, let $\mathcal{T}_L$ be the token positions included by the calibration pipeline under maximum sequence length L, and let $k_{\ell,t} \in \mathbb{R}^{d_\ell}$ denote the representation collected at the input to the edited feed-forward projection. The estimator is proportional to

$$\widehat{C}_{\ell,L} \;\propto\; \sum_{t \in \mathcal{T}_L} k_{\ell,t} k_{\ell,t}^\top. \tag{27}$$

The proportionality notation accommodates the normalization used when the statistic is saved; this convention is held fixed when comparing lengths, and its possible interaction with editor-specific update rules is examined in Section 4. Our default Wikipedia pipeline uses the `wikipedia/20200501.en` training split, `sample_size=100000`, `random_sample=1`, and `FixedRandomSubsetSampler(seed=1, end=100000)`. Holding this subset fixed isolates truncation length from a change in sampled documents. The implementation accumulates features at the input to `mlp.down_proj` and sets the batch token budget to `batch_tokens=3L`.

Memory accounting. The representation dimension is $d = 14{,}336$ for Llama-3.1-8B-Instruct and $d = 18{,}944$ for Qwen2.5-7B-Instruct in the configurations examined. If the second-moment accumulator and the matrix-product temporary coexist in float32, and two feature-sized buffers of at most $3Ld$ elements are retained, an approximate covariance-specific working-set budget is

$$M_{\text{cov}}(L, d) \;\simeq\; 4\big[2d^2 + 2(3L)d\big] \text{ bytes.} \tag{28}$$

Table 3 evaluates this expression. The $O(d^2)$ matrix storage is substantial even at short lengths; the feature-buffer term increases linearly with L. The table is an *analytical allocation estimate*, not a measurement of device-wide peak memory. In particular, it excludes model weights, forward activations, attention temporaries, the CUDA allocator, and any retained caches. Actual peak memory and wall-clock time depend on the execution schedule, precision, and hardware, and must be reported separately if used for a quantitative speed or hardware claim.

These costs matter because increasing L changes more than the number of useful calibration tokens. It also changes the maximum sequence and batch shapes encountered during covariance construction, and can increase forward-pass memory pressure. Thus, nominal length is an imperfect proxy for statistical information but remains an important determinant of worst-case resource demand. We next ask whether the resulting estimates produce measurably better edited models.

C.2 Details For Why Increasing Context Length Provide Limited Benefit.

Computing of covariance. This is because, in the covariance code, the token budget is set to `batch_tokens` $= 3L$ and the statistic is accumulated over the input to `mlp.down_proj`. The corresponding feature dimension d is large: $d = 14336$ for Llama-3.1-8B-Instruct and $d = 18944$ for Qwen2.5-7B-Instruct. The covariance working set is therefore roughly $2d^2 + 4 + 2(3L)d - 4$ bytes in float32, accounting for the `mom2` matrix that stores the unnormalized outer-product sum, the $a^\top$ temporary matrix, retained layer inputs, and flattened features. Table 9 reports this extra covariance-specific peak memory. The count covers the covariance-specific buffers only, and excludes model weights, forward activations, attention tensors, CUDA allocator overhead, and the KV cache. With fp32 model loading, the total $L = 4096$ memory footprint can approach or exceed the limit of 48GB GPUs, even though its statistical contribution to the covariance is modest.

Estimator normalization. For an estimator normalized by the number of included positions, when truncation preserves all earlier positions and their representations, the relationship can be written as

$$\widehat{C}_{\ell,L'} = \frac{T_L}{T_{L'}}\widehat{C}_{\ell,L} + \frac{T_{L'} - T_L}{T_{L'}}\widehat{C}_{\ell,\text{tail}}, \qquad L' > L, \tag{29}$$

where T_L is the effective position count and $\widehat{C}_{\ell,\text{tail}}$ summarizes newly included positions. This identity describes the comparable-prefix case, not arbitrary changes in packing, padding, or batch construction. In the default sample, most short documents contribute exactly the same content at different truncation lengths; additional observations are concentrated in the tails of longer documents. At the same time, the resource budget in Section 4 must accommodate longer maximum sequences. This provides a concrete explanation for why nominal estimation cost and additional statistical information can grow at different rates.

C.3 Effects of Calibration Corpus Construction

We investigate how the organization of calibration text affects the preservation second moment, $C = \mathbb{E}[\mathbf{k}\mathbf{k}^\top]$, and subsequent knowledge editing performance. We construct six corpus variants from the same pool of 100,000 Wikipedia articles sampled from the `20200501.en` training split with sampling seed 1. Each variant is evaluated at maximum sequence lengths $L \in \{512, 1024, 2048\}$. Table 4 summarizes the transformations.

The original baseline and Prefix use the same sampling and per-document truncation rules. Their distinction is operational: the original pipeline tokenizes articles during statistics collection, whereas Prefix materializes token sequences in advance and stores statistics in a separate cache. Given identical source ordering and tokenizer configurations, the two procedures should produce identical token sequences. We nevertheless report their historical runs separately because equivalence of the configurations and caches actually used in those runs has not been fully verified. Prefix serves as the control within the corpus-variant pipeline.

Token budgets. For each maximum length L, we define the reference budget as

$$B_L = \sum_{i=1}^{100{,}000} \left|\text{Encode}_{\le L}(d_i)\right|, \tag{30}$$

where $\text{Encode}_{\le L}$ tokenizes article d_i using the tokenizer's default special-token behavior and truncates the sequence to at most L tokens. The first five variants contain exactly B_L non-padding tokens. The Padding variant instead counts both text and padding tokens toward this budget. Its padding positions have attention mask 1 and therefore contribute to both the forward computation and the second-moment estimate. Retaining the final complete block introduces a small budget overshoot. Table 5 reports the realized counts.

Budget matching holds within each length setting, rather than across lengths: B_L increases with L. Furthermore, equal token counts do not imply equal document coverage, sequence-length distributions, or computational cost. In particular, Padding replaces a substantial fraction of text with padding tokens, reaching 69.51% padding at $L = 2048$. Thus, the length comparisons do not isolate context length at a fixed token budget, and the padding comparison changes both padding exposure and the amount of text. We exclude the supplementary real-token-budget padding condition, for which no editing results are available.

Editing protocol and results. We use Llama-3.1-8B-Instruct and perform 1,000 sequential CounterFact edits with editing seed 42. All requests are evaluated using the final edited model. Table 14 reports editing accuracy (ES), measured as target-token accuracy on the editing prompts, rather than exact match of complete answers. Of the 90 planned variant configurations (six constructions, three lengths, and five methods), 88 have endpoint results. The missing configurations are MEMIT and AlphaEdit with Global-token at $L = 2048$.

Prefix yields higher ES than the other corpus variants for ROME, R-ROME, and EMMET at all three lengths. Global token shuffling generally produces substantial degradation: ROME and R-ROME reach zero ES at $L = 1024$ and $L = 2048$, while AlphaEdit falls to 2.80% at $L = 1024$. AlphaEdit also exhibits a pronounced interaction between construction and length. Within-document sentence shuffling achieves 92.00% ES at $L = 512$, compared with 34.95% for Prefix, but this advantage does not persist at the larger lengths. Padding achieves 98.00% ES for AlphaEdit at $L = 1024$, yet decreases to 86.40% at $L = 2048$.

All available MEMIT endpoints have zero ES. These are degenerate endpoints and do not establish insensitivity to corpus construction. More generally, ES alone does not measure preservation or portability. The results demonstrate sensitivity to the calibration pipeline under the tested settings, but do not isolate semantic coherence as the sole cause: the transformations can also change document coverage, tokenization boundaries, and the amount of text. The single editing seed further limits conclusions about statistical significance.

Table 4: Calibration corpus constructions. All variants use the same sampled Wikipedia article pool. The original baseline follows the same text-construction rule as Prefix but uses a separate preparation and caching pipeline.

Construction	Procedure	Main changes
Prefix	Independently tokenize each article and retain at most its first L tokens. Keep shorter articles at their original length.	Reference construction based on article prefixes.
Sequential	Tokenize each article in its original order and split it into non-overlapping blocks of at most L tokens. Retain short trailing blocks and stop at the token budget.	Includes later article content and changes document coverage and context boundaries.
Within-doc	Shuffle sentences within each article, concatenate their token sequences, and split the result into blocks.	Preserves within-sentence order and document membership while disrupting sentence order.
Global-sentence	Pool sentences from all sampled articles, shuffle them globally, concatenate their token sequences, and split them into blocks.	Preserves within-sentence order while disrupting document boundaries and topical continuity.
Global-token	Concatenate token streams from all sampled articles, shuffle individual tokens globally, and split them into blocks.	Disrupts word order and syntax. This is not uniform sampling from the vocabulary.
Padding	Split articles sequentially and pad short trailing blocks to L. Set padding masks to 1 and count padding toward the budget.	Replaces some text with statistically active padding at an approximately matched total token count.

The corresponding implementation names are `prefix`, `sequential_chunk`, `within_doc_shuffle`, `global_sentence_shuffle`, `fixed_budget_uniform`, and `padded_chunk` with `padbudget`. The sentence-shuffling variants tokenize individual sentences without special tokens, introducing an additional tokenization-boundary difference. Padding uses token ID 128009.

	First five variants	Padding			
L	Text tokens (B_L)	Text tokens	Total tokens	Budget overshoot	Padding fraction
512	31,261,659	21,696,476	31,261,696	37	30.60%
1024	43,656,774	21,717,692	43,657,216	442	50.25%
2048	54,341,543	16,569,537	54,341,632	89	69.51%

"Text tokens" denotes non-padding tokens, including any special tokens introduced by the corresponding encoding procedure. Counts refer to positions included in second-moment estimation; ordinary masked batch padding is excluded. The final complete block causes the small overshoot in Padding. Matching these counts does not imply identical FLOPs or runtime.

Table 5: Realized token budgets. The first five constructions have identical non-padding token counts at each L. Padding counts both text and active padding tokens toward the reference budget B_L.

C.4 General Ability For Unedited Models

C.5 General Ability For Experiments

Table 7: Section 4.2: General ability across calibration lengths on Llama-3.1-8B-Instruct. Scores are percentages after 1,000 sequential CounterFact edits (seed 42). Mean is the unweighted mean of the five benchmarks; Δ is its change from the matching unedited model, in percentage points. An unavailable exact-checkpoint evaluation is shown as –; Invalid denotes a checkpoint with confirmed nonfinite weights. Neither has a reported mean.

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Unedited	–	–	72.0	67.9	50.0	75.6	65.4	66.2	+0.0
Llama-8B-Instruct	ROME	512	0.0	22.0	25.0	17.9	0.0	13.0	-53.2
Llama-8B-Instruct	ROME	1024	0.0	25.0	20.0	21.1	0.0	13.2	-52.9
Llama-8B-Instruct	ROME	2048	0.0	19.0	15.0	16.3	0.0	10.1	-56.1

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Llama-8B-Instruct	R-ROME	512	0.0	28.6	20.0	17.1	0.0	13.1	-53.0
Llama-8B-Instruct	R-ROME	1024	0.0	29.2	45.0	16.3	0.0	18.1	-48.1
Llama-8B-Instruct	R-ROME	2048	0.0	33.9	10.0	17.9	0.0	12.4	-53.8
Llama-8B-Instruct	EMMET	512	–	–	–	–	–	–	–
Llama-8B-Instruct	EMMET	1024	–	–	–	–	–	–	–
Llama-8B-Instruct	EMMET	2048	–	–	–	–	–	–	–
Llama-8B-Instruct	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
Llama-8B-Instruct	MEMIT	1024	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Llama-8B-Instruct	MEMIT	2048	0.0	23.8	35.0	22.8	0.0	16.3	-49.8
Llama-8B-Instruct	AlphaEdit	512	0.0	31.0	30.0	14.6	0.0	15.1	-51.0
Llama-8B-Instruct	AlphaEdit	1024	59.1	61.3	50.0	63.4	55.1	57.8	-8.4
Llama-8B-Instruct	AlphaEdit	2048	0.0	50.6	25.0	56.9	5.1	27.5	-38.6

Model	BBH	CSQA	GPQA	GSM8K	MMLU	Avg.
Qwen2.5-7B-Instruct	53.99	84.85	35.35	83.78	76.79	66.95
Llama-3.1-8B-Instruct	70.48	78.54	34.85	78.39	68.68	66.19
Llama-3.1-8B-Base	58.97	73.98	55.00	56.82	69.05	62.76
Llama-3.2-3B-Instruct	41.03	73.98	35.00	67.42	62.50	55.99

Table 6: General ability of unedited base models. Scores are percentages.

Table 8: Section 4.3: General ability across calibration corpora on Llama-3.1-8B-Instruct. Native and Wikipedia-length-matched corpora are reported separately. External-corpus cache provenance remains unresolved in the source audit. Scores are percentages after 1,000 sequential CounterFact edits (seed 42). Mean is the unweighted mean of the five benchmarks; Δ is its change from the matching unedited model, in percentage points. An unavailable exact-checkpoint evaluation is shown as –; Invalid denotes a checkpoint with confirmed nonfinite weights. Neither has a reported mean.

