Moir: Let the Model Direct Its Own Story for Robust Cross-Domain Knowledge Editing

Jea Kwon [1] **Jiwon Kim** [1] **Dong-Kyum Kim** [1] **Meeyoung Cha** [1]

Abstract

While language models remain frozen at their training state, the world evolves continuously. Knowledge editing has emerged as a key alternative to full retraining, but its deployment is bottlenecked by the erosion of core capabilities: mathematical and programmatic reasoning collapse while encyclopedic recall remains intact. We trace this asymmetric degradation to a distributional mismatch. Covariance-based editors preserve only the subspaces spanned by their reference corpus, but fail to capture the operative distribution shaped by post-training such as SFT and DPO. Static external corpora, including Wikipedia and even the original pretraining mixture, cannot recover this shifted manifold. We propose MOIR, which estimates the preservation covariance C directly from the model itself by sampling from its own decoding distribution. Seeding generation with a single random vocabulary token bypasses the instruction-following templates that otherwise dominate sampled outputs, exposing the broader subspaces the model has internalized. MOIR requires no external data and serves as a drop-in component for any covariance-based editor, a practical advantage given that the pre- and post-training corpora of most modern LLMs are not publicly accessible. Across OLMo-2, Llama-3.1, and Qwen-3 (7–8B), under both MEMIT and AlphaEdit and in batch and sequential regimes, MOIR consistently extends preservation in the most vulnerable domains, most strikingly on Qwen3-8B after 20,000 AlphaEdit batch edits, it retains 79.9% GSM8K accuracy compared to 10.9% with the Wikipedia baseline. These results suggest that aligning the preservation distribution with the model's operative distribution is a key factor in non-destructive editing, and that the model itself may be the most accessible source of that distribution for deployed systems.

[1]Max Planck Institute for Security and Privacy (MPI-SP), Bochum, Germany. Correspondence to: Meeyoung Cha <mia.cha@mpi-sp.org>.

Mechanistic Interpretability Workshop at the 43^{rd} International Conference on Machine Learning, Seoul, South Korea, 2026.

1. Introduction

Large language models (LLMs) act as static repositories of knowledge in a fundamentally dynamic world. While global events, scientific breakthroughs, and socio-political landscapes evolve continuously, the internal state of an LLM remains frozen at the conclusion of its training. Because full retraining to incorporate new information is computationally and environmentally prohibitive, *model editing* has emerged as a critical paradigm for incorporating precise, localized updates to specific facts (Geva et al., 2021; Mitchell et al., 2022a).

The central challenge in model editing lies in maintaining a delicate balance: successfully integrating new information without compromising the model's hard-earned structural integrity. Past research in this field has focused on the mechanics of the update, moving from *how to edit* (Meng et al., 2022a; 2023) to *how to preserve* existing parameters (Fang et al., 2025). This work introduces a new perspective. To achieve truly non-destructive editing, we show that the community must address an equally important, yet largely unexamined question: ***what exactly should be preserved?***

Current structure-preserving algorithms (Fang et al., 2025; Gu et al., 2024b) implicitly answer this question by using arbitrary proxy corpora, such as Wikipedia, to define the preserved knowledge space. In this work, we demonstrate that this deeply ingrained convention introduces a critical misalignment, causing a severe and disproportionate collapse of complex capabilities that lie outside this proxy's narrow distribution (e.g., mathematical reasoning and code generation) during the editing process. These implications extend beyond editing itself. As deployed models increasingly rely on editing for factual patching, content removal, and continuous updates, any intervention that inadvertently degrades core capabilities becomes a significant barrier to deployment as reliable infrastructure

We propose a shift in how the preservation space is defined through the following contributions:

- **Diagnosing the proxy pitfall.** We empirically show that arbitrary proxy corpora drive the cross-domain collapse observed in prior editing methods.
- **Self-generated manifolds.** We introduce MOIR, a data-free framework that approximates this distribution directly from the model's own generations (see Figure 1).
- **Inaccessibility as motivation.** We demonstrate that the operative distribution is generally inaccessible in deployed LLMs. Consequently, self-generation is presented not merely as a preference but as the only viable approach for maintaining the relevant distribution.

We introduce an approach that utilizes a model's own decoding distribution to access its internalized training data and post-training shifts without relying on external corpora. Empirically, we identify single-random-token seeding as the configuration that best balances mode-collapse mitigation and prefix-prior fidelity. Additionally, we show that aligning the preservation space with this self-generated manifold reduces the cross-domain degradation typical of external proxies. By applying MOIR, we achieve unprecedented stability across major LLMs and maintain complex reasoning manifolds.

MOIR is a drop-in replacement for any covariance-based editor, requiring no architectural changes to existing frameworks like MEMIT or AlphaEdit (Meng et al., 2023; Fang et al., 2025). By sampling directly from the live model, it captures the nuanced activation shifts induced by post-training, such as Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO), that static corpora fail to represent. This alignment enables large-scale, non-destructive editing across the OLMo-2, Llama-3.1, and Qwen-3 families. The resulting stability is substantial: the tested models sustain 79.9% accuracy on GSM8K after 20,000 edits, whereas Wikipedia baselines collapse to 10.9%. This stability is achieved at a manageable one-time generation cost of approximately 2 hours per model.

The implications reach beyond knowledge editing as a research problem. Once edits no longer cost a model its reasoning or coding ability, full retraining is no longer the only path to keeping a model factually up-to-date. A deployed model can instead evolve continuously, absorbing corrections in minutes rather than months, and stay in sync with the world for as long as it runs. What has been a static artifact, frozen at the end of training, becomes something closer to maintainable infrastructure.

2. Related Work

The Evolution of Editing Paradigms. Knowledge editing methods can be grouped based on whether they preserve the model's original parameters or not. Parameter-preserving approaches (Hu et al., 2022; Mitchell et al., 2022b; Hartvigsen et al., 2023) route edits through external adapters or memory modules. While safe by construction, they incur inference overhead, struggle with complex reasoning, and cannot truly remove information. Direct parameter-modifying methods (Mitchell et al., 2022a; Meng et al., 2022b; 2023) instead overwrite weights to internalize new facts. The cost is interference: representations in language models are entangled, so localized updates leak into neighboring activations and, under sequential editing, compound into catastrophic forgetting and model collapse. Recent structure-preserving editors (Fang et al., 2025; Ma et al., 2024; Gu et al., 2024b) address this by projecting updates onto subspaces deemed orthogonal to existing knowledge.

Out-of-Domain Capability Erosion. Recent stress tests expose a blind spot in editing evaluation: the standard locality metric draws held-out triples from the same distribution as the edits and thus cannot detect a collapse in capabilities lying outside it. Gu et al. (2024b) shows that a single editing pass modifying less than one percent of LLaMA-1's parameters drives out-of-domain accuracy to near zero, and Li et al. (2024b) reports that ten thousand sequential edits push the model into a "muting effect" that collapses broad capabilities in parallel. The damage is most severe in specialized domains: sequential MEMIT edits collapse mathematical problem-solving to zero (Gu et al., 2024a; Lin et al., 2024), and the same methods reduce syntactic validity on code-specialized models by up to 86 points (Chhetri et al., 2025). Concurrent work by Liu et al. (2026) formalizes this as a "Covariance Trap," attributing the failure to the projection metric itself.

3. Rethinking the Preservation Space

3.1. Background: Covariance-Constrained Knowledge Editing

To establish the mathematical role of the preservation space, we first formalize the objective of sequential model editing. Given a residual matrix R that encodes target values, an algorithm computes a parameter update ΔW to memorize new facts associated with keys K_{new}. This update must simultaneously preserve previously edited facts (K_{past}) and foundational pre-training knowledge (K_0) sampled from a proxy distribution $\mathcal{D}$. Both foundational and state-of-the-art methods rely on K_0 to construct their preservation spaces.