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Unedited	–	–	72.0	67.9	50.0	75.6	65.4	66.2	+0.0
Wikipedia	ROME	512	0.0	22.0	25.0	17.9	0.0	13.0	-53.2
Wikipedia	ROME	1024	0.0	25.0	20.0	21.1	0.0	13.2	-52.9
Wikipedia	ROME	2048	0.0	19.0	15.0	16.3	0.0	10.1	-56.1
Wikipedia	R-ROME	512	0.0	28.6	20.0	17.1	0.0	13.1	-53.0
Wikipedia	R-ROME	1024	0.0	29.2	45.0	16.3	0.0	18.1	-48.1
Wikipedia	R-ROME	2048	0.0	33.9	10.0	17.9	0.0	12.4	-53.8
Wikipedia	EMMET	512	–	–	–	–	–	–	–
Wikipedia	EMMET	1024	–	–	–	–	–	–	–
Wikipedia	EMMET	2048	–	–	–	–	–	–	–
Wikipedia	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
Wikipedia	MEMIT	1024	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Wikipedia	MEMIT	2048	0.0	23.8	35.0	22.8	0.0	16.3	-49.8
Wikipedia	AlphaEdit	512	0.0	31.0	30.0	14.6	0.0	15.1	-51.0
Wikipedia	AlphaEdit	1024	59.1	61.3	50.0	63.4	55.1	57.8	-8.4
Wikipedia	AlphaEdit	2048	0.0	50.6	25.0	56.9	5.1	27.5	-38.6
C4-native	ROME	512	0.0	31.0	45.0	13.0	0.0	17.8	-48.4
C4-native	ROME	1024	0.0	27.4	25.0	17.1	0.0	13.9	-52.3
C4-native	ROME	2048	0.0	25.0	30.0	14.6	0.0	13.9	-52.2
C4-native	R-ROME	512	0.0	29.2	15.0	16.3	0.0	12.1	-54.1
C4-native	R-ROME	1024	0.0	24.4	20.0	17.9	0.0	12.5	-53.7
C4-native	R-ROME	2048	0.0	28.0	30.0	14.6	0.0	14.5	-51.6
C4-native	EMMET	512	0.0	31.0	25.0	18.7	0.0	14.9	-51.2
C4-native	EMMET	1024	0.0	28.6	30.0	20.3	3.8	16.5	-49.6
C4-native	EMMET	2048	0.0	26.2	30.0	17.9	6.4	16.1	-50.1
C4-native	MEMIT	512	0.0	22.0	35.0	23.6	0.0	16.1	-50.0
C4-native	MEMIT	1024	0.0	22.0	35.0	23.6	0.0	16.1	-50.0
C4-native	MEMIT	2048	0.0	29.8	25.0	15.4	0.0	14.0	-52.1
C4-native	AlphaEdit	512	0.0	23.2	30.0	16.3	0.0	13.9	-52.3
C4-native	AlphaEdit	1024	0.0	32.1	15.0	34.1	5.1	17.3	-48.9
C4-native	AlphaEdit	2048	6.8	45.8	10.0	50.4	9.0	24.4	-41.8
C4-match	ROME	512	0.0	25.6	25.0	17.9	0.0	13.7	-52.5
C4-match	ROME	1024	0.0	29.2	15.0	18.7	0.0	12.6	-53.6
C4-match	ROME	2048	0.0	23.8	20.0	20.3	0.0	12.8	-53.3
C4-match	R-ROME	512	0.0	28.6	20.0	14.6	0.0	12.6	-53.5
C4-match	R-ROME	1024	0.0	31.5	20.0	16.3	0.0	13.6	-52.6
C4-match	R-ROME	2048	0.0	29.8	25.0	15.4	0.0	14.0	-52.1
C4-match	EMMET	512	0.0	23.2	35.0	17.9	2.6	15.7	-50.4
C4-match	EMMET	1024	0.0	26.8	35.0	17.1	2.6	16.3	-49.9
C4-match	EMMET	2048	0.0	26.2	15.0	17.9	7.7	13.4	-52.8
C4-match	MEMIT	512	0.0	22.0	35.0	23.6	0.0	16.1	-50.0
C4-match	MEMIT	1024	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
C4-match	MEMIT	2048	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
C4-match	AlphaEdit	512	0.0	21.4	25.0	23.6	0.0	14.0	-52.2
C4-match	AlphaEdit	1024	58.3	57.7	35.0	67.5	50.0	53.7	-12.5
C4-match	AlphaEdit	2048	0.0	28.6	15.0	15.4	0.0	11.8	-54.4
PG19-native	ROME	512	0.0	25.0	20.0	17.1	0.0	12.4	-53.7
PG19-native	ROME	1024	0.0	25.0	20.0	18.7	0.0	12.7	-53.4
PG19-native	ROME	2048	0.0	28.6	20.0	16.3	0.0	13.0	-53.2
PG19-native	R-ROME	512	0.0	30.4	20.0	13.8	0.0	12.8	-53.3
PG19-native	R-ROME	1024	0.0	28.0	30.0	12.2	0.0	14.0	-52.1
PG19-native	R-ROME	2048	0.0	26.2	20.0	13.8	0.0	12.0	-54.2
PG19-native	EMMET	512	0.0	25.0	30.0	13.8	0.0	13.8	-52.4

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
PG19-native	EMMET	1024	0.0	29.2	25.0	16.3	0.0	14.1	-52.1
PG19-native	EMMET	2048	0.0	34.5	25.0	17.1	0.0	15.3	-50.8
PG19-native	MEMIT	512	0.0	23.8	35.0	17.9	0.0	15.3	-50.8
PG19-native	MEMIT	1024	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
PG19-native	MEMIT	2048	–	–	–	–	–	–	–
PG19-native	AlphaEdit	512	0.0	28.0	25.0	15.4	0.0	13.7	-52.5
PG19-native	AlphaEdit	1024	0.8	32.7	15.0	30.9	19.2	19.7	-46.4
PG19-native	AlphaEdit	2048	–	–	–	–	–	–	–
PG19-match	ROME	512	0.0	30.4	20.0	16.3	0.0	13.3	-52.8
PG19-match	ROME	1024	0.0	27.4	20.0	17.1	0.0	12.9	-53.3
PG19-match	ROME	2048	0.0	24.4	20.0	16.3	0.0	12.1	-54.0
PG19-match	R-ROME	512	0.0	22.6	20.0	15.4	0.0	11.6	-54.6
PG19-match	R-ROME	1024	0.0	30.4	15.0	18.7	0.0	12.8	-53.4
PG19-match	R-ROME	2048	0.0	31.0	20.0	23.6	0.0	14.9	-51.3
PG19-match	EMMET	512	0.0	33.3	25.0	15.4	0.0	14.8	-51.4
PG19-match	EMMET	1024	0.0	23.8	20.0	19.5	0.0	12.7	-53.5
PG19-match	EMMET	2048	0.0	26.8	25.0	17.9	0.0	13.9	-52.2
PG19-match	MEMIT	512	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
PG19-match	MEMIT	1024	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
PG19-match	MEMIT	2048	0.0	22.0	35.0	23.6	0.0	16.1	-50.0
PG19-match	AlphaEdit	512	0.0	29.2	25.0	17.1	0.0	14.2	-51.9
PG19-match	AlphaEdit	1024	0.0	22.6	20.0	16.3	0.0	11.8	-54.4
PG19-match	AlphaEdit	2048	0.0	29.8	35.0	15.4	0.0	16.0	-50.1

Table 9: Section 4.3: General ability across Wikipedia calibration-data constructions on Llama-3.1-8B-Instruct. Scores are percentages after 1,000 sequential CounterFact edits (seed 42). Mean is the unweighted mean of the five benchmarks; Δ is its change from the matching unedited model, in percentage points. An unavailable exact-checkpoint evaluation is shown as –; Invalid denotes a checkpoint with confirmed nonfinite weights. Neither has a reported mean.

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Unedited	–	–	72.0	67.9	50.0	75.6	65.4	66.2	+0.0
Prefix	ROME	512	0.0	21.4	25.0	17.9	0.0	12.9	-53.3
Prefix	ROME	1024	0.0	25.0	20.0	21.1	0.0	13.2	-52.9
Prefix	ROME	2048	0.0	22.6	20.0	18.7	0.0	12.3	-53.9
Prefix	R-ROME	512	0.0	28.6	20.0	17.1	0.0	13.1	-53.0
Prefix	R-ROME	1024	0.0	29.2	45.0	16.3	0.0	18.1	-48.1
Prefix	R-ROME	2048	0.0	28.6	30.0	18.7	0.0	15.5	-50.7
Prefix	EMMET	512	0.0	30.4	30.0	21.1	9.0	18.1	-48.1
Prefix	EMMET	1024	0.0	25.6	45.0	17.1	6.4	18.8	-47.3
Prefix	EMMET	2048	0.0	30.4	30.0	16.3	5.1	16.3	-49.8
Prefix	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
Prefix	MEMIT	1024	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Prefix	MEMIT	2048	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Prefix	AlphaEdit	512	0.0	31.0	30.0	14.6	0.0	15.1	-51.0
Prefix	AlphaEdit	1024	59.1	61.3	50.0	63.4	55.1	57.8	-8.4
Prefix	AlphaEdit	2048	54.5	60.1	45.0	65.9	56.4	56.4	-9.8
Sequential	ROME	512	0.0	26.8	15.0	14.6	0.0	11.3	-54.9
Sequential	ROME	1024	0.0	25.6	20.0	15.4	0.0	12.2	-54.0
Sequential	ROME	2048	0.0	27.4	20.0	13.8	0.0	12.2	-53.9
Sequential	R-ROME	512	0.0	26.2	30.0	19.5	0.0	15.1	-51.0
Sequential	R-ROME	1024	0.0	23.2	20.0	14.6	0.0	11.6	-54.6
Sequential	R-ROME	2048	0.0	20.2	25.0	18.7	0.0	12.8	-53.4
Sequential	EMMET	512	0.0	28.0	25.0	21.1	0.0	14.8	-51.3
Sequential	EMMET	1024	0.0	26.2	30.0	13.8	1.3	14.3	-51.9
Sequential	EMMET	2048	0.0	29.2	20.0	18.7	0.0	13.6	-52.6
Sequential	MEMIT	512	0.0	31.5	35.0	18.7	0.0	17.0	-49.1
Sequential	MEMIT	1024	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Sequential	MEMIT	2048	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Sequential	AlphaEdit	512	0.0	24.4	25.0	18.7	2.6	14.1	-52.0
Sequential	AlphaEdit	1024	0.0	41.7	35.0	48.8	6.4	26.4	-39.8
Sequential	AlphaEdit	2048	0.0	38.7	10.0	30.1	17.9	19.3	-46.8
Within-document	ROME	512	0.0	23.2	25.0	14.6	0.0	12.6	-53.6
Within-document	ROME	1024	0.0	21.4	20.0	17.9	0.0	11.9	-54.3
Within-document	ROME	2048	0.0	24.4	20.0	22.0	0.0	13.3	-52.9
Within-document	R-ROME	512	0.0	26.2	25.0	22.0	0.0	14.6	-51.5
Within-document	R-ROME	1024	0.0	31.5	20.0	16.3	0.0	13.6	-52.6
Within-document	R-ROME	2048	0.0	26.2	20.0	18.7	0.0	13.0	-53.2
Within-document	EMMET	512	0.0	26.2	25.0	22.0	0.0	14.6	-51.5
Within-document	EMMET	1024	0.0	26.2	15.0	21.1	0.0	12.5	-53.7
Within-document	EMMET	2048	0.0	31.5	30.0	14.6	1.3	15.5	-50.7
Within-document	MEMIT	512	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Within-document	MEMIT	1024	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Within-document	MEMIT	2048	0.0	23.8	20.0	17.9	0.0	12.3	-53.8
Within-document	AlphaEdit	512	0.8	39.9	30.0	40.7	15.4	25.3	-40.8
Within-document	AlphaEdit	1024	0.0	19.0	30.0	22.0	12.8	16.8	-49.4
Within-document	AlphaEdit	2048	0.0	41.1	15.0	46.3	12.8	23.0	-43.1
Global-sentence	ROME	512	0.0	28.6	25.0	20.3	0.0	14.8	-51.4
Global-sentence	ROME	1024	0.0	28.6	25.0	19.5	0.0	14.6	-51.5
Global-sentence	ROME	2048	0.0	27.4	20.0	13.8	0.0	12.2	-53.9
Global-sentence	R-ROME	512	0.0	26.8	15.0	17.1	0.0	11.8	-54.4
Global-sentence	R-ROME	1024	0.0	32.1	25.0	16.3	0.0	14.7	-51.5
Global-sentence	R-ROME	2048	0.0	22.0	20.0	18.7	0.0	12.1	-54.0
Global-sentence	EMMET	512	0.0	29.8	20.0	17.9	0.0	13.5	-52.6
Global-sentence	EMMET	1024	0.0	26.8	20.0	23.6	0.0	14.1	-52.1
Global-sentence	EMMET	2048	0.0	19.6	30.0	22.8	0.0	14.5	-51.7
Global-sentence	MEMIT	512	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Global-sentence	MEMIT	1024	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Global-sentence	MEMIT	2048	0.0	29.8	35.0	15.4	0.0	16.0	-50.1
Global-sentence	AlphaEdit	512	0.0	29.8	20.0	14.6	0.0	12.9	-53.3
Global-sentence	AlphaEdit	1024	0.0	29.2	25.0	17.1	0.0	14.2	-51.9
Global-sentence	AlphaEdit	2048	0.0	20.2	30.0	14.6	6.4	14.3	-51.9
Uniform-token	ROME	512	0.0	28.6	20.0	14.6	0.0	12.6	-53.5
Uniform-token	ROME	1024	Invalid	Invalid	Invalid	Invalid	Invalid	–	–
Uniform-token	ROME	2048	Invalid	Invalid	Invalid	Invalid	Invalid	–	–
Uniform-token	R-ROME	512	0.0	25.6	15.0	15.4	0.0	11.2	-55.0
Uniform-token	R-ROME	1024	Invalid	Invalid	Invalid	Invalid	Invalid	–	–
Uniform-token	R-ROME	2048	Invalid	Invalid	Invalid	Invalid	Invalid	–	–

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Uniform-token	EMMET	512	0.0	29.2	20.0	15.4	0.0	12.9	-53.2
Uniform-token	EMMET	1024	0.0	28.6	25.0	15.4	0.0	13.8	-52.4
Uniform-token	EMMET	2048	–	–	–	–	–	–	–
Uniform-token	MEMIT	512	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Uniform-token	MEMIT	1024	–	–	–	–	–	–	–
Uniform-token	MEMIT	2048	–	–	–	–	–	–	–
Uniform-token	AlphaEdit	512	0.0	29.8	25.0	15.4	0.0	14.0	-52.1
Uniform-token	AlphaEdit	1024	–	–	–	–	–	–	–
Uniform-token	AlphaEdit	2048	–	–	–	–	–	–	–
Padded	ROME	512	0.0	25.0	20.0	13.0	0.0	11.6	-54.6
Padded	ROME	1024	0.0	26.8	20.0	14.6	0.0	12.3	-53.9
Padded	ROME	2048	0.0	28.0	25.0	15.4	0.0	13.7	-52.5
Padded	R-ROME	512	0.0	24.4	25.0	18.7	0.0	13.6	-52.5
Padded	R-ROME	1024	0.0	23.2	25.0	22.0	0.0	14.0	-52.1
Padded	R-ROME	2048	0.0	29.8	30.0	19.5	0.0	15.9	-50.3
Padded	EMMET	512	0.0	30.4	20.0	15.4	0.0	13.2	-53.0
Padded	EMMET	1024	0.0	26.8	15.0	18.7	0.0	12.1	-54.1
Padded	EMMET	2048	0.0	25.0	20.0	29.3	0.0	14.9	-51.3
Padded	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
Padded	MEMIT	1024	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Padded	MEMIT	2048	0.0	29.8	40.0	15.4	0.0	17.0	-49.1
Padded	AlphaEdit	512	0.0	26.8	30.0	22.0	0.0	15.7	-50.4
Padded	AlphaEdit	1024	53.8	58.9	40.0	71.5	62.8	57.4	-8.7
Padded	AlphaEdit	2048	3.0	39.9	25.0	46.3	15.4	25.9	-40.2

Table 10: Section 4.3: General ability across models. The Llama-8B-Instruct block repeats Section 4.2 to make the comparison self-contained. Scores are percentages after 1,000 sequential CounterFact edits (seed 42). Mean is the unweighted mean of the five benchmarks; Δ is its change from the matching unedited model, in percentage points. An unavailable exact-checkpoint evaluation is shown as –; Invalid denotes a checkpoint with confirmed nonfinite weights. Neither has a reported mean.

Setting	Editor	L	GSM8K	MMLU	GPQA	CSQA	BBH	Mean	Δ
Llama-8B-Instruct	Unedited	–	72.0	67.9	50.0	75.6	65.4	66.2	+0.0
Llama-8B-Instruct	ROME	512	0.0	22.0	25.0	17.9	0.0	13.0	-53.2
Llama-8B-Instruct	ROME	1024	0.0	25.0	20.0	21.1	0.0	13.2	-52.9
Llama-8B-Instruct	ROME	2048	0.0	19.0	15.0	16.3	0.0	10.1	-56.1
Llama-8B-Instruct	R-ROME	512	0.0	28.6	20.0	17.1	0.0	13.1	-53.0
Llama-8B-Instruct	R-ROME	1024	0.0	29.2	45.0	16.3	0.0	18.1	-48.1
Llama-8B-Instruct	R-ROME	2048	0.0	33.9	10.0	17.9	0.0	12.4	-53.8
Llama-8B-Instruct	EMMET	512	–	–	–	–	–	–	–
Llama-8B-Instruct	EMMET	1024	–	–	–	–	–	–	–
Llama-8B-Instruct	EMMET	2048	–	–	–	–	–	–	–
Llama-8B-Instruct	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-53.0
Llama-8B-Instruct	MEMIT	1024	0.0	22.0	25.0	23.6	0.0	14.1	-52.0
Llama-8B-Instruct	MEMIT	2048	0.0	23.8	35.0	22.8	0.0	16.3	-49.8
Llama-8B-Instruct	AlphaEdit	512	0.0	31.0	30.0	14.6	0.0	15.1	-51.0
Llama-8B-Instruct	AlphaEdit	1024	59.1	61.3	50.0	63.4	55.1	57.8	-8.4
Llama-8B-Instruct	AlphaEdit	2048	0.0	50.6	25.0	56.9	5.1	27.5	-38.6
Llama-8B-Base	Unedited	–	56.1	69.0	55.0	74.0	59.0	62.6	+0.0
Llama-8B-Base	ROME	512	0.0	26.2	30.0	17.9	0.0	14.8	-47.8
Llama-8B-Base	ROME	1024	–	–	–	–	–	–	–
Llama-8B-Base	ROME	2048	0.0	24.4	25.0	14.6	0.0	12.8	-49.8
Llama-8B-Base	R-ROME	512	0.0	28.0	15.0	17.9	0.0	12.2	-50.4
Llama-8B-Base	R-ROME	1024	–	–	–	–	–	–	–
Llama-8B-Base	R-ROME	2048	0.0	30.4	15.0	13.8	0.0	11.8	-50.8
Llama-8B-Base	EMMET	512	–	–	–	–	–	–	–
Llama-8B-Base	EMMET	1024	–	–	–	–	–	–	–
Llama-8B-Base	EMMET	2048	–	–	–	–	–	–	–
Llama-8B-Base	MEMIT	512	0.0	22.0	35.0	23.6	0.0	16.1	-46.5
Llama-8B-Base	MEMIT	1024	–	–	–	–	–	–	–
Llama-8B-Base	MEMIT	2048	0.0	22.0	35.0	23.6	0.0	16.1	-46.5
Llama-8B-Base	AlphaEdit	512	0.0	28.0	35.0	25.2	0.0	17.6	-45.0
Llama-8B-Base	AlphaEdit	1024	–	–	–	–	–	–	–
Llama-8B-Base	AlphaEdit	2048	0.0	22.6	20.0	15.4	0.0	11.6	-51.0
Llama-3B-Instruct	Unedited	–	64.4	62.5	35.0	74.0	41.0	55.4	+0.0
Llama-3B-Instruct	ROME	512	0.0	29.8	30.0	17.9	0.0	15.5	-39.9
Llama-3B-Instruct	ROME	1024	0.0	23.8	25.0	16.3	0.0	13.0	-42.4
Llama-3B-Instruct	ROME	2048	0.0	28.0	30.0	17.9	0.0	15.2	-40.2
Llama-3B-Instruct	R-ROME	512	0.0	26.2	20.0	23.6	0.0	14.0	-41.4
Llama-3B-Instruct	R-ROME	1024	0.0	30.4	30.0	21.1	0.0	16.3	-39.1
Llama-3B-Instruct	R-ROME	2048	0.0	32.1	30.0	14.6	0.0	15.4	-40.0
Llama-3B-Instruct	EMMET	512	0.0	23.8	30.0	24.4	5.1	16.7	-38.7
Llama-3B-Instruct	EMMET	1024	0.0	26.2	15.0	18.7	6.4	13.3	-42.1
Llama-3B-Instruct	EMMET	2048	0.0	31.0	35.0	20.3	9.0	19.1	-36.3
Llama-3B-Instruct	MEMIT	512	0.0	22.0	20.0	23.6	0.0	13.1	-42.3
Llama-3B-Instruct	MEMIT	1024	0.0	26.2	35.0	19.5	0.0	16.1	-39.2
Llama-3B-Instruct	MEMIT	2048	0.0	22.0	35.0	23.6	0.0	16.1	-39.3
Llama-3B-Instruct	AlphaEdit	512	15.9	50.6	40.0	57.7	38.5	40.5	-14.8
Llama-3B-Instruct	AlphaEdit	1024	39.4	56.5	25.0	64.2	37.2	44.5	-10.9
Llama-3B-Instruct	AlphaEdit	2048	28.8	53.0	35.0	61.0	39.7	43.5	-11.9
Qwen-7B-Instruct	Unedited	–	81.1	79.2	55.0	85.4	50.0	70.1	+0.0
Qwen-7B-Instruct	ROME	512	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	ROME	1024	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	ROME	2048	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	R-ROME	512	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	R-ROME	1024	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	R-ROME	2048	0.0	29.8	25.0	15.4	0.0	14.0	-56.1
Qwen-7B-Instruct	EMMET	512	–	–	–	–	–	–	–
Qwen-7B-Instruct	EMMET	1024	–	–	–	–	–	–	–
Qwen-7B-Instruct	EMMET	2048	–	–	–	–	–	–	–
Qwen-7B-Instruct	MEMIT	512	0.0	24.4	30.0	19.5	0.0	14.8	-55.3
Qwen-7B-Instruct	MEMIT	1024	–	–	–	–	–	–	–
Qwen-7B-Instruct	MEMIT	2048	0.0	24.4	35.0	19.5	0.0	15.8	-54.3
Qwen-7B-Instruct	AlphaEdit	512	–	–	–	–	–	–	–
Qwen-7B-Instruct	AlphaEdit	1024	–	–	–	–	–	–	–
Qwen-7B-Instruct	AlphaEdit	2048	–	–	–	–	–	–	–

C.6 Main Results

Setting	Editor	$L = 512$	$L = 1024$	$L = 2048$
Llama-3.1-8B-Instruct	ROME	27.52	25.79	24.88
Llama-3.1-8B-Instruct	R-ROME	31.95	30.25	26.85
Llama-3.1-8B-Instruct	EMMET	40.58	40.05	38.17
Llama-3.1-8B-Instruct	MEMIT	0.01†	0.01†	0.62†
Llama-3.1-8B-Instruct	AlphaEdit	17.80	66.11	59.44
Llama-3.1-8B Base	ROME	23.49	23.87	23.30
Llama-3.1-8B Base	R-ROME	29.76	27.82	27.74
Llama-3.1-8B Base	EMMET	37.46	36.17	32.42
Llama-3.1-8B Base	MEMIT	0.61†	1.58†	0.61†
Llama-3.1-8B Base	AlphaEdit	39.77	52.01	1.46
Llama-3.2-3B-Instruct	ROME	21.73	23.08	22.03
Llama-3.2-3B-Instruct	R-ROME	19.86	22.64	22.87
Llama-3.2-3B-Instruct	EMMET	44.76	43.11	41.17
Llama-3.2-3B-Instruct	MEMIT	0.62†	0.03†	0.03†
Llama-3.2-3B-Instruct	AlphaEdit	64.14	62.93	63.32
Qwen2.5-7B-Instruct	ROME	0.00†	0.00†	0.00†
Qwen2.5-7B-Instruct	R-ROME	0.00†	0.00†	0.00†
Qwen2.5-7B-Instruct	EMMET	65.48	65.74	64.31
Qwen2.5-7B-Instruct	MEMIT	1.29†	–	1.59†
Qwen2.5-7B-Instruct	AlphaEdit	–	–	–

Table 11: Default Wikipedia results across models. $S_4 = (ES + GE + PS + NS)/4$.

Calibration set	T_{512}	T_{1024}	T_{2048}	$512 \to 1024$	$1024 \to 2048$
Wikipedia	31.262	43.657	54.342	39.65%	24.47%
C4-native	28.289	36.416	41.562	28.73%	14.13%
C4-match	31.214	43.553	54.229	39.53%	24.51%
PG19-native	51.200	102.400	204.800	100.00%	100.00%
PG19-match	31.214	43.553	54.229	39.53%	24.51%

Table 13: Included positions (millions) and percentage increases. Wikipedia counts refer to the existing seed-1 main cache; matched sets use seed 42 and are not exactly token-matched to that cache.

Setting	Editor	$L = 512$	$L = 1024$	$L = 2048$
Prefix	ROME	27.52	25.79	27.35
Prefix	R-ROME	31.95	30.25	28.09
Prefix	EMMET	40.58	40.05	38.14
Prefix	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	$0.01^{\dagger}$
Prefix	AlphaEdit	17.80	66.11	65.71
Sequential	ROME	18.24	23.34	24.10
Sequential	R-ROME	23.52	26.61	25.03
Sequential	EMMET	34.36	35.23	28.91
Sequential	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	$0.01^{\dagger}$
Sequential	AlphaEdit	32.95	59.88	50.80
Within-document	ROME	22.95	22.52	21.08
Within-document	R-ROME	24.71	22.10	21.54
Within-document	EMMET	32.66	34.14	31.28
Within-document	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	$0.01^{\dagger}$
Within-document	AlphaEdit	53.38	53.36	61.87
Global-sentence	ROME	17.87	18.75	12.72
Global-sentence	R-ROME	18.50	17.34	16.72
Global-sentence	EMMET	26.01	27.37	20.28
Global-sentence	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	$0.01^{\dagger}$
Global-sentence	AlphaEdit	6.36	8.62	54.42
Uniform-token	ROME	19.60	invalid	invalid
Uniform-token	R-ROME	17.88	invalid	invalid
Uniform-token	EMMET	22.19	20.73	22.77
Uniform-token	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	–
Uniform-token	AlphaEdit	8.60	0.91	–
Padded	ROME	23.17	22.97	19.74
Padded	R-ROME	26.70	24.82	25.89
Padded	EMMET	31.76	26.57	25.57
Padded	MEMIT	$0.01^{\dagger}$	$0.01^{\dagger}$	$0.01^{\dagger}$
Padded	AlphaEdit	8.43	62.97	49.65

Table 12: Wikipedia sequence-construction results on Llama-3.1-8B-Instruct. Construction changes multiple properties at once. Here $S_4 = (ES + GE + PS + NS)/4$;

Setting	Editor	L	ES	GE	PS	NS	S_4	GA
C4-native	ROME	512	62.15	54.20	4.19	2.49	30.76	–
C4-native	ROME	1024	61.40	42.40	3.84	2.36	27.50	–
C4-native	ROME	2048	63.45	55.50	3.11	2.17	31.06	–
C4-native	R-ROME	512	62.20	53.15	4.61	1.96	30.48	–
C4-native	R-ROME	1024	61.25	52.40	4.12	2.34	30.03	–
C4-native	R-ROME	2048	59.55	51.05	3.76	2.44	29.20	–
C4-native	EMMET	512	67.90	63.60	9.25	5.85	36.65	–
C4-native	EMMET	1024	70.00	64.80	7.75	5.59	37.04	–
C4-native	EMMET	2048	71.45	66.45	9.04	4.11	37.76	–
C4-native	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	–
C4-native	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	–
C4-native	MEMIT	2048	0.00	0.00	0.03	0.00	0.01†	–
C4-native	AlphaEdit	512	9.35	3.55	3.62	7.41	5.98	–
C4-native	AlphaEdit	1024	91.70	58.15	28.31	27.02	51.29	–
C4-native	AlphaEdit	2048	93.65	75.90	32.51	25.15	56.80	–
C4-match	ROME	512	63.45	57.00	3.39	2.47	31.58	–
C4-match	ROME	1024	61.05	54.10	3.38	2.08	30.15	–
C4-match	ROME	2048	59.90	51.70	2.79	2.16	29.14	–
C4-match	R-ROME	512	59.85	49.60	3.18	1.93	28.64	–
C4-match	R-ROME	1024	62.15	54.40	3.15	1.99	30.42	–
C4-match	R-ROME	2048	57.40	46.80	3.03	2.26	27.37	–
C4-match	EMMET	512	66.55	66.75	6.91	3.15	35.84	–
C4-match	EMMET	1024	65.85	65.35	8.04	3.57	35.70	–
C4-match	EMMET	2048	68.15	69.90	6.17	3.72	36.99	–
C4-match	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	–
C4-match	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	–
C4-match	MEMIT	2048	0.00	0.00	0.03	0.00	0.01†	–
C4-match	AlphaEdit	512	8.80	6.45	2.73	5.93	5.98	–
C4-match	AlphaEdit	1024	97.60	84.35	39.68	28.65	62.57	–
C4-match	AlphaEdit	2048	51.20	39.25	12.06	15.72	29.56	–
PG19-native	ROME	512	57.50	37.30	2.03	1.95	24.69	–
PG19-native	ROME	1024	57.05	39.10	1.97	2.43	25.14	–
PG19-native	ROME	2048	44.70	25.75	3.07	1.65	18.79	–
PG19-native	R-ROME	512	59.50	44.80	2.07	2.17	27.13	–
PG19-native	R-ROME	1024	56.30	42.70	3.28	2.71	26.25	–
PG19-native	R-ROME	2048	54.65	37.85	3.20	1.76	24.37	–
PG19-native	EMMET	512	59.70	52.65	3.17	2.06	29.40	–
PG19-native	EMMET	1024	62.00	54.40	2.64	2.16	30.30	–
PG19-native	EMMET	2048	60.85	48.70	2.34	2.04	28.48	–
PG19-native	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	–
PG19-native	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	–
PG19-native	MEMIT	2048	0.00	0.00	0.00	0.01	0.00†	–
PG19-native	AlphaEdit	512	24.95	17.30	1.60	1.74	11.40	–
PG19-native	AlphaEdit	1024	94.05	77.40	24.63	23.59	54.92	–
PG19-native	AlphaEdit	2048	22.70	13.70	2.44	4.11	10.74	–
PG19-match	ROME	512	61.65	45.80	3.73	1.98	28.29	–
PG19-match	ROME	1024	62.35	44.80	2.93	2.37	28.11	–
PG19-match	ROME	2048	56.60	40.25	1.98	2.44	25.32	–
PG19-match	R-ROME	512	62.60	49.75	3.97	2.14	29.62	–
PG19-match	R-ROME	1024	59.65	47.25	2.44	2.35	27.92	–
PG19-match	R-ROME	2048	61.75	45.45	4.07	2.43	28.42	–
PG19-match	EMMET	512	62.40	53.65	3.40	1.96	30.35	–
PG19-match	EMMET	1024	60.85	47.65	3.25	2.50	28.56	–
PG19-match	EMMET	2048	66.80	57.45	4.17	2.16	32.65	–
PG19-match	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	–
PG19-match	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	–
PG19-match	MEMIT	2048	0.00	0.00	0.03	0.00	0.01†	–
PG19-match	AlphaEdit	512	72.20	37.45	7.29	17.05	33.50	–
PG19-match	AlphaEdit	1024	20.80	14.00	1.42	0.54	9.19	–
PG19-match	AlphaEdit	2048	24.45	20.10	2.15	3.70	12.60	–

Setting	Editor	L	ES	GE	PS	NS	S_4	GA

Table 14: Full editing metrics for cross corpus. $N = 1000$, seed 42; all scores are percentages. ES: rewrite; GE: rephrase; PS: portability; NS: locality. GA is the five-task mean, joined only by exact checkpoint name.