Covariance-Constrained Formulations. Modern editing methods rely on the uncentered covariance matrix

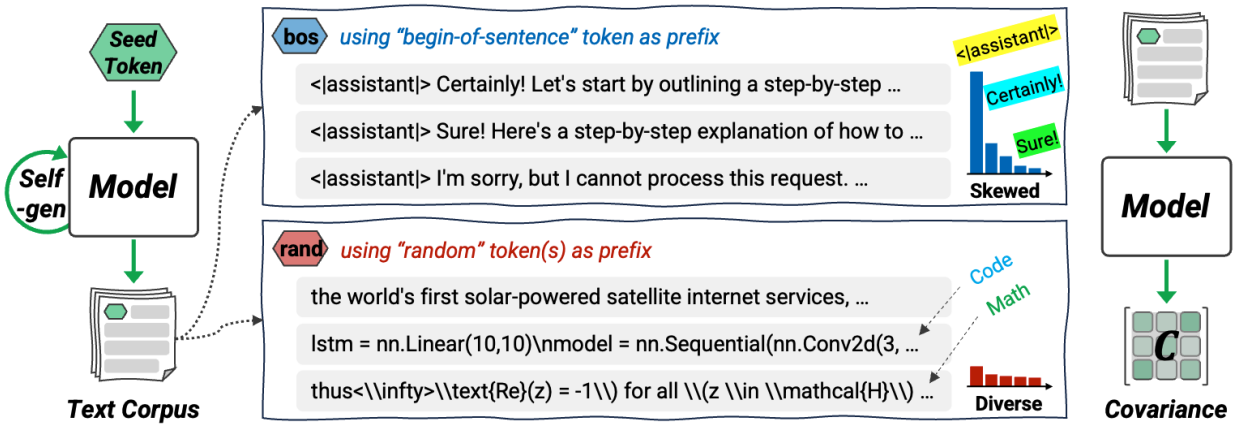


Figure 1. **MOIR's self-generation pipeline.** The model autoregressively generates samples from a seed token, from which C is estimated. Using begin-of-sentence token (top) collapses generations skewed toward an instruction-style, whereas a random token (bottom) diversifies generations across domains (e.g., general text, code, and math) better approximating the model's internalized distribution.

$C = K_0 K_0^\top$ to define the preservation space, albeit through slightly different implementations. MEMIT (Meng et al., 2023) incorporates C as a soft regularization penalty, computing updates proportional to $(\ldots + C)^{-1}$. Conversely, AlphaEdit (Fang et al., 2025) enforces a hard constraint by projecting updates strictly into the null space of C. Applying SVD to C yields an orthogonal projection matrix P that structurally guarantees $PK_0 = \mathbf{0}$. Despite these algorithmic differences, both methods are fundamentally bottlenecked by the same geometric commitment: the distribution $\mathcal{D}$ used to sample K_0 and estimate C. Please refer to Appendix C for the full mathematical formulations of these parameter updates.

From "How" to "What" to Preserve Recent structure-preserving model editing methods (Fang et al., 2025; Ma et al., 2024; Gu et al., 2024b) have achieved significant gains by refining *how* parameter displacement is constrained. Yet, as illustrated in Figure 2, we observe a catastrophic erosion of specialized knowledge regardless of the preservation mechanism employed. While general linguistic performance is maintained, out-of-domain capabilities, specifically for mathematics and coding, collapse to near-zero levels far more rapidly.

This convention introduces a structural bias. Modern pretraining corpora such as Dolma (Soldaini et al., 2024) and RedPajama (Together Computer, 2023) are highly heterogeneous, whereas Wikipedia is restricted to encyclopedic prose. A covariance matrix estimated from such a narrow proxy spans only the activation subspaces relevant to factual recall, leaving directions critical for these specialized domains unconstrained. Consequently, editing algorithms that rely on this incomplete C unintentionally distort the unprotected subspaces, leading to the rapid degradation illustrated in Figure 2 and quantified in Table 1.

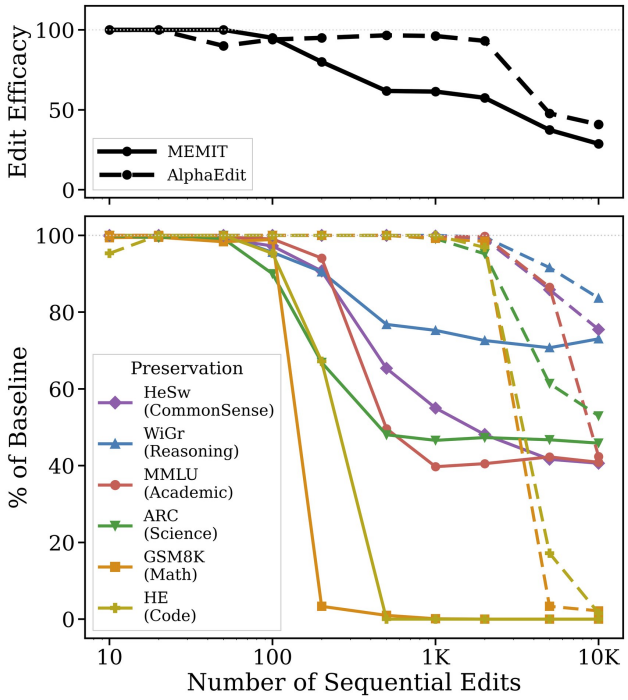


Figure 2. Differential degradation rates across knowledge domains during sequential editing on Counter Factual data with OLMo-2-Inst.

Why C Alone Determines the Update For the closed-form covariance-constrained editors examined here, the parameterupdate depends on the input distribution only through the uncentered covariance C; the output-side factor G cancels in the closed-form optimum. This is not an assumption but a consequence of the cancellation established in proposition 1 (with MEMIT requiring the soft-regularization limit per Remark 1).

Table 1. Impact of C source distribution on OLMo-2-Instruct under MEMIT batch editing. Aligning C with OLMo-2's actual pretraining corpus (OLMo-Mix-1124) substantially mitigates cross-domain collapse compared to the conventional Wikipedia proxy (see evaluation metrics in C.3). Values in parentheses denote the absolute improvement ($+\Delta$) over the Wikipedia baseline in the most vulnerable domains (Code and Math). Oracle construction details are provided in Appendix A.2. $N = 2{,}000$ edits

	Edit Metrics							Preserve Metrics					
C Source	ES	PS	Effi	Gen	Spec	Flu	Con	HeSw	WiGr	MMLU	ARC	HE	GSM8K
Conventional proxy													
Wikipedia	.961	.831	.830	.443	.481	.946	.258	.776	.672	.527	.505	.140	.352
Pretraining-aligned distribution (OLMo-2)													
Dolma 1.7 (Wiki subset only)	.950	.791	.789	.383	.462	.951	.245	.769	.657	.550	.501	.232 (+.092)	.378 (+.026)
OLMo-Mix (pretrain-oracle)	.942	.796	.767	.386	.455	.961	.242	.774	.676	.552	.521	**.238** (+.098)	**.485** (+.133)

Beyond the pretraining oracle. A natural hypothesis is that the optimal C should be derived from the model's original pre-training mixture. We test this by constructing an "Oracle" covariance using the official OLMo-Mix-1124 corpus, sampled at exact pretraining proportions (Table 1). Relative to the Wikipedia baseline, the pretraining-aligned oracle substantially improves preservation of specialized capabilities at a modest cost to edit quality. Surprisingly, a self-generated distribution of the kind we propose is highly competitive with and frequently surpasses even this oracle (Table 9).

As shown in Figure 3, post-training procedures (e.g., SFT, DPO) shift the model's operative manifold away from its raw pretraining distribution until the data on disk no longer matches the internal representations encoded in the weights. Any static external corpus, regardless of its historical relevance, incurs an inherent geometric bias. A principled way to capture this shifted distribution is to elicit it directly from the live model via self-generation, a data-free framework we propose next.