Table 15: Full editing metrics for construction. $N = 1000$, seed 42; all scores are percentages. ES: rewrite; GE: rephrase; PS: portability; NS: locality. GA is the five-task mean, joined only by exact checkpoint name.

Setting	Editor	L	ES	GE	PS	NS	S_4	GA
Prefix	ROME	512	58.90	45.20	2.37	3.59	27.52	12.86
Prefix	ROME	1024	54.60	43.15	2.33	3.08	25.79	13.23
Prefix	ROME	2048	59.90	43.85	2.74	2.93	27.35	12.26
Prefix	R-ROME	512	62.25	54.50	4.27	6.79	31.95	13.13
Prefix	R-ROME	1024	58.95	53.10	3.46	5.50	30.25	18.09
Prefix	R-ROME	2048	57.25	48.25	3.83	3.04	28.09	15.45
Prefix	EMMET	512	77.15	65.15	12.39	7.62	40.58	18.09
Prefix	EMMET	1024	75.70	66.20	11.65	6.65	40.05	18.82
Prefix	EMMET	2048	72.45	65.05	9.30	5.75	38.14	16.35
Prefix	MEMIT	512	0.00	0.00	0.00	0.06	0.01†	13.12
Prefix	MEMIT	1024	0.00	0.00	0.00	0.06	0.01†	14.12
Prefix	MEMIT	2048	0.00	0.00	0.00	0.06	0.01†	14.12
Prefix	AlphaEdit	512	34.95	19.25	5.34	11.67	17.80	15.12
Prefix	AlphaEdit	1024	99.10	83.05	50.99	31.30	66.11	57.79
Prefix	AlphaEdit	2048	98.80	83.45	49.26	31.35	65.71	56.39
Sequential	ROME	512	44.10	24.15	2.88	1.85	18.24	11.28
Sequential	ROME	1024	52.15	36.85	2.30	2.06	23.34	12.21
Sequential	ROME	2048	52.50	39.10	2.90	1.89	24.10	–
Sequential	R-ROME	512	47.30	42.25	2.65	1.87	23.52	15.14
Sequential	R-ROME	1024	53.40	47.50	2.99	2.55	26.61	11.57
Sequential	R-ROME	2048	52.75	42.65	2.88	1.86	25.03	12.79
Sequential	EMMET	512	61.75	66.70	6.37	2.61	34.36	14.82
Sequential	EMMET	1024	64.85	67.00	6.73	2.35	35.23	14.26
Sequential	EMMET	2048	53.45	57.45	2.46	2.26	28.91	13.57
Sequential	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	17.05
Sequential	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	16.04
Sequential	MEMIT	2048	0.00	0.00	0.03	0.00	0.01†	16.04
Sequential	AlphaEdit	512	46.40	43.50	26.36	15.54	32.95	14.13
Sequential	AlphaEdit	1024	95.20	75.75	37.99	30.60	59.88	26.37
Sequential	AlphaEdit	2048	87.15	52.90	33.57	29.58	50.80	19.34
Within-document	ROME	512	48.55	39.90	1.92	1.42	22.95	12.57
Within-document	ROME	1024	50.30	35.60	2.58	1.58	22.52	11.86
Within-document	ROME	2048	43.40	36.00	2.56	2.35	21.08	13.27
Within-document	R-ROME	512	50.05	45.00	1.65	2.14	24.71	14.63
Within-document	R-ROME	1024	44.05	40.35	2.28	1.71	22.10	13.56
Within-document	R-ROME	2048	43.65	38.75	2.02	1.75	21.54	12.98
Within-document	EMMET	512	57.65	64.30	6.47	2.22	32.66	14.63
Within-document	EMMET	1024	61.35	66.20	7.16	1.85	34.14	12.47
Within-document	EMMET	2048	57.10	61.45	5.00	1.56	31.28	15.49
Within-document	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	16.04
Within-document	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	16.04
Within-document	MEMIT	2048	0.00	0.00	0.00	0.06	0.01†	12.34
Within-document	AlphaEdit	512	92.00	65.85	29.05	26.60	53.38	25.33
Within-document	AlphaEdit	1024	87.95	65.35	30.40	29.75	53.36	16.76
Within-document	AlphaEdit	2048	93.60	77.40	46.70	29.79	61.87	23.05
Global-sentence	ROME	512	38.55	30.45	1.14	1.36	17.87	14.78
Global-sentence	ROME	1024	37.95	33.80	1.64	1.59	18.75	14.62
Global-sentence	ROME	2048	27.75	19.50	2.15	1.47	12.72	–
Global-sentence	R-ROME	512	38.70	32.65	1.55	1.11	18.50	11.77
Global-sentence	R-ROME	1024	35.30	31.60	0.99	1.47	17.34	14.68
Global-sentence	R-ROME	2048	34.05	30.20	1.09	1.55	16.72	12.14
Global-sentence	EMMET	512	45.70	54.00	2.23	2.11	26.01	13.53
Global-sentence	EMMET	1024	47.35	56.75	3.11	2.27	27.37	14.07
Global-sentence	EMMET	2048	35.15	42.75	1.41	1.79	20.28	14.48
Global-sentence	MEMIT	512	0.00	0.00	0.03	0.00	0.01†	16.04
Global-sentence	MEMIT	1024	0.00	0.00	0.03	0.00	0.01†	16.04
Global-sentence	MEMIT	2048	0.00	0.00	0.03	0.00	0.01†	16.04
Global-sentence	AlphaEdit	512	11.15	8.40	3.05	2.82	6.36	12.88

Setting	Editor	L	ES	GE	PS	NS	S_4	GA
Global-sentence	AlphaEdit	1024	19.10	8.25	0.55	6.59	8.62	14.25
Global-sentence	AlphaEdit	2048	89.75	70.40	31.39	26.14	54.42	14.26
Uniform-token	ROME	512	42.05	33.00	1.83	1.53	19.60	12.64
Uniform-token	ROME	1024	–	–	–	–	invalid	–
Uniform-token	ROME	2048	–	–	–	–	invalid	–
Uniform-token	R-ROME	512	37.65	29.90	2.78	1.18	17.88	11.21
Uniform-token	R-ROME	1024	–	–	–	–	invalid	–
Uniform-token	R-ROME	2048	–	–	–	–	invalid	–
Uniform-token	EMMET	512	43.45	41.15	2.34	1.84	22.19	12.92
Uniform-token	EMMET	1024	39.30	40.50	1.69	1.42	20.73	13.80
Uniform-token	EMMET	2048	45.45	42.25	1.49	1.89	22.77	–
Uniform-token	MEMIT	512	0.00	0.00	0.03	0.00	$0.01^{\dagger}$	14.12
Uniform-token	MEMIT	1024	0.00	0.00	0.00	0.06	$0.01^{\dagger}$	–
Uniform-token	MEMIT	2048	–	–	–	–	–	–
Uniform-token	AlphaEdit	512	19.15	12.25	2.10	0.91	8.60	14.04
Uniform-token	AlphaEdit	1024	2.80	0.70	0.00	0.15	0.91	–
Uniform-token	AlphaEdit	2048	–	–	–	–	–	–
Padded	ROME	512	49.05	39.75	2.22	1.65	23.17	11.60
Padded	ROME	1024	49.45	38.35	1.99	2.09	22.97	12.28
Padded	ROME	2048	46.80	28.70	1.20	2.25	19.74	13.68
Padded	R-ROME	512	54.05	47.90	2.48	2.35	26.70	13.62
Padded	R-ROME	1024	51.85	42.70	2.59	2.14	24.82	14.03
Padded	R-ROME	2048	53.10	45.75	2.82	1.90	25.89	15.85
Padded	EMMET	512	60.45	60.25	4.03	2.33	31.76	13.16
Padded	EMMET	1024	50.45	51.20	2.79	1.84	26.57	12.10
Padded	EMMET	2048	49.75	48.25	2.51	1.79	25.57	14.85
Padded	MEMIT	512	0.00	0.00	0.00	0.06	$0.01^{\dagger}$	13.12
Padded	MEMIT	1024	0.00	0.00	0.03	0.00	$0.01^{\dagger}$	14.12
Padded	MEMIT	2048	0.00	0.00	0.03	0.00	$0.01^{\dagger}$	17.04
Padded	AlphaEdit	512	16.95	11.35	2.23	3.17	8.43	15.75
Padded	AlphaEdit	1024	98.00	83.75	40.42	29.71	62.97	57.42
Padded	AlphaEdit	2048	86.40	54.10	33.12	24.98	49.65	25.93

Table 16: Full editing metrics for cross model. $N = 1000$, seed 42; all scores are percentages. ES: rewrite; GE: rephrase; PS: portability; NS: locality. GA is the five-task mean, joined only by exact checkpoint name.

Setting	Editor	L	ES	GE	PS	NS	S_4	GA
Llama-3.1-8B-Instruct	ROME	512	58.90	45.20	2.37	3.59	27.52	12.86
Llama-3.1-8B-Instruct	ROME	1024	54.60	43.15	2.33	3.08	25.79	13.23
Llama-3.1-8B-Instruct	ROME	2048	55.10	38.90	2.83	2.70	24.88	10.06
Llama-3.1-8B-Instruct	R-ROME	512	62.25	54.50	4.27	6.79	31.95	13.13
Llama-3.1-8B-Instruct	R-ROME	1024	58.95	53.10	3.46	5.50	30.25	18.09
Llama-3.1-8B-Instruct	R-ROME	2048	59.85	41.30	2.65	3.62	26.85	12.36
Llama-3.1-8B-Instruct	EMMET	512	77.15	65.15	12.39	7.62	40.58	–
Llama-3.1-8B-Instruct	EMMET	1024	75.70	66.20	11.65	6.65	40.05	–
Llama-3.1-8B-Instruct	EMMET	2048	73.35	64.65	9.00	5.66	38.17	–
Llama-3.1-8B-Instruct	MEMIT	512	0.00	0.00	0.00	0.06	0.01†	13.12
Llama-3.1-8B-Instruct	MEMIT	1024	0.00	0.00	0.00	0.06	0.01†	14.12
Llama-3.1-8B-Instruct	MEMIT	2048	0.00	0.00	2.21	0.28	0.62†	16.31
Llama-3.1-8B-Instruct	AlphaEdit	512	34.95	19.25	5.34	11.67	17.80	15.12
Llama-3.1-8B-Instruct	AlphaEdit	1024	99.10	83.05	50.99	31.30	66.11	58.20
Llama-3.1-8B-Instruct	AlphaEdit	2048	96.60	71.80	38.93	30.41	59.44	27.27
Llama-3.1-8B Base	ROME	512	49.95	37.20	2.01	4.79	23.49	14.98
Llama-3.1-8B Base	ROME	1024	53.05	37.10	1.57	3.77	23.87	–
Llama-3.1-8B Base	ROME	2048	52.00	34.30	2.70	4.21	23.30	12.81
Llama-3.1-8B Base	R-ROME	512	56.65	53.80	2.75	5.85	29.76	12.17
Llama-3.1-8B Base	R-ROME	1024	55.60	47.10	3.09	5.51	27.82	–
Llama-3.1-8B Base	R-ROME	2048	57.75	45.40	3.22	4.61	27.74	11.84
Llama-3.1-8B Base	EMMET	512	71.05	62.95	9.35	6.50	37.46	–
Llama-3.1-8B Base	EMMET	1024	71.50	61.35	6.84	5.00	36.17	–
Llama-3.1-8B Base	EMMET	2048	65.50	55.25	4.30	4.62	32.42	–
Llama-3.1-8B Base	MEMIT	512	0.00	0.00	2.21	0.23	0.61†	16.12
Llama-3.1-8B Base	MEMIT	1024	0.00	0.00	0.06	6.26	1.58†	–
Llama-3.1-8B Base	MEMIT	2048	0.00	0.00	2.21	0.23	0.61†	16.12
Llama-3.1-8B Base	AlphaEdit	512	67.80	43.95	20.48	26.83	39.77	17.64
Llama-3.1-8B Base	AlphaEdit	1024	82.10	65.05	29.01	31.89	52.01	–
Llama-3.1-8B Base	AlphaEdit	2048	2.20	0.90	0.71	2.02	1.46	11.61
Llama-3.2-3B-Instruct	ROME	512	50.45	31.85	1.72	2.92	21.73	15.53
Llama-3.2-3B-Instruct	ROME	1024	53.70	34.95	1.63	2.03	23.08	13.01
Llama-3.2-3B-Instruct	ROME	2048	52.10	30.95	2.16	2.91	22.03	15.17
Llama-3.2-3B-Instruct	R-ROME	512	46.45	28.30	1.15	3.55	19.86	13.95
Llama-3.2-3B-Instruct	R-ROME	1024	50.75	34.65	1.83	3.35	22.64	16.30
Llama-3.2-3B-Instruct	R-ROME	2048	49.25	36.65	1.95	3.64	22.87	15.36
Llama-3.2-3B-Instruct	EMMET	512	77.40	70.30	16.66	14.67	44.76	16.67
Llama-3.2-3B-Instruct	EMMET	1024	77.05	69.85	14.51	11.05	43.11	13.26
Llama-3.2-3B-Instruct	EMMET	2048	73.05	67.15	13.47	11.00	41.17	19.05
Llama-3.2-3B-Instruct	MEMIT	512	0.00	0.00	2.21	0.28	0.62†	13.12
Llama-3.2-3B-Instruct	MEMIT	1024	0.00	0.00	0.00	0.11	0.03†	16.30
Llama-3.2-3B-Instruct	MEMIT	2048	0.00	0.00	0.00	0.11	0.03†	16.12
Llama-3.2-3B-Instruct	AlphaEdit	512	98.50	81.30	45.62	31.16	64.14	40.54
Llama-3.2-3B-Instruct	AlphaEdit	1024	98.10	82.45	41.78	29.41	62.93	43.61
Llama-3.2-3B-Instruct	AlphaEdit	2048	98.40	82.50	42.52	29.88	63.32	43.80
Qwen2.5-7B-Instruct	ROME	512	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	ROME	1024	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	ROME	2048	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	R-ROME	512	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	R-ROME	1024	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	R-ROME	2048	0.00	0.00	0.00	0.00	0.00†	14.04
Qwen2.5-7B-Instruct	EMMET	512	95.85	72.20	47.71	46.17	65.48	–
Qwen2.5-7B-Instruct	EMMET	1024	96.05	74.35	48.73	43.81	65.74	–
Qwen2.5-7B-Instruct	EMMET	2048	95.90	70.85	47.17	43.34	64.31	–
Qwen2.5-7B-Instruct	MEMIT	512	0.00	0.00	0.04	5.12	1.29†	14.78
Qwen2.5-7B-Instruct	MEMIT	1024	–	–	–	–	–	–
Qwen2.5-7B-Instruct	MEMIT	2048	0.00	0.00	0.04	6.33	1.59†	15.78

Setting	Editor	L	ES	GE	PS	NS	S_4	GA
Qwen2.5-7B-Instruct	AlphaEdit	512	–	–	–	–	–	–
Qwen2.5-7B-Instruct	AlphaEdit	1024	–	–	–	–	–	–
Qwen2.5-7B-Instruct	AlphaEdit	2048	–	–	–	–	–	–

Tokenizer	Median	q_{75}	≤ 512	> 2048	512→1024	1024→2048	2048→4096
Llama-3.1-8B	313	722	65.1%	6.5%	+39.65%	+24.47%	+13.71%
Qwen2.5-7B	329	761	63.6%	6.9%	+40.96%	+25.35%	+14.24%

Table 17: Calibration-data attribution for covariance length effects on Wikipedia. Effective-token increases are computed on the same fixed 100,000-sample Wikipedia subset used for covariance estimation.

Model	$L = 512$	$L = 1024$	$L = 2048$	$L = 4096$
Llama-3.1-8B	1.36 GiB	1.64 GiB	2.19 GiB	2.84 GiB
Qwen2.5-7B	1.99 GiB	2.51 GiB	3.54 GiB	4.41 GiB

Table 18: Estimated covariance-specific extra peak memory. The estimate includes the `mom2` matrix, the $a^\top a$ temporary matrix, retained layer inputs, and flattened features, but excludes model weights and forward-time activations.

C.7 Analysis Table

C.8 Extended Discussions

Table 19: Full model editing and per-task general ability metrics.

Model	Method	N	L	Edit	Rewrite	Port.	Local.	Gen.	BBH	CSQA	GPQA	GSM8K	MMLU	Has Gen.
Qwen2.5-7B	EMMET	500	512	31.07	0.70	41.45	51.06	61.50	45.78	82.56	31.31	74.37	73.49	yes
Qwen2.5-7B	EMMET	500	1024	29.86	0.30	42.29	46.99	62.29	41.79	83.78	31.82	79.61	74.45	yes
Qwen2.5-7B	EMMET	500	2048	29.35	0.50	41.11	46.43	61.48	41.18	81.74	30.30	80.82	73.35	yes
Qwen2.5-7B	EMMET	1000	512	26.20	0.70	38.92	39.00	59.41	40.64	81.00	31.82	73.24	70.33	yes
Qwen2.5-7B	EMMET	1000	1024	24.38	0.45	38.15	34.55	58.02	37.04	80.26	27.27	76.04	69.51	yes
Qwen2.5-7B	EMMET	1000	2048	24.54	0.40	38.04	35.18	56.15	32.44	77.31	31.31	71.27	68.41	yes
Qwen2.5-7B	MEMIT	500	512	65.42	95.40	47.21	53.66	65.51	52.68	83.62	34.85	81.96	74.45	yes
Qwen2.5-7B	MEMIT	500	1024	61.75	91.30	45.02	48.94	65.80	56.13	83.70	33.84	80.89	74.45	yes
Qwen2.5-7B	MEMIT	500	2048	55.21	82.40	43.05	40.18	62.97	44.79	82.06	36.36	78.17	73.49	yes
Qwen2.5-7B	MEMIT	1000	512	1.72	0.00	0.04	5.12	15.05	0.00	20.64	28.79	0.00	25.82	yes
Qwen2.5-7B	MEMIT	1000	2048	2.15	0.00	0.04	6.41	15.07	0.00	20.88	28.79	0.00	25.69	yes
Qwen2.5-7B	R-ROME	500	512	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	R-ROME	500	1024	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	R-ROME	500	2048	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	R-ROME	1000	512	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	R-ROME	1000	1024	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	R-ROME	1000	2048	0.00	0.00	0.00	0.00	14.01	0.00	19.57	25.76	0.00	24.73	yes
Qwen2.5-7B	ROME	500	512	14.96	39.20	2.56	3.14	13.76	0.00	20.31	23.23	0.00	25.27	yes
Qwen2.5-7B	ROME	500	1024	14.96	39.20	2.56	3.14	13.76	0.00	20.31	23.23	0.00	25.27	yes
Qwen2.5-7B	ROME	500	2048	14.96	39.20	2.56	3.14	13.76	0.00	20.31	23.23	0.00	25.27	yes
Qwen2.5-7B	ROME	1000	512	16.31	44.65	2.30	1.97	13.98	0.00	20.64	24.24	0.00	25.00	yes
Qwen2.5-7B	ROME	1000	1024	16.31	44.65	2.30	1.97	13.98	0.00	20.64	24.24	0.00	25.00	yes
Qwen2.5-7B	ROME	1000	2048	16.31	44.65	2.30	1.97	13.98	0.00	20.64	24.24	0.00	25.00	yes
Llama-3.1-8B	AlphaEdit	100	512	75.58	100.00	62.81	63.93	65.45	70.09	77.89	32.83	76.95	69.51	yes
Llama-3.1-8B	AlphaEdit	100	1024	74.49	100.00	60.80	62.67	65.54	69.71	78.21	32.83	77.71	69.23	yes
Llama-3.1-8B	AlphaEdit	100	2048	74.70	100.00	62.01	62.10	–	–	–	–	–	–	no
Llama-3.1-8B	AlphaEdit	500	512	65.71	99.20	55.55	42.39	64.13	69.25	77.56	30.30	75.97	67.58	yes
Llama-3.1-8B	AlphaEdit	500	1024	65.71	99.20	55.55	42.39	64.13	69.25	77.56	30.30	75.97	67.58	yes
Llama-3.1-8B	AlphaEdit	500	2048	65.71	99.20	55.55	42.39	64.13	69.25	77.56	30.30	75.97	67.58	yes
Llama-3.1-8B	AlphaEdit	1000	512	59.09	97.60	48.16	31.51	38.91	12.96	65.36	25.76	34.72	55.77	yes
Llama-3.1-8B	AlphaEdit	1000	1024	59.09	97.60	48.16	31.51	38.91	12.96	65.36	25.76	34.72	55.77	yes
Llama-3.1-8B	AlphaEdit	1000	2048	59.09	97.60	48.16	31.51	38.91	12.96	65.36	25.76	34.72	55.77	yes
Llama-3.1-8B	EMMET	100	512	36.97	0.00	44.25	66.67	65.17	71.24	77.15	33.84	77.41	66.21	yes
Llama-3.1-8B	EMMET	100	1024	37.13	0.00	46.52	64.87	65.22	69.71	77.15	33.84	78.09	67.31	yes
Llama-3.1-8B	EMMET	100	2048	35.97	0.50	42.32	65.10	–	–	–	–	–	–	no
Llama-3.1-8B	EMMET	500	512	17.32	0.20	20.83	30.91	32.49	33.67	31.29	31.31	27.75	38.46	yes
Llama-3.1-8B	EMMET	500	1024	15.75	0.30	18.76	28.18	31.93	36.66	34.73	22.22	23.58	42.45	yes
Llama-3.1-8B	EMMET	500	2048	11.56	0.40	14.18	20.09	24.33	20.86	29.16	22.73	10.84	38.05	yes
Llama-3.1-8B	EMMET	1000	512	5.24	0.00	3.11	12.61	16.29	7.44	21.21	26.26	1.82	24.73	yes
Llama-3.1-8B	EMMET	1000	1024	4.63	0.00	3.28	10.61	14.06	3.91	20.97	21.72	1.44	22.25	yes
Llama-3.1-8B	EMMET	1000	2048	3.11	0.00	2.75	6.59	14.97	3.30	20.64	24.75	1.29	24.86	yes
Llama-3.1-8B	MEMIT	100	512	36.50	76.00	14.31	19.20	28.51	35.58	26.78	26.77	10.16	43.27	yes
Llama-3.1-8B	MEMIT	100	1024	21.48	53.00	4.93	6.50	16.68	6.60	20.15	27.27	1.36	28.02	yes
Llama-3.1-8B	MEMIT	100	2048	13.53	33.00	3.67	3.93	–	–	–	–	–	–	no
Llama-3.1-8B	MEMIT	500	512	0.03	0.00	0.00	0.10	13.38	0.00	20.56	21.21	0.00	25.14	yes
Llama-3.1-8B	MEMIT	500	1024	0.03	0.00	0.00	0.10	14.29	0.00	20.56	25.76	0.00	25.14	yes
Llama-3.1-8B	MEMIT	500	2048	0.03	0.00	0.00	0.10	14.29	0.00	20.56	25.76	0.00	25.14	yes
Llama-3.1-8B	MEMIT	1000	512	0.02	0.00	0.00	0.06	13.38	0.00	20.56	21.21	0.00	25.14	yes
Llama-3.1-8B	MEMIT	1000	1024	0.02	0.00	0.00	0.06	14.29	0.00	20.56	25.76	0.00	25.14	yes
Llama-3.1-8B	MEMIT	1000	2048	0.02	0.00	0.00	0.06	14.29	0.00	20.56	25.76	0.00	25.14	yes
Llama-3.1-8B	R-ROME	100	512	74.32	98.00	63.67	61.30	64.50	68.33	76.90	34.85	76.50	65.93	yes
Llama-3.1-8B	R-ROME	100	1024	74.01	99.00	63.12	59.90	64.32	67.94	77.56	33.84	75.21	67.03	yes
Llama-3.1-8B	R-ROME	100	2048	73.20	98.00	61.59	60.00	–	–	–	–	–	–	no
Llama-3.1-8B	R-ROME	500	512	3.41	9.00	0.18	1.05	15.03	0.00	22.77	27.78	0.00	24.59	yes
Llama-3.1-8B	R-ROME	500	1024	3.41	9.00	0.18	1.05	15.03	0.00	22.77	27.78	0.00	24.59	yes
Llama-3.1-8B	R-ROME	500	2048	3.41	9.00	0.18	1.05	15.03	0.00	22.77	27.78	0.00	24.59	yes
Llama-3.1-8B	R-ROME	1000	512	1.80	5.20	0.18	0.01	13.97	0.00	19.98	26.26	0.00	23.63	yes
Llama-3.1-8B	R-ROME	1000	1024	1.80	5.20	0.18	0.01	13.97	0.00	19.98	26.26	0.00	23.63	yes
Llama-3.1-8B	R-ROME	1000	2048	1.80	5.20	0.18	0.01	13.97	0.00	19.98	26.26	0.00	23.63	yes
Llama-3.1-8B	ROME	100	512	73.57	99.00	60.75	60.97	64.24	68.17	77.48	32.83	75.28	67.45	yes
Llama-3.1-8B	ROME	100	1024	73.56	98.00	61.89	60.80	64.69	68.33	77.15	34.34	77.03	66.62	yes
Llama-3.1-8B	ROME	100	2048	72.82	98.00	61.19	59.27	–	–	–	–	–	–	no
Llama-3.1-8B	ROME	500	512	2.08	5.60	0.10	0.55	14.73	0.00	21.54	27.27	0.00	24.86	yes
Llama-3.1-8B	ROME	500	1024	2.08	5.60	0.10	0.55	14.73	0.00	21.54	27.27	0.00	24.86	yes
Llama-3.1-8B	ROME	500	2048	2.08	5.60	0.10	0.55	14.73	0.00	21.54	27.27	0.00	24.86	yes
Llama-3.1-8B	ROME	1000	512	0.50	1.20	0.23	0.06	14.54	0.00	20.88	26.26	0.00	25.55	yes
Llama-3.1-8B	ROME	1000	1024	0.50	1.20	0.23	0.06	14.54	0.00	20.88	26.26	0.00	25.55	yes
Llama-3.1-8B	ROME	1000	2048	0.50	1.20	0.23	0.06	14.54	0.00	20.88	26.26	0.00	25.55	yes

D Detailed Results For Update Geometry

Section 4 shows that a more expensive covariance estimate does not necessarily produce a better edited model.

The next question is therefore not whether two covariance matrices are different in isolation, but whether an editor converts that difference into a different parameter update. This distinction is important because the same preservation statistic enters different editing algorithms through different computations: a change that is consequential for one editor may cancel, be absorbed, or be transformed for another.

We use *update geometry* to refer to the direction, magnitude, and layer-wise distribution of the realized weight change. Starting from a reference covariance

$$C_\star = U\Lambda U^\top, \tag{31}$$

we intervene on three properties of the covariance—global scale, spectral shape, and directional structure—and track both editing behavior and the update that is ultimately applied to the model. For a covariance condition C, let $\Delta(C)$ denote the realized weight update and $\Delta_\star = \Delta(C_\star)$ the update under the reference covariance. For multi-layer editors, $\Delta(C)$ denotes the concatenation of the edited-layer updates unless otherwise stated.

The analysis follows the full path

$$C \longrightarrow \text{editor-specific update computation} \longrightarrow \Delta W \longrightarrow \{\Delta W_l\}. \tag{32}$$

This lets us separate matrix-level changes that are visually or numerically large from changes that actually alter the edit applied to the model.

D.1 Scale

Global rescaling is a useful first intervention because it changes the magnitude of the covariance without changing its eigendirections or spectral shape. We construct

$$C_s = sC_\star. \tag{33}$$

However, scale does *not* have the same mathematical meaning for every editor. In the ideal single-edit ROME update, scalar covariance rescaling cancels exactly:

$$\Delta_{\mathrm{ROME}}(sC) = r_e \frac{k_e^\top (sC)^{-1}}{k_e^\top (sC)^{-1} k_e} \tag{34}$$

$$= r_e \frac{s^{-1} k_e^\top C^{-1}}{s^{-1} k_e^\top C^{-1} k_e} = \Delta_{\mathrm{ROME}}(C). \tag{35}$$

The same cancellation holds for the equality-constrained EMMET factor

$$\left(K_E^\top (sC)^{-1} K_E\right)^{-1} K_E^\top (sC)^{-1} = \left(K_E^\top C^{-1} K_E\right)^{-1} K_E^\top C^{-1}. \tag{36}$$

Thus, departures from scale invariance for these idealized updates should be interpreted cautiously: extreme-scale failures can reflect conditioning, regularization, finite precision, or other implementation choices rather than a fundamental change in preservation direction.

MEMIT is different. Its update contains the preservation term inside a sum,

$$\Delta_{\mathrm{MEMIT}}(sC) = R K_E^\top \left(\lambda sC + K_E K_E^\top\right)^{-1}, \tag{37}$$

so s changes the relative operating point between preservation and memorization. Consistent with this form, the scale sweep is strongly non-monotonic for MEMIT: small covariance scales produce large, damaging updates, an intermediate scale can improve the editing–preservation trade-off, and very large scales approach a no-editing regime. In the current sweep, for example, the aggregate editing score increases from 0.2296 at the reference condition to 0.5952 at $s = 10$, while still larger scales increasingly suppress the edit.

AlphaEdit requires a separate interpretation. Its effective update is constructed through acovariance-derived projector rather than a direct C^{-1} preconditioner. Consequently, a scalar change to C matters only insofar as it changes the projector that AlphaEdit actually uses. We defer that pathway to Section E.3, where we explicitly distinguish recomputed, fixed-rank, and frozen-projector conditions.

Finding. Global covariance scale is not a universal edit-strength knob. It cancels in ideal equality-constrained updates, changes the preservation–memorization balance in MEMIT, and affects AlphaEdit primarily through projector construction. Scale sensitivity must therefore be interpreted through the update rule that consumes the covariance.

D.2 Spectrum

We next change how strongly different covariance directions are weighted while preserving the reference eigenvectors. Given $C_\star = U\Lambda U^\top$, we construct

$$C_\gamma = \alpha_\gamma U \Lambda^\gamma U^\top, \tag{38}$$

where α_γ normalizes the trace to match $C_\star$. We evaluate $\gamma \in \{0, 0.25, 0.5, 0.75, 1, 1.25, 1.5\}$: $\gamma < 1$ flattens the spectrum, $\gamma = 1$ recovers the reference covariance, and $\gamma > 1$ sharpens its anisotropy.