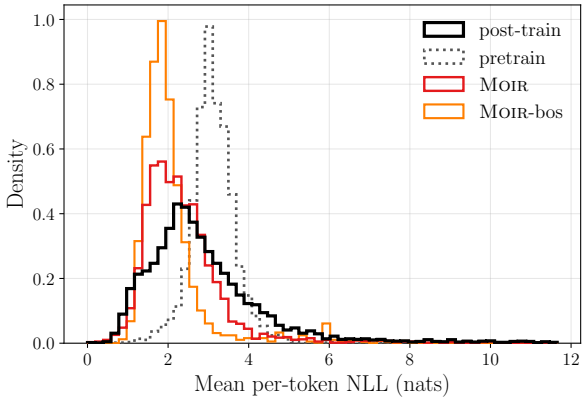


Figure 3. **Tracking the Post-Training Manifold.** Per-document NLL distributions demonstrating the clear leftward shift from the pretraining to the post-training regime. MOIR ($\langle\texttt{rand}\rangle \times 1$, red) recovers this shifted operative distribution, visibly achieving a tighter match than the $\langle\texttt{bos}\rangle$-only variant (orange).

4. MOIR: Self-generated Covariance Estimation

MOIR (**M**emories **O**f **I**nternal **R**epresentations) elicits the preservation distribution from the model itself. Rather than estimating C from an external corpus, we construct it from samples produced by the model under its own decoding distribution. The estimator follows the standard formulation:

$$C_{\text{MOIR}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{k}_i\,\mathbf{k}_i^\top, \tag{1}$$

where each key $\mathbf{k}_i$ is the input to the targeted MLP layer during the forward pass over a sequence sampled from $p_\theta(\cdot \mid s)$, and N is the total number of token positions across sequences (see Appendix A.1 for the full extraction procedure). Within this framework, the primary design choice distinguishing MOIR from prior methods is the specification of the seed sequence s.

Self-generation Strategy. The seed sequence s serves as the entry point into the model's latent manifold. While conventional methods rely on a static $\langle$bos$\rangle$ prefix, such a choice is overly deterministic and limits the generation to high-probability instruction-following paths. By setting s as a single uniformly-random vocabulary token ($\langle$rand$\rangle \times 1$), we ensure that the resulting activations are sufficiently diverse to represent the model's broad operative distribution while remaining grounded in the model's learned k-gram statistics (see Table 2).

Prefix Length Controls Diversity vs. Fidelity. The empirical results in Table 2 confirm that the choice of seed s is critical for representative sampling. $\langle\texttt{rand}\rangle \times 1$ achieves the highest total performance (1.093, +0.087 gain over $\langle\texttt{bos}\rangle$), whereas longer random prefixes lead to a gradual degradation toward the baseline ($\langle\texttt{rand}\rangle \times 4$: 1.049). These results validate our hypothesis: while a single random token is sufficient to break the mode collapse associated with $\langle$bos$\rangle$, excessive randomization steers generation away from coherent linguistic structures and the model's true operative activation statistics. We adopt $\langle\texttt{rand}\rangle \times 1$ as the default in all subsequent experiments.

Distributional Analysis. To characterize the text generated by our framework, we analyze its per-document Negative Log-Likelihood (NLL) (Eq. 21) (Shumailov et al., 2024) against the model's pre- and post-training corpora. Here, NLL quantifies how probable a document is under the model's current weights; comparing these NLL distributions allows us to verify if the generated text spans the same probability landscape as the post-training data.

Figure 3 reveals a stark gap between raw pretraining data (dotted black) and post-training (SFT/DPO) data (solid black). The conventional Wikipedia proxy is anchored to the pretraining distribution (Figure 11 in Appendix D.3), meaning it completely misses the structural shifts induced by alignment tuning. In contrast, by generating text directly from the live model, MOIR naturally captures this shifted post-training distribution (orange and red in Figure 3).

Crucially, our $\langle \texttt{rand} \rangle \times 1$-prefix strategy (MOIR, red) matches this target distribution significantly better than the $\langle \texttt{bos} \rangle$-only variant (MOIR-bos, orange). This is because $\langle \texttt{bos} \rangle$-seeding forces the model into repetitive, mode-collapsed chat templates (e.g., starting with "Certainly!", "Sure!"), whereas seeding with a single random token diversifies the output to cover the model's true breadth of knowledge (see Figure 10 in Appendix D). By avoiding this mode collapse, MOIR provides a highly accurate, data-free representation of the deployed model's true operative distribution.

5. A Closer Look at the Preservation Distribution

5.1. A K-FAC View of Closed-Form Editors

Why a K-FAC view. The preservation matrix $C = K_0 K_0^\top$ that MEMIT and AlphaEdit treat as a regularizer or projector is, structurally, the empirical estimator of the K-FAC input factor $\Sigma_\ell(p) = \mathbb{E}_{x\sim p}[\phi_\ell(x)\phi_\ell(x)^\top]$ under whatever distribution p supplies the samples K_0. Reading C this way reframes covariance-constrained editing as an input-side projection under the Fisher metric induced by p: the editor commits to a Riemannian geometry on the activation space, and the choice of p is the choice of metric. Two consequences fall out of this reading and motivate Proposition 1 below. First, because the constraint $\Delta W K_{\text{new}} = R$ leaves no output-side degrees of freedom, the output factor $G_\ell(p)$ cannot affect the closed-form solution—the metric on the output side is inert. Second, the input factor $\Sigma_\ell(p)$ is the *only* distributional object the editor sees.

Let $\phi_\ell : \mathcal{X} \to \mathbb{R}^{d_\ell}$ denote the activation map to the targeted MLP at layer ℓ, and let $\Sigma_\ell(p) := \mathbb{E}_{x\sim p}[\phi_\ell(x)\phi_\ell(x)^\top]$ and $G_\ell(p) := \mathbb{E}_{x\sim p}[g_\ell(x)g_\ell(x)^\top]$ denote the K-FAC input and output factors, where g_ℓ is the output-side gradient.

Proposition 1 (G-independence of closed-form covariance–constrained editing)**.** *Let $A \in \mathbb{R}^{d_{\text{in}}\times d_{\text{in}}}$ be positive definite and $G \in \mathbb{R}^{d_{\text{out}}\times d_{\text{out}}}$ be positive semi-definite, and let $K_{\text{new}} \in \mathbb{R}^{d_{\text{in}}\times m}$, $R \in \mathbb{R}^{d_{\text{out}}\times m}$. The constrained minimum-norm problem*

$$\min_{\Delta W} \operatorname{vec}(\Delta W)^\top (A \otimes G) \operatorname{vec}(\Delta W) \quad \textit{s.t.} \quad \Delta W\, K_{\text{new}} = R \tag{2}$$

admits the unique optimum $\Delta W^\star = R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1} K_{\text{new}}^\top A^{-1}$, which is independent of G. The soft-penalty (MEMIT) and null-space-projection (AlphaEdit) formulations of Appendix B adopt the same input-only design by construction: in all three cases, ΔW depends on the input distribution only through $A_\ell = \Sigma_\ell(p)$.

Remark 1 (Specialization of MEMIT and AlphaEdit to the canonical form)**.** *MEMIT and AlphaEdit reduce to the canonical form of proposition 1 through two distinct mechanisms, each preserving A-only dependence.*

(i) MEMIT (penalty → constraint). Applying the matrix push-through identity to the Tikhonov-regularized update (Eq. 7) and taking $\lambda \to 0^+$ on the well-posed branch ($K_{\text{new}}^\top C^{-1} K_{\text{new}} \succ 0$, $K_{\text{past}} = 0$) yields

$$\Delta W_{\text{MEMIT}} \longrightarrow R\left(K_{\text{new}}^\top C^{-1} K_{\text{new}}\right)^{-1} K_{\text{new}}^\top C^{-1},$$

the $A = C$ specialization of proposition 1. For finite λ, MEMIT is the soft analogue of the same problem; G appears in neither form.

(ii) AlphaEdit (hard projection). Replacing the soft penalty with the projector P onto $\ker(C)$ enforces $\Delta W\, K_0 = 0$ structurally, applying proposition 1 with the feasible set restricted to $\operatorname{range}(P)$.