Table 21 summarizes the main pattern. Flattening the spectrum is damaging for the inverse-covariance-based editors, whereas moderate sharpening is typically safe and can be beneficial. ROME drops from 0.7610 to 0.0979 under the isotropic $\gamma = 0$ condition; at the other end, ROME becomes nonfinite at γ= 1.5, so "more sharpening is always better" is not a safe rule. MEMIT is also highly spectrum-sensitive: low-γ conditions collapse to zero, whereas γ = 1.25 raises the score from 0.1569 to 0.3628. AlphaEdit changes least under the same family of interventions, ranging from 0.7216 to 0.7596.

The important point is not that a particular sharpening exponent is universally optimal. Rather,flattening removes distinctions between high- and low-variance representation directions, changing which components of an edit key are amplified by the inverse geometry. The preferred spectral shape therefore depends on the editor, its solver, and the edit-key distribution.

Finding. Spectral anisotropy is functionally relevant. Destroying it can destabilize the realized update, while moderate sharpening can preserve or improve the update in some editor–model regimes. The sweep does not establish a universal optimal spectrum.

D.3 Directional Structure

Spectrum determines *how much* different principal components are weighted, but not *which representation directions* receive those weights. We therefore test whether directional structure matters beyond scale and eigenvalue magnitudes.

Direction–eigenvalue association. We first compare the full reference covariance with three structural ablations: identity covariance I, the coordinate-diagonal matrix $\mathrm{diag}(C_\star)$, and an eigenvalue-permuted matrix

$$C_\pi = U\Lambda_\pi U^\top. \tag{39}$$

These interventions should not all be described as "eigenbasis changes." Identity and diagonal covariances remove different parts of the full structure, while C_π keeps the eigenvectors fixed and instead breaks the pairing between covariance directions and eigenvalues. The latter therefore asks whether assigning the *right amount of preservation* to the *right direction* matters.

The result is strong for ROME, EMMET, and MEMIT. For ROME, the reference score 0.7610 falls to 0.0979 with identity covariance and to 0.0697 after eigenvalue permutation; keeping only the coordinate-wise diagonal partially recovers performance to 0.4621. MEMIT and EMMET degrade even more sharply under these ablations (MEMIT: 0.0000 / 0.0000 / 0.0103; EMMET: 0.1591 / 0.5688 / 0.1647).

Controlled rotation at fixed spectrum. The previous ablations alter several matrix properties at once. To isolate orientation more cleanly, we additionally rotate the covariance while preserving its eigenvalues:

$$C_\theta = R_\theta C_\star R_\theta^\top, \qquad \theta \in \{0^\circ, 15^\circ, 30^\circ, 60^\circ\}, \tag{40}$$

where R_θ is orthogonal. This construction preserves the complete eigenvalue multiset, trace, spectral norm, and Frobenius norm; only the orientation of the covariance relative to the edit keys changes. We evaluate this intervention for ROME and MEMIT and track the inverse-preconditioned keys, the edit-key Gram geometry, and realized update alignment.

At $\theta = 0^\circ$, the construction exactly recovers the reference covariance and the update alignment is one. Rotating the covariance can substantially change both editing behavior and the realized update even though the spectrum is fixed. The response is not monotonic in the rotation angle, however: a larger angle is not uniformly worse. This rules out a simple "more rotation means lower quality" interpretation. Instead, the controlled experiment indicates that what matters is the alignment between preservation directions and the edit-relevant representation geometry. For MEMIT, we interpret the fixed $C^{-1}K$ quantities only as diagnostics, because its actual multi-edit solver uses a term of the form $\lambda C + KK^\top$ and can further transform the covariance-induced geometry.

Finding. Editing depends on directional structure beyond eigenvalue magnitudes. In particular, both the assignment of covariance strength to representation directions and the orientation of those directions relative to edit keys can change the realized update. The evidence supports a directional-alignment claim, not the stronger claim that an "eigenbasis" is universally more important than the spectrum.

D.4 From Covariance Changes to Realized Updates

The interventions above show that covariance properties can matter, but they still leave open which intermediate quantity is closest to the editing outcome. We therefore compare matrix-side and update-side diagnostics. For each covariance variant C, we compute

$$a_k(C) = \cos\big(C^{-1}k, C_\star^{-1}k\big)\,, \tag{41}$$

$$h_C(k) = k^\top C^{-1}k, \tag{42}$$

$$n_\Delta(C) = \|\Delta(C)\|_F, \tag{43}$$

$$a_\Delta(C) = \cos(\operatorname{vec}\Delta(C), \operatorname{vec}\Delta_\star)\,. \tag{44}$$

For multi-layer editors, the update diagnostics use the concatenated edited-layer update.

Across the non-reference interventions in the current geometry sweep, update-side quantities carry more information about the editing outcome than covariance-side ones. Update alignment is more closely associated with editing score than the inverse-key diagnostics. Its Spearman correlation is 0.897 across all intervention families and 0.966 when restricted to spectrum and directional-structure interventions. Update norm is negatively associated with score (-0.691 across all variants and -0.777 on spectrum/directional variants), while scalar leverage is not stable across families. At the method level, inverse-key alignment is highly associated with score for ROME and EMMET under spectrum/directional perturbations (Spearman 0.971 and 0.957, respectively), but is less informative for MEMIT, consistent with its multi-layer update composition.

These correlations should not be interpreted as an out-of-sample prediction result. In particular, $a_\Delta(C)$ is a *post-edit diagnostic*: it is computed from the realized update and is therefore unavailable before the edit is performed. The evidence supports the narrower statement that, within the intervention set considered here, update-level similarity is more closely associated with editing behavior than raw covariance-side diagnostics.

The same distinction is useful for general ability. Among the evaluated quantities, neighborhood specificity has Spearman correlation 0.904 with the aggregate general-ability score, while update norm has correlation -0.895. This pattern is consistent with the view that preserving downstream ability is tied to avoiding overly destructive updates, rather than simply maximizing edit success.

Finding. Covariance differences become consequential when they survive the editor's update computation and change the realized update. Final update alignment is more closely associated with editing score than the covariance-side diagnostics considered here. Because it is computed from the realized update, it is a post-edit consistency measure rather than a validated pre-edit predictor.

D.5 Layer-wise Distribution of Covariance-Induced Update Differences

The preceding analysis characterizes the update produced under controlled covariance interventions. We now ask how covariance sensitivity appears after sequential editing has produced an actual checkpoint. Let $\Delta^{L,N}$ denote the concatenated edited-weight change after N sequential edits using a

Table 20: Checkpoint-level update drift relative to the $L = 2048$ covariance reference. $D_{L,N} = 1 - \cos(\Delta^{L,N}, \Delta^{2048,N})$ over the concatenated edited weights.

Method	Edited layers	$D_{512,500}$	$D_{1024,500}$	$D_{512,1000}$	$D_{1024,1000}$
AlphaEdit	4–8	0.0000	0.0000	0.0000	0.0000
EMMET	5	0.7182	0.7323	0.6969	0.6717
MEMIT	4–8	0.8207	0.7754	0.8143	0.7676
ROME	5	0.0000	0.0000	0.0000	0.0000

covariance estimated with maximum calibration length L. Taking $L = 2048$ as the reference, we define checkpoint-level direction drift as

$$D_{L,N} = 1 - \cos\big(\operatorname{vec}\Delta^{L,N}, \operatorname{vec}\Delta^{2048,N}\big)\,. \tag{45}$$

Table 20 shows that the same change in covariance length can produce very different parameter-space responses. At 500 and 1000 sequential edits, AlphaEdit and ROME produce nearly identical edited-weight directions across the tested covariance lengths, whereas EMMET and MEMIT exhibit large drift. MEMIT is the most sensitive multi-layer editor in this comparison.

For a multi-layer editor, overall drift does not reveal whether the difference is diffuse or localized. We therefore measure the update-energy share and the cross-length mismatch share of layer l:

$$e_l^{L,N} = \frac{\|\Delta_l^{L,N}\|_F^2}{\sum_j \|\Delta_j^{L,N}\|_F^2}, \tag{46}$$

$$m_l^{L,N} = \frac{\|\Delta_l^{L,N} - \Delta_l^{2048,N}\|_F^2}{\sum_j \|\Delta_j^{L,N} - \Delta_j^{2048,N}\|_F^2}. \tag{47}$$

Both AlphaEdit and MEMIT place substantial update energy in later edited layers, so high-layer energy alone does not explain sensitivity. The mismatch distribution is more informative. For MEMIT at 500 edits, layers 4–8 account for 0.0016, 0.0069, 0.0784, 0.2505, and 0.6626 of the cross-length mismatch, respectively. At 1000 edits, the corresponding shares are 0.0012, 0.0078, 0.0822, 0.2631, and 0.6458. Thus, layer 8 alone accounts for roughly two thirds of the mismatch, layers 7–8 account for about 91%, and layers 6–8 account for about 99%.

The mismatch also does not simply grow with the number of edits: the 500-to-1000 direction change varies across covariance lengths and editors. We therefore do not interpret the layer concentration as mere monotonic accumulation. Instead, it identifies where covariance-length sensitivity is expressed in the realized MEMIT checkpoint.

Finding. Covariance sensitivity is strongly editor-dependent in parameter space. For MEMIT, the length-induced checkpoint mismatch is highly localized in the upper edited layers. This localization identifies *where* the sensitivity appears, but not why those layers become functionally important. We address that question with direct update-procedure and layer interventions in Section 6.

D.6 Main Results

Method	Group	Condition	ES	GE	PS	NS	Score4
ROME	-	baseline	98.00	83.00	62.82	60.60	76.10
ROME	scale	scale_0.001	NF	NF	NF	NF	NF
ROME	scale	scale_0.01	NF	NF	NF	NF	NF
ROME	scale	scale_0.1	NF	NF	NF	NF	NF
ROME	scale	scale_10	99.00	82.00	63.25	60.50	76.19
ROME	scale	scale_100	98.00	83.00	62.59	60.93	76.13
ROME	scale	scale_1000	99.00	82.00	63.19	60.80	76.25
ROME	spectrum	spectrum_gamma_0	27.50	8.50	1.58	1.57	9.79
ROME	spectrum	spectrum_gamma_0.25	93.50	81.00	31.99	24.37	57.71
ROME	spectrum	spectrum_gamma_0.5	98.00	80.00	58.85	49.47	71.58
ROME	spectrum	spectrum_gamma_0.75	98.00	82.00	60.09	57.50	74.40
ROME	spectrum	spectrum_gamma_1.25	98.00	82.00	62.35	61.27	75.90
ROME	spectrum	spectrum_gamma_1.5	NF	NF	NF	NF	NF
ROME	eigenbasis	eigen_I	27.50	8.50	1.58	1.57	9.79
ROME	eigenbasis	eigen_diag	82.50	71.50	16.55	14.30	46.21
ROME	eigenbasis	eigen_perm_seed42	17.50	6.50	3.75	0.13	6.97
EMMET	-	baseline	99.00	79.00	63.70	59.23	75.23
EMMET	scale	scale_0.001	99.00	86.00	60.71	50.20	73.98
EMMET	scale	scale_0.01	98.00	84.00	57.53	52.30	72.96
EMMET	scale	scale_0.1	99.00	84.00	61.75	52.13	74.22
EMMET	scale	scale_10	55.00	30.00	58.21	81.53	56.19
EMMET	scale	scale_100	4.00	1.50	51.02	94.87	37.85
EMMET	scale	scale_1000	0.00	0.00	49.08	99.50	37.15
EMMET	spectrum	spectrum_gamma_0	35.00	24.00	2.58	2.07	15.91
EMMET	spectrum	spectrum_gamma_0.25	96.00	79.00	50.40	39.83	66.31
EMMET	spectrum	spectrum_gamma_0.5	97.00	75.00	62.63	52.10	71.68
EMMET	spectrum	spectrum_gamma_0.75	99.00	73.00	63.18	58.53	73.43
EMMET	spectrum	spectrum_gamma_1.25	99.00	83.00	63.42	56.53	75.49
EMMET	spectrum	spectrum_gamma_1.5	99.00	82.00	60.39	56.00	74.35
EMMET	eigenbasis	eigen_I	35.00	24.00	2.58	2.07	15.91
EMMET	eigenbasis	eigen_diag	87.50	75.50	35.31	29.20	56.88
EMMET	eigenbasis	eigen_perm_seed42	41.00	19.00	3.75	2.13	16.47
MEMIT	-	baseline	26.50	32.50	2.17	1.60	15.69
MEMIT	scale	scale_0.001	0.00	0.00	0.00	3.93	0.98
MEMIT	scale	scale_0.01	0.00	0.00	0.00	0.07	0.02
MEMIT	scale	scale_0.1	1.00	1.00	0.00	0.03	0.51
MEMIT	scale	scale_10	65.50	44.50	59.49	68.17	59.41
MEMIT	scale	scale_100	3.00	1.50	50.28	95.73	37.63
MEMIT	scale	scale_1000	0.00	0.00	49.08	99.83	37.23
MEMIT	spectrum	spectrum_gamma_0	0.00	0.00	0.00	0.00	0.00
MEMIT	spectrum	spectrum_gamma_0.25	0.00	0.00	2.93	0.43	0.84
MEMIT	spectrum	spectrum_gamma_0.5	0.00	0.00	0.00	0.00	0.00
MEMIT	spectrum	spectrum_gamma_0.75	4.00	1.00	0.67	0.27	1.48
MEMIT	spectrum	spectrum_gamma_1.25	60.50	55.50	15.59	13.53	36.28
MEMIT	spectrum	spectrum_gamma_1.5	53.00	45.00	8.03	11.17	29.30
MEMIT	eigenbasis	eigen_I	0.00	0.00	0.00	0.00	0.00
MEMIT	eigenbasis	eigen_diag	0.00	0.00	0.00	0.00	0.00
MEMIT	eigenbasis	eigen_perm_seed42	0.00	0.00	0.00	4.13	1.03
AlphaEdit	-	baseline	100.00	77.00	61.01	62.30	75.08
AlphaEdit	scale	scale_0.001	100.00	80.00	60.37	48.87	72.31
AlphaEdit	scale	scale_0.01	100.00	80.00	59.59	48.50	72.02
AlphaEdit	scale	scale_0.1	100.00	80.00	62.18	52.17	73.59
AlphaEdit	scale	scale_10	100.00	76.00	64.28	71.43	77.93
AlphaEdit	scale	scale_100	19.00	8.50	53.17	91.63	43.07
AlphaEdit	scale	scale_1000	0.00	0.00	48.88	100.00	37.22
AlphaEdit	spectrum	spectrum_gamma_0	100.00	81.00	60.42	48.37	72.45

AlphaEdit	spectrum	spectrum_gamma_0.25	100.00	79.00	60.59	49.07	72.16
AlphaEdit	spectrum	spectrum_gamma_0.5	100.00	78.50	63.10	51.70	73.33
AlphaEdit	spectrum	spectrum_gamma_0.75	100.00	81.00	63.58	57.50	75.52
AlphaEdit	spectrum	spectrum_gamma_1.25	100.00	81.00	60.28	62.57	75.96
AlphaEdit	spectrum	spectrum_gamma_1.5	100.00	79.00	62.35	61.60	75.74
AlphaEdit	eigenbasis	eigen_I	100.00	80.00	59.57	48.07	71.91
AlphaEdit	eigenbasis	eigen_diag	100.00	82.00	58.73	52.07	73.20
AlphaEdit	eigenbasis	eigen_perm_seed42	100.00	78.00	60.91	47.77	71.67
AlphaEdit	absolute	baseline	100.00	77.00	63.25	62.20	75.61
AlphaEdit	absolute	scale_0.1	100.00	80.00	62.18	52.17	73.59
AlphaEdit	absolute	scale_1	100.00	77.00	63.25	62.20	75.61
AlphaEdit	absolute	scale_10	100.00	76.00	64.28	71.43	77.93
AlphaEdit	absolute	scale_100	19.00	8.50	53.17	91.63	43.07
AlphaEdit	absolute	scale_1000	0.00	0.00	48.88	100.00	37.22
AlphaEdit	fixed_rank	baseline	43.00	19.50	55.66	85.97	51.03
AlphaEdit	fixed_rank	scale_0.1	43.00	19.50	55.66	85.97	51.03
AlphaEdit	fixed_rank	scale_1	43.00	19.50	55.66	85.97	51.03
AlphaEdit	fixed_rank	scale_10	43.00	19.50	55.66	85.97	51.03
AlphaEdit	fixed_rank	scale_100	43.00	19.50	55.66	85.97	51.03
AlphaEdit	fixed_rank	scale_1000	43.00	19.50	55.66	85.97	51.03
AlphaEdit	frozen_baseline	baseline	100.00	77.00	63.25	62.20	75.61
AlphaEdit	frozen_baseline	scale_0.1	100.00	77.00	63.25	62.20	75.61
AlphaEdit	frozen_baseline	scale_1	100.00	77.00	63.25	62.20	75.61
AlphaEdit	frozen_baseline	scale_10	100.00	77.00	63.25	62.20	75.61
AlphaEdit	frozen_baseline	scale_100	100.00	77.00	63.25	62.20	75.61
AlphaEdit	frozen_baseline	scale_1000	100.00	77.00	63.25	62.20	75.61

Table 21: All selected Section 5 editing conditions for Llama-3.1-8B-Instruct. Methods and intervention groups are shown in rows; metrics are percentages.

Method	Group	Condition	ES	GE	PS	NS	Score4
ROME	baseline	baseline	98.00	82.00	55.97	59.23	73.80
ROME	scale	scale_0.5	98.00	82.00	55.97	59.23	73.80
ROME	scale	scale_2	98.00	82.00	55.97	59.23	73.80
ROME	scale	scale_10	99.00	83.00	53.73	59.27	73.75
ROME	spectrum	spectrum_gamma_0	19.00	5.00	2.25	1.43	6.92
ROME	spectrum	spectrum_gamma_0.5	97.00	84.00	56.86	47.43	71.32
ROME	spectrum	spectrum_gamma_1.25	99.00	83.00	53.72	60.97	74.17
ROME	rotation	basis_rotate_0	98.00	82.00	55.97	59.23	73.80
ROME	rotation	basis_rotate_15	88.50	60.00	28.34	21.77	49.65
ROME	rotation	basis_rotate_30	4.00	1.00	3.60	0.43	2.26
ROME	rotation	basis_rotate_60	2.00	0.00	0.00	0.07	0.52
MEMIT	baseline	baseline	27.00	16.00	2.78	1.10	11.72
MEMIT	scale	scale_0.5	2.00	2.00	0.00	0.23	1.06
MEMIT	scale	scale_2	88.50	79.50	27.19	34.67	57.46
MEMIT	scale	scale_10	53.00	33.50	49.57	73.20	52.32
MEMIT	spectrum	spectrum_gamma_0	0.00	0.00	2.93	0.37	0.83
MEMIT	spectrum	spectrum_gamma_0.5	0.00	0.00	2.93	0.37	0.83
MEMIT	spectrum	spectrum_gamma_1.25	48.50	30.50	4.46	8.77	23.06
MEMIT	rotation	basis_rotate_0	27.00	16.00	2.78	1.10	11.72
MEMIT	rotation	basis_rotate_15	0.00	0.00	0.00	4.40	1.10
MEMIT	rotation	basis_rotate_30	0.00	0.00	2.93	0.37	0.83
MEMIT	rotation	basis_rotate_60	0.00	0.00	0.00	0.00	0.00

Table 22: All selected Section 5 editing conditions for llama3-8b-base. Methods and intervention groups are shown in rows; metrics are percentages.

Method	Group	Condition	ES	GE	PS	NS	Score4
ROME	baseline	baseline	44.00	31.50	2.97	2.23	20.17
ROME	scale	scale_0.5	44.00	31.50	2.97	2.23	20.17
ROME	scale	scale_2	44.00	31.50	2.97	2.23	20.17
ROME	scale	scale_10	43.50	27.50	2.83	2.63	19.12
ROME	spectrum	spectrum_gamma_0	56.00	46.00	16.14	30.47	37.15
ROME	spectrum	spectrum_gamma_0.5	95.00	74.00	35.68	35.17	59.96
MEMIT	baseline	baseline	100.00	78.50	48.16	69.23	73.97
MEMIT	scale	scale_0.5	100.00	77.00	50.89	68.77	74.16
MEMIT	scale	scale_2	100.00	78.50	46.70	70.73	73.98
MEMIT	scale	scale_10	87.50	53.50	47.92	79.97	67.22
MEMIT	spectrum	spectrum_gamma_0	81.00	74.50	36.45	36.27	57.06
MEMIT	spectrum	spectrum_gamma_0.5	99.00	82.00	49.99	64.73	73.93
MEMIT	spectrum	spectrum_gamma_1.25	99.00	75.50	49.54	68.47	73.13
MEMIT	rotation	basis_rotate_0	100.00	78.50	48.16	69.23	73.97
MEMIT	rotation	basis_rotate_15	92.00	66.00	38.83	40.93	59.44
MEMIT	rotation	basis_rotate_30	0.00	0.00	2.77	0.30	0.77
MEMIT	rotation	basis_rotate_60	0.00	0.00	0.00	0.00	0.00
ROME	spectrum	spectrum_gamma_1.25	–	–	–	–	–
ROME	rotation	basis_rotate_0	–	–	–	–	–
ROME	rotation	basis_rotate_15	–	–	–	–	–
ROME	rotation	basis_rotate_30	–	–	–	–	–
ROME	rotation	basis_rotate_60	–	–	–	–	–

Table 23: All selected Section 5 editing conditions for qwen2.5-7b. Methods and intervention groups are shown in rows; metrics are percentages.

Set	Editor	n	Update cos.	Norm	Key cos.	Leverage
All	Pooled	51	0.897	-0.570	0.340	0.000
All	Excl. AlphaEdit	38	0.907	-0.662	0.438	-0.081
All	ROME	10	0.988	-0.806	0.988	-0.624
All	EMMET	14	0.877	0.143	0.372	0.644
All	MEMIT	14	0.506	-0.784	0.575	-0.369
All	AlphaEdit	13	0.945	-0.874	0.149	-0.005
Spect./dir.	Pooled	31	0.966	-0.859	0.587	0.268
Spect./dir.	Excl. AlphaEdit	23	0.958	-0.883	0.675	0.365
Spect./dir.	ROME	7	1.000	-0.857	1.000	0.071
Spect./dir.	EMMET	8	0.905	-0.381	0.952	0.429
Spect./dir.	MEMIT	8	0.708	-0.634	0.659	0.634
Spect./dir.	AlphaEdit	8	0.952	-0.881	0.905	0.357

Table 24: Descriptive Spearman correlations with editing score, recomputed from the selected August runs and corrected EMMET. Baselines, identity duplicates, nonfinite updates, and undefined diagnostics are excluded. Each row uses a common finite set. Pooled results mix editors and are not prediction validation. The AlphaEdit-excluded rows are sensitivity checks.

Edits	L	Layer 6	Layer 7	Layer 8	Layers 7–8
500	512	7.84	25.05	66.26	91.31
500	1024	10.62	21.30	64.16	85.46
1000	512	8.22	26.31	64.58	90.89
1000	1024	10.46	21.98	59.39	81.37

Table 25: MEMIT shares (percent) of squared update mismatch relative to length 2048.

E MORE DETAILS FOR MECHANIC ANALYSIS

Section 5 shows that covariance differences become consequential when they change the parameter update actually applied to the model. Among the diagnostics considered there, update alignment is more closely associated with editing performance than the covariance-side diagnostics, while the magnitude and layer-wise distribution of the update vary substantially across editors. These observations move the analysis closer to the edited checkpoint, but they do not yet explain why the same covariance change can produce a large update drift for one editor and almost no change for another.

We therefore intervene on the editor-specific computations that transform covariance into realized updates. We focus on two representative mechanisms. For MEMIT, we separate the effect of multi-layer residual allocation from the magnitude of the resulting update, and then test whether the layer-localized drift observed in Section 5 has functional consequences. For AlphaEdit, we separate covariance scaling from changes in the projection subspace used to construct the edit. Together, these experiments examine the pathway

$$C \longrightarrow g_{\text{editor}}(C) \longrightarrow \Delta W \longrightarrow \text{editing behavior},$$

and identify where editor-specific computations amplify, redistribute, or suppress covariance differences.

E.1 MEMIT: UPDATE ALLOCATION AND THE ROLE OF UPDATE MAGNITUDE

MEMIT distributes the residual correction across multiple edited layers. In the implementation studied here, the original *remaining* rule divides the residual at each layer by the number of edited layers that remain, whereas a *uniform* variant uses the total number of edited layers at every step. Changing this rule substantially improves editing in our diagnostic setting. However, the intervention changes two quantities at once: it changes how the residual is distributed across layers and also changes the magnitude of the update ultimately written to the model. Consequently, an original-versus-uniform comparison alone cannot identify residual allocation as the mechanism.

We therefore introduce a reverse norm-matching control. Let $\Delta_{t,l}$ denote the update computed for edit step t at edited layer l, and define the joint per-edit update norm

$$\|\Delta_t\|_{\text{joint}} = \left(\sum_{l \in \mathcal{L}_{\text{edit}}} \|\Delta_{t,l}\|_F^2 \right)^{1/2}.$$

For each edit request, we first compute the update using the original remaining allocation and then rescale the updates at all edited layers by a shared factor

$$\alpha_t = \frac{\|\Delta_t^{\text{uniform}}\|_{\text{joint}}}{\|\Delta_t^{\text{remaining}}\|_{\text{joint}}}, \qquad \widetilde{\Delta}_{t,l}^{\text{remaining}} = \alpha_t \Delta_{t,l}^{\text{remaining}}.$$

The rescaled update is applied before the next edit request is processed. This preserves the remaining-allocation computation while matching its per-edit global update budget to the uniform reference. In this experiment, the matching error is below 3.1×10^{-16}, and all scaling factors are below one, confirming that the control primarily reduces the original remaining update rather than amplifying it.

For this mechanism experiment, we summarize the four editing metrics using $S_4 = (\text{ES} + \text{GE} + \text{PS} + \text{NS})/4$ and report the final checkpoint update norm. Table 26 shows a clear separation between allocation and update magnitude. The original remaining rule is both lower-performing and substantially more sensitive to covariance length: its S_4 decreases from 44.36 at $L = 512$ to 28.72 at $L = 2048$, a gap of 15.64 points. Uniform allocation reduces this gap to 2.72 points, but it also reduces the final update norm from roughly 87–96 to roughly 33.

Crucially, preserving the original remaining allocation while matching its update norm to the uniform condition recovers both performance and length stability. The norm-matched remaining condition reaches $S_4 = 72.14$, 70.63, and 70.01 at $L = 512$, 1024, and 2048, respectively, with a length gap of only 2.14 points. Its final update norms are also close to those of uniform allocation.

A complementary stress test gives the same directional evidence. When the uniform update is instead enlarged toward the much larger remaining-update budget, performance deteriorates sharply. Because

Condition	$L = 512$		$L = 1024$		$L = 2048$		Gap
	S_4	$\|\Delta W\|$	S_4	$\|\Delta W\|$	S_4	$\|\Delta W\|$	512→2048
Remaining	44.36	87.41	27.49	95.57	28.72	91.40	15.64
Uniform	70.64	32.51	69.74	33.22	67.91	33.50	2.72
Remaining + norm match	72.14	32.92	70.63	33.70	70.01	33.96	2.14

Table 26: MEMIT residual-allocation and norm controls. S_4 is the mean of ES, GE, PS, and NS for this diagnostic experiment. The norm-matched condition preserves the original remaining allocation while matching the per-edit joint update norm to the uniform reference.

this intervention changes the subsequent editing trajectory more aggressively, we treat it as supporting evidence rather than the primary control.

These results do not imply that layer allocation is irrelevant. Uniform and norm-matched remaining updates still produce different ES/GE/PS/NS trade-offs, and their layer-wise trajectories are not identical. They do show, however, that the apparent advantage of uniform allocation cannot be attributed to allocation alone. A substantial part of MEMIT's covariance-length sensitivity is associated with the larger effective update budget induced by the original update trajectory.

Finding. In MEMIT, residual allocation changes the editing trajectory, but update magnitude explains a substantial portion of the observed covariance-length sensitivity. Matching the original allocation to the smaller uniform update budget recovers most of the lost performance and length stability.

E.2 Update Alignment and Functional Localization of Layer-wise Drift

The previous experiment explains an important source of MEMIT's global sensitivity, but it does not explain where covariance-induced differences become expressed in the final checkpoint. Section 5 provides two complementary observations. First, global update alignment is more closely associated with editing performance than raw covariance similarity. Second, for MEMIT, the update mismatch between covariance lengths is highly localized rather than diffuse.

For 500 sequential edits, approximately 66.3% of the covariance-length mismatch is concentrated in edited layer 8, 91.3% in layers 7–8, and 99.2% in layers 6–8. The corresponding shares after 1000 edits are 64.6%, 90.9%, and 99.1%. Thus, the global difference in realized update direction is dominated by a small number of later edited layers. Localization by itself, however, does not establish functional importance: a large parameter-space difference may be a consequence of the editing trajectory without being sufficient to alter the editing endpoint.

To test whether the localized drift is functionally relevant, we construct cross-length hybrid checkpoints. Let $W^{(512)}$ and $W^{(2048)}$ denote checkpoints produced using the corresponding covariance lengths. For an edited layer l, we form

$$W^{(512\leftarrow 2048,l)} \quad \text{and} \quad W^{(2048\leftarrow 512,l)}$$

by replacing only the edited `down_proj.weight` at layer l with the corresponding tensor from the other checkpoint, leaving the remaining parameters unchanged. We perform the intervention for the high-mismatch layers identified above.

The layer-swap experiments show that these localized parameter differences are not behaviorally inert. Replacing selected high-layer edited weights changes the editing endpoint, and several swaps improve the recipient checkpoint relative to its unswapped endpoint in both swap directions. However, the effects are not symmetric across source direction and layer: a weight block that helps one recipient does not simply transfer the complete behavior of its source checkpoint. This asymmetry is important because it rules out the strongest interpretation that one "good" layer is independently responsible for the covariance-length effect.