In both cases, ΔW depends on the input distribution only through $C = \Sigma_\ell(p)$.

Across all three editors, ΔW therefore depends on the input distribution *only* through $A_\ell = \Sigma_\ell(p)$.

5.2. Which Distribution? An Information-Geometric View with Hypotheses

Given proposition 1, the only distributional object the editor sees is the input covariance $\Sigma_\ell(p)$, which we treat as the empirical proxy for the geometry of the model's operative manifold p_θ. The remaining question is which sampling distribution q best matches p_θ at the level of activations.

H1: Static external corpora are confined to a narrow chart of the manifold. A pretraining proxy such as Wikipedia places nearly all of its mass on the encyclopedic chart of the model's internalized manifold; charts corresponding to code, mathematical reasoning, or instruction-following receive vanishing density. A covariance estimated

Table 2. Effect of self-generation C quality by different seed token. Four random-prefix variants ($\langle\texttt{rand}\rangle \times 1$–$\langle\texttt{rand}\rangle \times 4$) are compared against ($\langle\texttt{bos}\rangle$). Both overall edit score (S_e) and preservation scores (S_p) are measured by harmonic mean of metrics (see Appendix C.3 for equations) across $N \in \{1\text{K}, 2\text{K}, 3\text{K}, 5\text{K}\}$ edits; Total = Edit Avg. + Pres. Avg.; OLMo-2-7B-Inst.

Seed Token	Edit HM (S_e)					Pres. HM (S_p)					Total
	1K	2K	3K	5K	Avg	1K	2K	3K	5K	Avg	
$\langle\texttt{rand}\rangle \times 1$	.564	.508	.473	.399	.486	.908	.810	.555	.155	**.607**	**1.093**
$\langle\texttt{rand}\rangle \times 2$	.563	.531	.501	.404	**.500**	.930	.735	.571	.077	.578	1.078
$\langle\texttt{rand}\rangle \times 3$	.570	.506	.483	.379	.484	.942	.791	.466	.100	.575	1.059
$\langle\texttt{rand}\rangle \times 4$	.568	.513	.481	.373	.484	.931	.801	.511	.019	.565	1.049
$\langle\texttt{bos}\rangle$	.550	.495	.443	.360	.462	.932	.710	.492	.041	.544	1.006

from such a corpus is therefore not only narrow in *support* but *tilted* in mass: the directions the model actually uses for these domains are averaged in at near-zero weight, regardless of sample size. Self-generation, by drawing from p_θ directly, allocates density across whichever charts the model itself has internalized, without requiring us to identify those charts in advance.

H2: BOS-seeding collapses onto the post-training attractor. Post-training installs a strong prior at the start of generation. Conditioned on $\langle$BOS$\rangle$, the model funnels into a small set of instruction-following openings, so the resulting sample distribution is sharply concentrated rather than broad. This constitutes a different failure mode from H1: where Wikipedia is biased in *topic*, BOS-seeding is biased in *form*, sampling the manifold only along the few trajectories that survive the chat-template attractor.

H3: A single random token escapes the attractor at low geometric cost. A uniformly drawn first token rarely sits on the natural prefix manifold, since many tokens are syntactically marginal or semantically vacuous. Yet this small off-manifold perturbation dislodges generation from the BOS basin: after one or two autoregressive steps, the trajectory re-enters the model's natural distribution, seeded into a region the BOS prior would never visit, a small fidelity cost for a large gain in exploration.

H4: Longer random prefixes drift into a sparse region of the product space. For a length-k random seed, the prefix lives in $\mathcal{V}^k$, whose meaningful subset shrinks combinatorially with k. Most points in $\mathcal{V}^k$ are nonsense k-grams that the model has never seen during training, so conditioning on such a prefix steers generation into low-density regions of p_θ. Unlike the single-token case, the autoregressive process cannot fully recover here, because the prefix bias propagates forward through the context. The exploration benefit thus saturates after $k = 1$ while the off-manifold drift compounds, and we expect a sweet spot at $k = 1$: enough randomness to break the attractor, but not enough to leave the manifold.

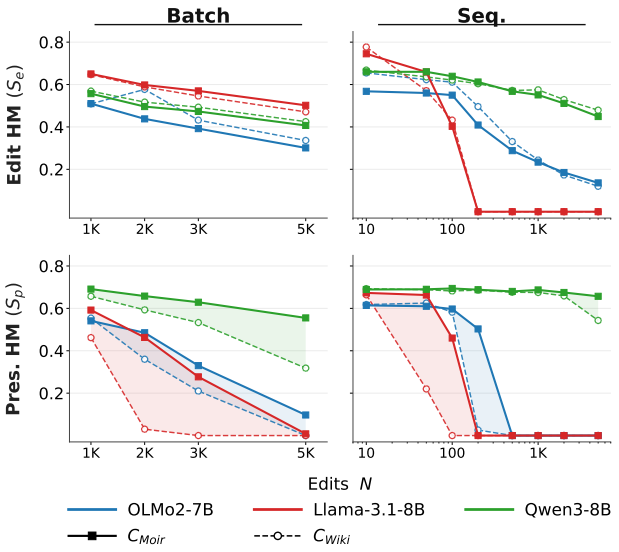


Figure 4. C_{MOIR} Improves Preservation. Harmonic means over edit metrics (top) and preservation tasks (bottom), under MEMIT batch (left) and sequential (right) editing. Shades in bottom row shows the improved preservation capacity.

6. Generality Across Editors, Models, and Regimes

We evaluate MOIR as a drop-in replacement for C in two representative editors: MEMIT (Meng et al., 2023) (soft preservation penalty) and AlphaEdit (Fang et al., 2025) (hard null-space projection). Our evaluation covers both batch and sequential regimes on three open-weight instruction-tuned models: OLMo-2-7B-Instruct (Groeneveld et al., 2024), Llama-3.1-8B-Instruct (Grattafiori et al., 2024), and Qwen3-8B (Bai et al., 2023). The editing score (S_e) and preservation score (S_p) are reported as harmonic means over five standard COUNTERFACT editing metrics and six downstream benchmarks (MMLU, GSM8K, HellaSwag, WinoGrande, ARC-Challenge, HumanEval). The full setup, including editing layers, hyperparameters, and metric definitions, is provided in Appendix C.

Consistency across editors, models, and scales. Figures 4 and 12 compare C_{MOIR} against the standard C_{Wiki} across all editor-regime combinations (MEMIT/AlphaEdit $\times$ batch/sequential) and models, as well as edit counts spanning four orders of magnitude. Two patterns hold throughout. One is that *editing quality is preserved.* Across all settings (top row), C_{MOIR} matches or exceeds C_{Wiki} in editing performance. The largest gap appears under MEMIT sequential editing on Llama-3, where C_{Wiki} collapses to zero within $\sim$200 edits while C_{MOIR} sustains non-trivial editing performance throughout. Where C_{Wiki} remains stable, C_{MOIR} tracks it closely; the self-generated covariance is therefore not a trade-off against editing quality but a strict improvement on average.

Another pattern is that *out-of-domain preservation is dramatically extended.* The bottom row shows the central finding. Under MEMIT batch editing, C_{Wiki} degrades roughly linearly with edit count and reaches near-zero preservation at 5×10^3 edits on Llama-3, while C_{MOIR} retains 0.3–0.5 Preservation HM at the same scale. Under MEMIT sequential editing, the gap is more extreme: C_{Wiki} collapses to zero within 10^2–10^3 edits across all three models, while C_{MOIR} sustains 0.5–0.7 preservation throughout 10^4 edits. AlphaEdit exhibits the same pattern in attenuated form: C_{MOIR} maintains preservation flat across the full edit range, while C_{Wiki} on Qwen-3 degrades sharply beyond 10^3 sequential edits.