Instead, the results support an interaction view. Covariance length changes the multi-layer editing trajectory; this trajectory produces strongly localized differences in the later edited weights; and intervening on those weights can change the final editing behavior. The high-layer concentration

Projector	Scale	Selected dimensions (layers 4–8)	S_4	ES / GE	$\|\Delta W\|$
Absolute	0.1	14335, 14334, 14335, 14334, 14334	73.59	100.0 / 80.0	14.05
Absolute	1	14320, 14307, 14292, 14286, 14275	75.61	100.0 / 77.0	12.87
Absolute	10	13786, 13359, 12998, 12886, 12595	77.93	100.0 / 76.0	13.52
Absolute	100	4526, 1727, 624, 327, 117	43.08	19.0 / 8.5	16.29
Absolute	1000	0, 0, 0, 0, 0	37.22	0.0 / 0.0	0.00
Fixed rank	all	1024, 1024, 1024, 1024, 1024	51.03	43.0 / 19.5	25.67
Frozen baseline	all	14320, 14307, 14292, 14286, 14275	75.61	100.0 / 77.0	12.87

Table 27: AlphaEdit projector controls. In the absolute condition, covariance scaling can change the selected projection dimension. Fixed-rank and frozen-projector controls remove this pathway. S_4 is the mean of ES, GE, PS, and NS for the mechanism experiment. "All" indicates that the reported result is unchanged across the tested scales.

is therefore functionally relevant, but it should be interpreted as part of a coupled multi-layer mechanism rather than as evidence for a single causal bottleneck. The intervention also replaces only `mlp.down_proj.weight`, not the complete transformer block, which further motivates this conservative interpretation.

This analysis also clarifies the role of update alignment. Alignment is useful as a global post-edit diagnostic because it summarizes whether two covariance conditions lead to similar realized updates. The layer-wise decomposition and swap intervention then refine that diagnostic by showing where the disagreement is concentrated and whether manipulating the corresponding parameter block changes behavior. Thus, update alignment identifies consequential update drift, while the layer intervention provides a more direct test of its functional localization.

Finding. MEMIT's covariance-length-induced update drift is highly concentrated in later edited layers, and cross-length swaps of these parameter blocks alter the editing endpoint. The effects are asymmetric, supporting a layer–trajectory interaction rather than a single-layer causal explanation.

E.3 AlphaEdit: Projection Construction Determines Scale Sensitivity

AlphaEdit exhibits a qualitatively different response. Over a broad range of covariance scales, its editing behavior changes much less than MEMIT's, yet sufficiently large scaling can still cause a sharp transition. This pattern suggests that covariance scale does not act on AlphaEdit in the same way as it acts on an inverse-preconditioned update. AlphaEdit uses preservation information through a projection construction, so rescaling covariance can matter indirectly by changing the effective projection subspace selected by the method.

We test this explanation with three controls. In the *absolute* condition, the projector is recomputed at every covariance scale using the original thresholding rule, so the selected projection dimension is allowed to change. In the *fixed-rank* condition, the selected dimension is fixed to 1024 directions at every edited layer. In the *frozen-projector* condition, we construct the projector once at the baseline scale and reuse exactly the same projector for all subsequent scale conditions. These controls separate the nominal covariance scale from changes in the projection regime.

Table 27 shows that AlphaEdit is stable only while the effective projector remains in a similar regime. Under the absolute construction, scales from 0.1 to 10 retain large selected dimensions at all five edited layers and maintain strong editing performance. At scale 100, the selected dimensions collapse to $(4526, 1727, 624, 327, 117)$ and edit success deteriorates sharply. At scale 1000, the selected dimension is zero at every edited layer, the realized update norm becomes zero, and the edit fails entirely. The nonzero aggregate S_4 in this condition is driven by preservation-oriented terms and should not be interpreted as successful editing.

The controls isolate the source of this transition. With fixed rank, changing scale no longer changes the result; all tested scales produce the same selected dimensions, update norm, and evaluation metrics. However, this stable fixed-rank regime performs substantially worse than the natural baseline, showing that scale invariance is not equivalent to a good projector. More decisively, when the baseline projector is frozen and reused across scales, every scale reproduces the baseline result. Thus, in the current

Condition	L	ES	GE	PS	NS	S_4	Norm
Remaining	512	76.00	68.00	14.88	18.57	44.36	87.41
Remaining	1024	54.50	44.50	5.53	5.43	27.49	95.56
Remaining	2048	51.50	46.50	5.83	11.03	28.72	91.40
Uniform	512	96.50	79.50	52.24	54.30	70.64	32.51
Uniform	1024	94.50	81.50	51.88	51.07	69.74	33.22
Uniform	2048	94.50	80.50	48.32	48.33	67.91	33.50
Remaining + norm match	512	93.50	75.50	57.44	62.13	72.14	32.92
Remaining + norm match	1024	91.50	77.50	54.93	58.60	70.63	33.70
Remaining + norm match	2048	92.50	75.00	54.52	58.00	70.01	33.96

Table 28: Llama-3.1-8B-Instruct, CounterFact, 100 sequential edits, seed 42. Scores are on a 0–100 scale; S_4 is the equal-weight mean of ES, GE, PS, and NS. Every allocation/length condition is shown. Norm matching retains the remaining allocation and matches each request's joint update norm to the independent uniform reference. Norm denotes the final checkpoint difference, not the matched per-request quantity.

L	Requests	Min. scale	Max. scale	Match error	Write error
512	100	0.5292	0.7202	2.54e-16	1.60e-04
1024	100	0.5399	0.7521	2.84e-16	1.56e-04
2048	100	0.5202	0.7452	3.09e-16	1.58e-04

Table 29: Per-edit global norm checks. Match error refers to the proposed rescaled update; write error is the maximum relative norm difference after writing the low-precision model weights. This is not final-checkpoint norm matching.

implementation, the dominant effect of covariance scaling is not an independent multiplicative effect after the projector has been fixed. It acts by changing the projector that AlphaEdit constructs from the preservation statistic.

This explains the apparent robustness observed in Section 5. AlphaEdit is not invariant to covariance in general. It is stable over a broad range in which covariance perturbations leave the effective projection regime largely unchanged. Once scaling causes a large change in selected dimension and projection subspace, the realized update and editing behavior change abruptly.

Finding. AlphaEdit's scale sensitivity is primarily mediated by projection construction. Moderate rescaling is benign while the effective projector remains similar; large rescaling becomes destructive when it changes the selected subspace, and freezing the baseline projector removes the observed scale dependence.

E.4 Main Results

Recipient L	Donor L	Layer	ES	GE	PS	NS	S_4
2048	512	6	47.50	43.50	4.58	4.87	25.11
2048	512	7	44.50	43.50	3.54	3.63	23.79
2048	512	8	48.50	46.50	3.70	4.30	25.75
512	2048	6	66.00	63.50	13.70	14.87	39.52
512	2048	7	66.50	59.50	11.72	13.97	37.92
512	2048	8	62.00	57.50	9.19	10.07	34.69

Table 30: Llama-3.1-8B-Instruct, CounterFact, 100 sequential edits, seed 42. Scores are on a 0–100 scale; S_4 is the equal-weight mean of ES, GE, PS, and NS. Exploratory single-layer swaps of the edited down-projection weight.

Projector	Condition	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8
absolute	baseline	14320	14307	14292	14286	14275
absolute	s=0.1	14335	14334	14335	14334	14334
absolute	s=1	14320	14307	14292	14286	14275
absolute	s=10	13786	13359	12998	12886	12595
absolute	s=100	4526	1727	624	327	117
absolute	s=1000	0	0	0	0	0
fixed rank	baseline	1024	1024	1024	1024	1024
fixed rank	s=0.1	1024	1024	1024	1024	1024
fixed rank	s=1	1024	1024	1024	1024	1024
fixed rank	s=10	1024	1024	1024	1024	1024
fixed rank	s=100	1024	1024	1024	1024	1024
fixed rank	s=1000	1024	1024	1024	1024	1024
frozen baseline	baseline	14320	14307	14292	14286	14275
frozen baseline	s=0.1	14320	14307	14292	14286	14275
frozen baseline	s=1	14320	14307	14292	14286	14275
frozen baseline	s=10	14320	14307	14292	14286	14275
frozen baseline	s=100	14320	14307	14292	14286	14275
frozen baseline	s=1000	14320	14307	14292	14286	14275

Table 31: Exact selected dimensions for all 18 projector conditions, in the same row order as the editing metrics. Dimensions are not rounded projector traces.

Projector	Condition	ES	GE	PS	NS	S_4	Norm
absolute	baseline	100.00	77.00	63.25	62.20	75.61	12.87
absolute	s=0.1	100.00	80.00	62.18	52.17	73.59	14.05
absolute	s=1	100.00	77.00	63.25	62.20	75.61	12.87
absolute	s=10	100.00	76.00	64.28	71.43	77.93	13.52
absolute	s=100	19.00	8.50	53.17	91.63	43.07	16.29
absolute	s=1000	0.00	0.00	48.88	100.00	37.22	0.00
fixed rank	baseline	43.00	19.50	55.66	85.97	51.03	25.67
fixed rank	s=0.1	43.00	19.50	55.66	85.97	51.03	25.67
fixed rank	s=1	43.00	19.50	55.66	85.97	51.03	25.67
fixed rank	s=10	43.00	19.50	55.66	85.97	51.03	25.67
fixed rank	s=100	43.00	19.50	55.66	85.97	51.03	25.67
fixed rank	s=1000	43.00	19.50	55.66	85.97	51.03	25.67
frozen baseline	baseline	100.00	77.00	63.25	62.20	75.61	12.87
frozen baseline	s=0.1	100.00	77.00	63.25	62.20	75.61	12.87
frozen baseline	s=1	100.00	77.00	63.25	62.20	75.61	12.87
frozen baseline	s=10	100.00	77.00	63.25	62.20	75.61	12.87
frozen baseline	s=100	100.00	77.00	63.25	62.20	75.61	12.87
frozen baseline	s=1000	100.00	77.00	63.25	62.20	75.61	12.87

Table 32: Llama-3.1-8B-Instruct, CounterFact, 100 sequential edits, seed 42. Scores are on a 0–100 scale; S_4 is the equal-weight mean of ES, GE, PS, and NS. Calibration length 2048; projectors are constructed and loaded in FP32. All 18 conditions are reported individually, including baseline and scale one.

Condition	ES	GE	PS	NS	S_4	$\|\Delta W\|_F$	Norm matched
Absolute baseline	100.00	79.00	62.90	61.23	75.78	12.8486	No
Matched-rank covariance	100.00	78.00	61.81	61.67	75.37	12.8178	No
Identity ($P = I$)	100.00	81.00	58.56	49.27	72.21	14.3303	No
Random matched-rank, seed 11	100.00	83.00	58.14	49.50	72.66	14.2835	No
Random matched-rank, seed 22	100.00	80.00	59.77	49.03	72.20	14.3220	No
Random matched-rank, seed 33	100.00	82.00	63.09	49.33	73.61	14.2777	No
Matched-rank covariance	100.00	78.00	61.11	61.83	75.24	12.8493	Yes
Identity ($P = I$)	100.00	80.00	61.57	51.63	73.30	12.8789	Yes
Random matched-rank, seed 11	100.00	79.00	58.42	50.83	72.06	12.8769	Yes
Random matched-rank, seed 22	100.00	82.00	61.05	51.87	73.73	12.8782	Yes
Random matched-rank, seed 33	100.00	80.00	59.82	51.20	72.75	12.8785	Yes

Table 33: AlphaEdit per-request joint update-norm controls on Llama-3.1-8B-Instruct with 100 sequential CounterFact edits (seed 42, calibration length 2048). Scores are percentages; S_4 is the equal-weight mean of ES, GE, PS, and NS. The norm-matched rows match the absolute-baseline joint update norm separately for each request. $\|\Delta W\|_F$ is the final checkpoint difference and is not the per-request matching target.

Condition	Min scale	Max scale	Max applied error	Max realized error
Matched-rank covariance	0.9293	1.0524	5.74×10^{-8}	7.00×10^{-4}
Identity ($P = I$)	0.7619	1.0424	5.66×10^{-8}	7.41×10^{-4}
Random matched-rank, seed 11	0.7723	1.0441	4.13×10^{-8}	7.03×10^{-4}
Random matched-rank, seed 22	0.7962	1.0461	3.98×10^{-8}	6.85×10^{-4}
Random matched-rank, seed 33	0.8012	1.0480	5.52×10^{-8}	6.92×10^{-4}

Table 34: Per-request joint norm-matching diagnostics for the AlphaEdit controls. Applied error is measured before low-precision weight storage; realized error is measured after writing the model weights. The reference is the absolute-baseline proposed joint update norm at each request.

Condition	ES	GE	PS	NS	S_4	$\|\Delta W\|_F$	Energy fraction f
Absolute baseline	100.00	77.00	63.25	62.20	75.61	12.8690	–
Energy-matched random, $f = 0.25$, seed 11	100.00	82.00	60.23	57.33	74.89	13.2610	0.25
Energy-matched random, $f = 0.25$, seed 22	100.00	83.00	62.41	56.57	75.49	13.2955	0.25
Energy-matched random, $f = 0.25$, seed 33	100.00	81.00	62.81	56.57	75.09	13.2981	0.25
Energy-matched random, $f = 0.50$, seed 11	100.00	83.00	63.47	53.80	75.07	13.7062	0.50
Energy-matched random, $f = 0.50$, seed 22	100.00	80.00	61.05	53.40	73.61	13.7859	0.50
Energy-matched random, $f = 0.50$, seed 33	100.00	80.00	61.35	53.30	73.66	13.7298	0.50

Table 35: AlphaEdit random projectors constructed at common feasible covariance-energy targets on Llama-3.1-8B-Instruct with 100 sequential CounterFact edits (seed 42, calibration length 2048). Scores are percentages; S_4 is the equal-weight mean of ES, GE, PS, and NS. For a fixed layer-wise rank, the covariance-selected projector is the minimum-energy endpoint; therefore f interpolates between this endpoint and the expected energy of a random rank-matched projector, rather than matching the baseline energy exactly.

F Implementation and Reproduction Details

In this section, we would like to provide details for reproducing our experimental results.

F.1 Code Base

Here, we list the code base used in our paper.

- For all of the models, we use HuggingFace Transformers as the default https://github.com/huggingface/transformers.
- For editing language models, we use the EasyEdit framework https://github.com/zjunlp/EasyEdit.
- For model evaluation, we use the code and data from chain-of-thought hub https://github.com/FranxYao/chain-of-thought-hub and Language Model Evaluation Harness https://github.com/EleutherAI/lm-evaluation-harness.
- For accelerating model evaluation on GPU, we use the vLLM framework https://github.com/vllm-project/vllm.
- For general ability, we use the lm evaluation harness in https://github.com/EleutherAI/lm-evaluation-harness.

Our code will be available once accepted.

F.2 Models

Here, we list all of the model checkpoints used in our experiments.

- Qwen/Qwen2.5-7B-Instruct https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
- meta-llama/Llama-3.1-8B-Instruct https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
- meta-llama/Llama-3.1-8B https://huggingface.co/meta-llama/Llama-3.1-8B
- meta-llama/Llama-3.2-3B-Instruct https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

F.3 Hyperparameter Settings

In editing experiments, our hyperparameters are in accord with the EasyEdit framework at https://github.com/zjunlp/EasyEdit/tree/main/hparams.

F.4 Running Devices

All of our experiments in this paper are running on a server with 2 RTX A6000 GPUs with 48GB VRAM.