Where the preservation gap lives. Figure 5 and 13 shows task-wise preservation capacity (shades). Figure 6 resolves this by showing the per-benchmark preservation profile on Qwen-3-8B across three editing settings, overlaid on the pre-edit baseline. The collapse is sharply asymmetric: C_{Wiki} holds encyclopedic capabilities (MMLU, WinoGrande) near baseline while visibly shrinking the polygon inward on the most distributionally distant axes: HumanEval and GSM8K. C_{MOIR} closes this gap. GSM8K rises from .06 to .78 on AlphaEdit batch editing and from .32 to .92 on AlphaEdit sequential, with similar recovery on HumanEval, while the encyclopedic axes that C_{Wiki} already preserves remain preserved. The improvement, therefore, comes from extending coverage into capability subspaces that C_{Wiki} never protected, not from redistributing it.

The Nature of Domain Collapse. It is important to note that the near-zero performance on GSM8K and HumanEval under C_{Wiki} represents a comprehensive collapse of these specific domains. Whether this degradation manifests as a loss of underlying logic or the destruction of domain-specific formatting (e.g., code syntax, step-by-step mathematical templates), the practical utility of these subspaces is entirely compromised. C_{Wiki} fails to protect the structural integrity of these out-of-domain manifolds, whereas MOIR preserves the exact distributional footprint required to maintain them.

Why the gap widens with scale. Both editors accumulate parameter perturbations ΔW as edits proceed; the role of C is to constrain how those perturbations project onto the activation space. When C misses entire capability subspaces (as C_{Wiki} does for code and math, per Table 1 and Figure 6) the cumulative perturbation freely distorts those subspaces, and the distortion compounds with each edit. C_{MOIR} samples from the model's actual internalized distribution, covering exactly the subspaces the model relies on, regardless of whether they appear in any external corpus. The benefit grows with the number of edits and the distance between the editing distribution and the protected capability.

7. Discussion

Our results show that this geometric mismatch determines which capabilities survive an edit and which collapse. Replacing the Wikipedia-derived C_{Wiki} with C_{MOIR}, estimated from the model's own random-prefix generations, restores out-of-domain capability across all four editor-regime combinations and three model families we tested, without sacrificing editing quality and without requiring access to training data. The gap is most striking under sequential editing on specialized domains: C_{Wiki} collapses GSM8K to 0.2% on Qwen-3 within $5{\times}10^3$ AlphaEdit edits, while C_{MOIR} preserves 56.6%.

NLL and JSD comparisons (Figure 3, Appendix D.3) show that external proxies suffer from an inherent geometric bias relative to the operative distribution p_θ that persists regardless of sample size. Empirically, this led to a result we did not anticipate: C_{MOIR} matches and sometimes exceeds the performance of an "Oracle" covariance built from the official OLMo-Mix-1124 corpus (Table 1). We attribute this to the fact that post-training procedures like instruction tuning shift the model's operative manifold away from its raw pre-training state. In contrast, self-generation provides a dynamic map of this evolved geometry that static external corpora cannot replicate.

Limitations. Our study is bounded by three primary considerations. First, evaluation considered only two locate-then-edit families (MEMIT, AlphaEdit). Even though our distributional findings should generalize to any paradigm utilizing a covariance-like operator to define the preservation space (including memory-based or hypernetwork editors), this remains to be empirically verified. Second, the quality of MOIR depends on the model's generation behavior. In cases of extreme over-optimization as in extreme Reinforcement Learning from Human Feedback (RLHF), where the decoding distribution is pathologically mode-collapsed, self-generation might under-sample specialized or rare capa-

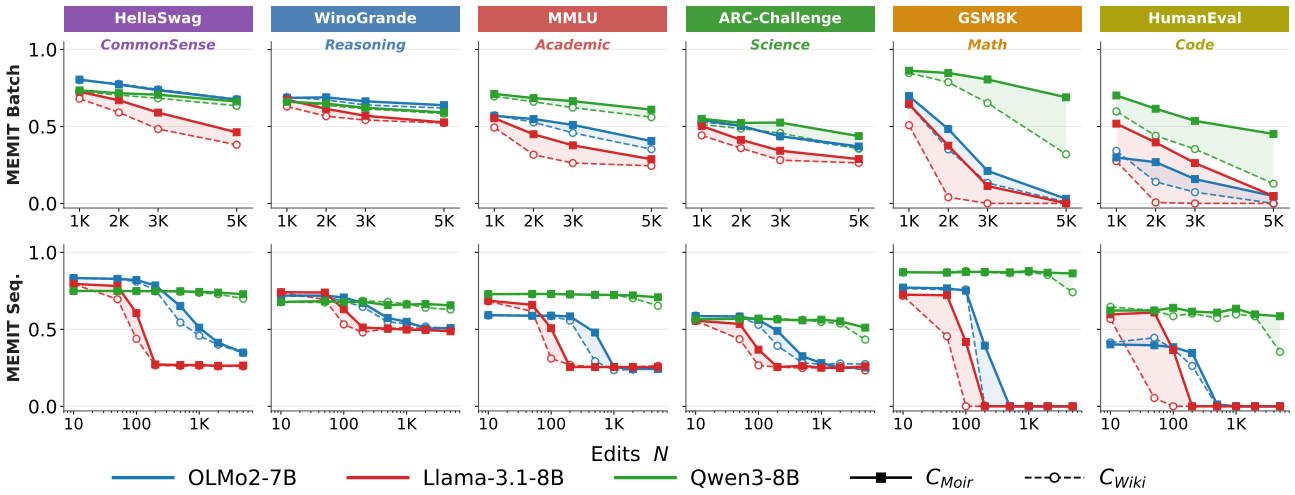


Figure 5. Per-task preservation under MEMIT. Six benchmark scores after batch and sequential editing on three instruction-tuned models. Shaded bands mark the gap. C_{MOIR} consistently delays collapse, most visibly on GSM8K, HumanEval, and across the sequential cliff regime.

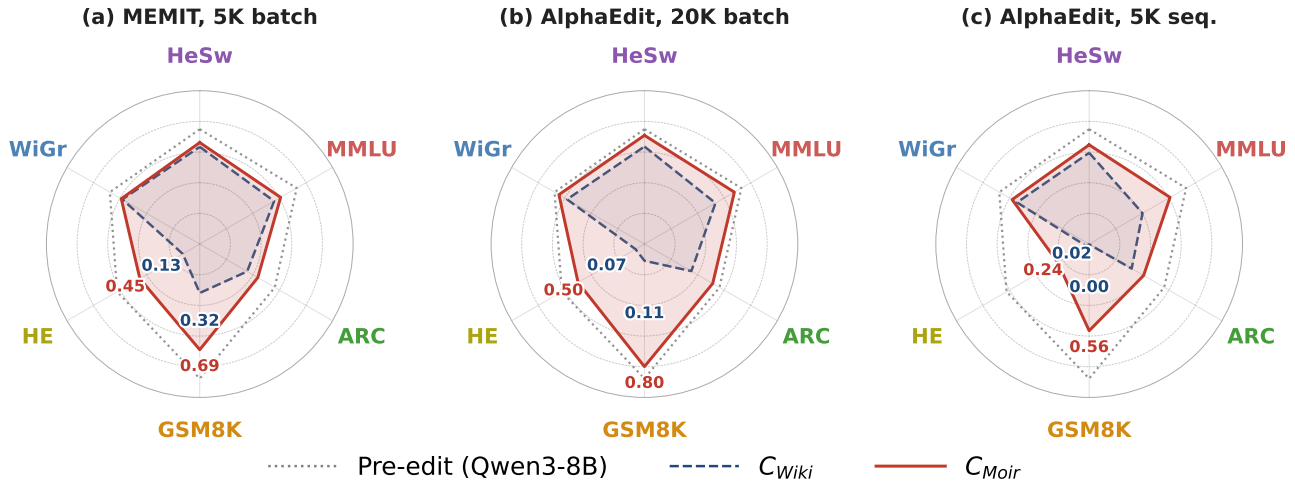


Figure 6. Representative radar chart on Qwen3-8B (analogous patterns for OLMo-2 and Llama-3.1 in Fig. 5). The collapse is asymmetric: C_{Wiki} preserves encyclopedic axes (MMLU, WiGr, HeSw) but GSM8K/HE collapse toward zero. C_{MOIR} closes this gap without disturbing the preserved axes.

bilities. Third, our experiments focus on instruction-tuned models in the 7–8B range; behavior at frontier scale (70B+) or on raw base (non-instruct) models warrants separate study. While MOIR consistently improves preservation, we do not claim self-generation is uniquely optimal. Stronger external mixtures may close part of the gap against Wikipedia, but the distributions most relevant to deployed models, such as post-training and usage data, are typically inaccessible. MOIR offers a practical, data-free approximation aligned with the model's post-training distribution.

Implications for editing as infrastructure. If knowledge editing is to serve as a robust operational layer for keeping deployed models current (enabling real-time factual patching, content removal, regulatory compliance, continuous updates), then preservation cannot be an afterthought tied to a 2020 Wikipedia dump. Our results indicate that this limitation is fixable without any change to existing editors: C_{MOIR} requires a one-time, ∼2-hour generation as a drop-in component for MEMIT, AlphaEdit, or any future covariance-based editor. The broader methodological lesson is that proxies inherited from earlier work warrant re-evaluation as the field matures and models evolve. Treating this component with rigor is essential for transitioning model editing from a research demonstration into a reliable and practical discipline for model maintenance.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning, specifically the reliability of knowledge editing for deployed language models. By reducing

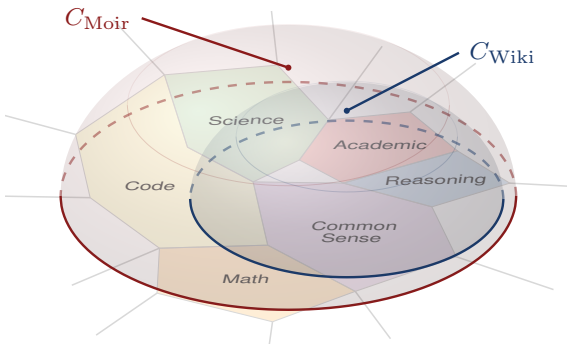


Figure 7. Knowledge Coverage: C_{MOIR} vs C_{Wiki}.

the collateral damage that editing inflicts on reasoning and coding capabilities, MOIR may make continuous factual patching, content removal, and regulatory compliance updates more practical, lowering the need for full retraining. As with any editing technique, the same machinery could in principle be used to insert incorrect or harmful facts; MOIR does not change this dual-use profile, as it modifies only how the preservation space is estimated. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An open language model for mathematics. In *International Conference on Learning Representations (ICLR)*, 2024.

Bai, J., Bai, S., Chu, Y., et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chhetri, V., Siddique, A. B., and Farooq, U. Understanding robustness of model editing in code LLMs: An empirical study. *arXiv preprint arXiv:2511.03182*, 2025.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Fang, J., Jiang, H., Wang, K., Ma, Y., Shi, J., Wang, X., He, X., and Chua, T.-S. AlphaEdit: Null-space constrained model editing for language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Geva, M., Schuster, R., Berant, J., and Levy, O. Transformer feed-forward layers are key-value memories. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Groeneveld, D., Beltagy, I., Walsh, P., et al. OLMo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

Gu, J.-C., Xu, H.-X., Ma, J.-Y., Lu, P., Ling, Z.-H., Chang, K.-W., and Peng, N. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*, 2024a.

Gu, J.-C., Xu, H.-X., Ma, J.-Y., Lu, P., Ling, Z.-H., Chang, K.-W., and Peng, N. Model editing harms general abilities of large language models: Regularization to the rescue. In *International Conference on Machine Learning (ICML)*, 2024b.

Hartvigsen, T., Sankaranarayanan, S., Palangi, H., Kim, Y., and Ghassemi, M. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S., Bansal, H., Guha, E., Keh, S., et al. DataComp-LM: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Li, Q., Liu, X., Tang, Z., Dong, P., Li, Z., Pan, X., and Chu, X. Should we really edit language models? on the evaluation of edited language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., et al. StarCoder: May the source be with you! *Transactions on Machine Learning Research (TMLR)*, 2023.

Lin, Z., Beigi, M., Li, H., Zhou, Y., Zhang, Y., Wang, Q., Yin, W., and Huang, L. Navigating the dual facets: A comprehensive evaluation of sequential memory editing in large language models. In *Association for Computational Linguistics (ACL)*, 2024.

Liu, X., Si, Q., Liu, Z., Yang, C., Gu, N., and Lin, Z. Beyond the covariance trap: Unlocking generalization in same-subject knowledge editing for large language models. *arXiv preprint arXiv:2603.15518*, 2026.

Ma, J.-Y., Wang, H., Xu, H.-X., Ling, Z.-H., and Gu, J.-C. Perturbation-restrained sequential model editing. *arXiv preprint arXiv:2405.16821*, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b.

Meng, K., Sharma, A. S., Andonian, A., Belinkov, Y., and Bau, D. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*, 2023.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022a.

Mitchell, E., Lin, C., Bosselut, A., Manning, C. D., and Finn, C. Memory-based model editing at scale. In *International Conference on Machine Learning (ICML)*, 2022b.

OLMo Team, Walsh, P., Soldaini, L., Groeneveld, D., et al. 2 OLMo 2 furious. In *Conference on Language Modeling (COLM)*, 2025. arXiv:2501.00656.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, 2021.

Shumailov, I., Shumaylov, Z., Zhao, Y., Papernot, N., Anderson, R., and Gal, Y. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024.

Soldaini, L., Kinney, R., Bhagia, A., et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*, 2024.

Together Computer. RedPajama-Data: An open source recipe to reproduce LLaMA training dataset, April 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Wang, P., Zhang, N., Tian, B., Xi, Z., Yao, Y., Xu, Z., Wang, M., Mao, S., Wang, X., Cheng, S., Liu, K., Ni, Y., Zheng, G., and Chen, H. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In *Association for Computational Linguistics (ACL)*, 2019.

Zhang, N., Yao, Y., Tian, B., Wang, P., Deng, S., Wang, M., Xi, Z., Mao, S., Zhang, J., Ni, Y., Cheng, S., Xu, Z., Xu, X., Gu, J.-C., Jiang, Y., Xie, P., Huang, F., Liang, L., Zhang, Z., Zhu, X., Zhou, J., and Chen, H. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

Appendix Overview

The supplementary material is organized as follows:

- **Appendix A (Extended Methodology):** Details the MOIR operational algorithm (A.1) and the construction of the OLMo-Mix-1124 Oracle (A.2).
- **Appendix B (Theoretical Definitions and Proofs):** Provides the full parameter update formulations, K-FAC setup, and proofs for Proposition 1.
- **Appendix C (Experimental Setup and Baselines):** Details models, editors, hyperparameters, evaluation metrics, causal tracing, and pre-edit baselines.
- **Appendix D (Additional Distributional Analyses):** Contains the self-generated token distributions and per-document NLL analyses.
- **Appendix E (Full Tabular Results):** Presents comprehensive tabular results for all editors, models, and regimes.

A. Extended Methodology

A.1. MOIR Operational Details

Notation. Let $\phi_\ell(\cdot)$ denote the function that maps an input token (in context) to its *key* $\mathbf{k} \in \mathbb{R}^d$ at layer ℓ, defined as the input to the second MLP linear projection W^ℓ_{out} in the targeted transformer block—the same activation extracted by ROME and MEMIT for C estimation. The sample estimator we compute matches the form used by prior work,

$$\hat{C}^\ell_{\text{MOIR}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{k}^\ell_i \, (\mathbf{k}^\ell_i)^\top, \qquad \mathbf{k}^\ell_i = \phi_\ell(x_i), \tag{3}$$

and differs only in how the keys $\{\mathbf{k}^\ell_i\}_{i=1}^N$ are obtained.

Key insight. An autoregressive language model p_θ defines an implicit distribution over token sequences shaped by its full training history. Sampling tokens from this distribution and extracting their keys yields an empirical estimator whose population limit approximates the key covariance under the true training distribution:

$$\mathbb{E}_{x\sim p_\theta(\,\cdot\,|s)}\left[\phi_\ell(x)\,\phi_\ell(x)^\top\right] \;\approx\; \mathbb{E}_{x\sim \mathcal{D}_{\text{train}}}\left[\phi_\ell(x)\,\phi_\ell(x)^\top\right], \tag{4}$$

where s is a short seed prefix from which generation proceeds. The approximation is exact when $p_\theta(\,\cdot \mid s) = \mathcal{D}_{\text{train}}$ marginally over s; in practice, the choice of s controls how broadly the model's internalized distribution is sampled (see Section 4 and Table 2).

Self-generation protocol. Given a target model p_θ, layer set $\mathcal{L}$, total token budget N, and per-sequence length S, MOIR estimates $\{\hat{C}^\ell_{\text{MOIR}}\}_{\ell\in\mathcal{L}}$ via Algorithm 1.

Algorithm 1 MOIR: Self-Generated Covariance Estimation

Require: Model p_θ, target layers $\mathcal{L}$, token budget N, max sequence length S, vocabulary $\mathcal{V}$
Ensure: Sample covariances $\{\hat{C}^\ell_{\text{MOIR}}\}_{\ell\in\mathcal{L}}$

$\hat{C}^\ell \leftarrow \mathbf{0}_{d\times d}$ for each $\ell \in \mathcal{L}$; $\quad n \leftarrow 0$
while $n < N$ **do**
  Sample seed token $z \sim \text{Uniform}(\mathcal{V})$ {`rand` × 1}
  Generate $x_{1:T} \sim p_\theta(\,\cdot \mid z)$ with $T \le S$ {Autoregressive sampling}
  for each $\ell \in \mathcal{L}$ **do**
    Extract keys $\mathbf{k}^\ell_t = \phi_\ell(x_t)$ for $t = 1, \ldots, T$ via forward pass
    $\hat{C}^\ell \leftarrow \hat{C}^\ell + \sum_{t=1}^{T} \mathbf{k}^\ell_t \, (\mathbf{k}^\ell_t)^\top$
  end for
  $n \leftarrow n + T$
end while
$\hat{C}^\ell_{\text{MOIR}} \leftarrow \hat{C}^\ell / n$ for each $\ell \in \mathcal{L}$
return $\{\hat{C}^\ell_{\text{MOIR}}\}_{\ell\in\mathcal{L}}$

Design choices. We initialize generation from a single uniformly random vocabulary token rather than from a structured prompt or the BOS token alone. As shown in Section 4, this minimal perturbation breaks the mode collapse induced by post-training (where BOS-conditioned generation concentrates on instruction-following openings) without steering generation away from the model's natural activation statistics. The resulting key set $\{\mathbf{k}^\ell_t\}$ is collected directly from the forward pass during generation, so MOIR adds no inference cost beyond the generation itself.

A.2. OLMo-Mix-1124 Oracle Construction

To test whether distributional alignment between C and the model's pretraining mixture can recover the out-of-domain capabilities lost under conventional Wikipedia-based estimation, we construct an oracle covariance that mirrors the official OLMo-Mix-1124 corpus used to pretrain OLMo-2 (OLMo Team et al., 2025). OLMo-Mix-1124 consists of five components spanning web text, code, scientific writing, mathematical content, and encyclopedic prose, in proportions chosen by the OLMo team to balance generalist coverage with domain-specific capability.

Component-wise covariance estimation. Rather than sampling from the joint mixture directly, we estimate a separate uncentered covariance matrix $C^{(i)}$ for each of the five components by drawing 100K tokens from each and computing

$$C^{(i)} = \frac{1}{n_i} \sum_{k=1}^{n_i} \mathbf{k}_k^{(i)} \mathbf{k}_k^{(i)\top}, \tag{5}$$

where $\mathbf{k}_k^{(i)}$ denotes the k-th key vector at the targeted MLP layer for tokens drawn from component i, and $n_i = 100\text{K}$ for all i. The five components and their corresponding sources are:

- **Web (DCLM-baseline)** (Li et al., 2024a): filtered Common Crawl text used as the dominant pretraining source.
- **Code (StarCoder)** (Li et al., 2023): source code repositories spanning multiple programming languages.
- **Science (Dolma 1.7 papers)** (Soldaini et al., 2024): the academic papers subset of Dolma 1.7, primarily peS2o.
- **Math (Proof Pile II)** (Azerbayev et al., 2024): mathematical text including formal proofs, ArXiv math papers, and StackExchange.
- **Wikipedia**: English Wikipedia, the same source used by the conventional baseline but resampled here for consistency.

Mixture composition. The oracle covariance is composed as a weighted sum of the component-wise estimates:

$$C^{\text{oracle}} = \sum_{i=1}^{5} w_i \, C^{(i)}, \tag{6}$$

with weights w_i matching the official OLMo-Mix-1124 mixture proportions: 95.4% web, 2.1% code, 2.1% science, 0.6% math, and 0.1% Wikipedia (OLMo Team et al., 2025).

Justification. This decomposed construction yields the same expectation as direct sampling from the joint mixture, since the uncentered covariance operator is linear in the underlying distribution: if $p_{\text{mix}} = \sum_i w_i p_i$, then $\mathbb{E}_{\mathbf{k}\sim p_{\text{mix}}}[\mathbf{k}\mathbf{k}^\top] = \sum_i w_i \mathbb{E}_{\mathbf{k}\sim p_i}[\mathbf{k}\mathbf{k}^\top]$. The decomposition has two practical advantages over direct sampling. First, it ensures that each $C^{(i)}$ is statistically well-conditioned, since 100K tokens per component is sufficient for stable estimation even for low-weight components such as Math (0.6%) and Wikipedia (0.1%), which would receive only 600 and 100 samples respectively under proportional sampling at the same total budget. Second, it allows efficient ablation across mixture ratios: alternative weightings w_i can be evaluated without re-sampling, by simply recomposing the precomputed component matrices.

Dolma 1.7 (Wiki subset only) baseline. For comparison, we also report a baseline that estimates C exclusively from the Wikipedia subset of Dolma 1.7. This represents a minimal departure from the conventional Wikipedia proxy: the data source is encyclopedic prose, but drawn from the curated Dolma release rather than the HuggingFace `wikipedia` dump used by ROME and MEMIT. The persistence of cross-domain collapse under this baseline (Table 1) confirms that the distributional limitation, not the specific Wikipedia source, is responsible for the observed degradation.

B. Theoretical Definitions and Proofs

This appendix collects the mathematical material referenced from Section 5. Appendix B.1 states the parameter updates of MEMIT and AlphaEdit and the K-FAC setup. Appendix B.2 proves proposition 1 and clarifies its scope across closed-form covariance-constrained editors. Appendix B.3 records the standard JSD-based upper bound on covariance bias used as scaffolding in Section 5.2, and discusses why we do not elevate it to a proposition.

B.1. Parameter Updates and K-FAC Setup

Full Parameter Update Formulations. For completeness, we detail the exact parameter updates ΔW for the covariance-constrained editors analyzed in this work. Given a residual matrix R encoding target values and keys K_{new} for the new facts, the update must preserve previously edited facts (K_{past}) and foundational knowledge (K_0).

MEMIT (Soft Constraint): Incorporates $C = K_0 K_0^\top$ as a soft regularization penalty:

$$\Delta W_{\text{MEMIT}} = R K_{\text{new}}^\top \big(K_{\text{past}} K_{\text{past}}^\top + K_{\text{new}} K_{\text{new}}^\top + \lambda C\big)^{-1}, \tag{7}$$

AlphaEdit (Hard Constraint): Projects the update strictly into the null space of C. Applying SVD to C and discarding eigenvectors with non-zero eigenvalues yields a submatrix $\hat{U}$ spanning the null space. The orthogonal projection matrix is $P = \hat{U}\hat{U}^\top$, guaranteeing $PK_0 = \mathbf{0}$. The update is:

$$\Delta W_{\text{AlphaEdit}} = R K_{\text{new}}^\top P \left(K_{\text{past}} K_{\text{past}}^\top P + K_{\text{new}} K_{\text{new}}^\top P + I\right)^{-1}. \tag{8}$$

Neither (7) nor (8) contains an output-side factor G_ℓ: both reduce to an input-only form via the G-cancellation of proposition 1; the original derivations of MEMIT and AlphaEdit involve a K-FAC-style weighted objective in which G appears, but it is eliminated in the closed-form solution.

Activations and the K-FAC framework. To understand the geometric role of C in these updates, we view them through the lens of the K-FAC framework. For an MLP layer indexed by ℓ in a transformer, let $\phi_\ell : \mathcal{X} \to \mathbb{R}^{d_\ell}$ denote the layer-input activation map—specifically, the input to the second linear projection W_{out}^ℓ in the targeted block, matching the activation extracted by ROME and MEMIT for C estimation. Throughout this appendix we assume the boundedness condition

$$M_\ell := \sup_{x \in \mathcal{X}} \|\phi_\ell(x)\|_2 < \infty, \tag{9}$$

which holds in practice via layer normalization. Under K-FAC, the layer Fisher information matrix factorizes as

$$F_\ell \approx A_\ell \otimes G_\ell, \quad A_\ell(p) := \mathbb{E}_{x\sim p}\big[\phi_\ell(x)\phi_\ell(x)^\top\big], \quad G_\ell(p) := \mathbb{E}_{x\sim p}\big[g_\ell(x) g_\ell(x)^\top\big], \tag{10}$$

where $g_\ell(x)$ is the output-side gradient. We interchangeably write $\Sigma_\ell(p) = A_\ell(p)$ to emphasize that this is also the population target of the empirical preservation matrix C in MEMIT and AlphaEdit.

B.2. Proof of proposition 1 and its Scope

We restate proposition 1 for convenience, give two complementary proofs, and then discuss the scope across the three editors analyzed in this work.

Proposition 2 (Restatement of proposition 1)**.** *Let $A \in \mathbb{R}^{d_{\text{in}}\times d_{\text{in}}}$ be positive definite and $G \in \mathbb{R}^{d_{\text{out}}\times d_{\text{out}}}$ be positive semi-definite. For $K_{\text{new}} \in \mathbb{R}^{d_{\text{in}}\times m}$ satisfying $K_{\text{new}}^\top A^{-1} K_{\text{new}} \succ 0$ and $R \in \mathbb{R}^{d_{\text{out}}\times m}$, the constrained minimum-norm problem*

$$\min_{\Delta W \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}} \operatorname{vec}(\Delta W)^\top (A \otimes G) \operatorname{vec}(\Delta W) \quad \textit{s.t.} \quad \Delta W\, K_{\text{new}} = R \tag{11}$$

admits

$$\Delta W^\star \;=\; R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1}\, K_{\text{new}}^\top A^{-1}$$

as a minimizer, and this expression is independent of G. The minimizer is unique whenever $G \succ 0$.

We give two proofs: a perturbation argument that handles the PSD case directly, and a Lagrangian argument that recovers the original ROME-style derivation in the PD case and extends to PSD G by continuity.

First proof (perturbation). Feasibility. Direct substitution gives

$$\Delta W^\star K_{\text{new}} = R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1}(K_{\text{new}}^\top A^{-1} K_{\text{new}}) = R.$$

Optimality. Any feasible ΔW admits the decomposition $\Delta W = \Delta W^\star + \delta W$ with $\delta W\, K_{\text{new}} = 0$. The objective expands as

$$\begin{aligned}\operatorname{Tr}(G\,\Delta W\,A\,\Delta W^\top) = \operatorname{Tr}(G\,\Delta W^\star A\,\Delta W^{\star\top}) \\ + 2\operatorname{Tr}(G\,\Delta W^\star A\,\delta W^\top) + \operatorname{Tr}(G\,\delta W\,A\,\delta W^\top).\end{aligned}$$

The identity $\Delta W^\star A = R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1} K_{\text{new}}^\top$ collapses the cross term to a Frobenius inner product against $\delta W\, K_{\text{new}}$:

$$\operatorname{Tr}(G\,\Delta W^\star A\,\delta W^\top) = \left\langle\, GR\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1},\ \delta W\, K_{\text{new}} \,\right\rangle_F = 0$$

for *any* G, since $\delta W\, K_{\text{new}} = 0$. The residual satisfies

$$\operatorname{Tr}(G\,\delta W\,A\,\delta W^\top) = \|G^{1/2}\,\delta W\,A^{1/2}\|_F^2 \geq 0,$$

with equality forcing $\delta W = 0$ when $G \succ 0$. The single-vector case ($k^* \in \mathbb{R}^{d_{\text{in}}}$, $b \in \mathbb{R}^{d_{\text{out}}}$) follows identically, yielding $\Delta W^\star = b\,(A^{-1}k^*)^\top / (k^{*\top} A^{-1} k^*)$. □

Second proof (Lagrangian; PD case directly, PSD by continuity). Assume first $G \succ 0$. The Lagrangian

$$\mathcal{L} = \operatorname{Tr}(G\,\Delta W\,A\,\Delta W^\top) - \operatorname{Tr}\big(\Lambda^\top(\Delta W\,K_{\text{new}} - R)\big)$$

has stationarity condition

$$2G\,\Delta W\,A = \Lambda\,K_{\text{new}}^\top \implies \Delta W = \tfrac{1}{2}\,G^{-1}\,\Lambda\,K_{\text{new}}^\top\,A^{-1}.$$

Substituting into the constraint $\Delta W\,K_{\text{new}} = R$ yields $\frac{1}{2}\,G^{-1}\,\Lambda\,(K_{\text{new}}^\top A^{-1} K_{\text{new}}) = R$, so $\Lambda = 2\,G\,R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1}$. Back-substitution cancels G:

$$\Delta W^\star = G^{-1}\cdot G\,R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1}\,K_{\text{new}}^\top A^{-1} = R\,(K_{\text{new}}^\top A^{-1} K_{\text{new}})^{-1}\,K_{\text{new}}^\top A^{-1}.$$

Strict convexity gives uniqueness.

For PSD G, set $G_\epsilon := G + \epsilon I \succ 0$ for $\epsilon > 0$. Applying the derivation above to G_ϵ produces a minimizer that is *independent of* ϵ (since G_ϵ cancels identically): $\Delta W_\epsilon^\star = \Delta W^\star$. For any feasible ΔW,

$$\operatorname{Tr}(G_\epsilon\,\Delta W\,A\,\Delta W^\top) \ \geq\ \operatorname{Tr}(G_\epsilon\,\Delta W^\star\,A\,\Delta W^{\star\top}).$$

Letting $\epsilon \to 0^+$ and using continuity of the objective in G,

$$\operatorname{Tr}(G\,\Delta W\,A\,\Delta W^\top) \ \geq\ \operatorname{Tr}(G\,\Delta W^\star\,A\,\Delta W^{\star\top}),$$

so $\Delta W^\star$ minimizes the PSD problem. Uniqueness can fail when $G \not\succ 0$: if $\delta W\,K_{\text{new}} = 0$ and $G\,\delta W = 0$, then $\Delta W^\star + \delta W$ is also a minimizer. □

B.3. Covariance Bias and the JSD Upper Bound

Section 5.2 cites a standard upper bound on the covariance bias $\|\Sigma_\ell(q) - \Sigma_\ell(p_\theta)\|_F$ in terms of the Jensen–Shannon divergence. We record that bound here for completeness.

Lemma 3 (TV-Lipschitz continuity of Σ_ℓ)**.** *For probability distributions p, q on $\mathcal{X}$ and bound M_ℓ from (9),*

$$\big\|\Sigma_\ell(p) - \Sigma_\ell(q)\big\|_F \ \leq\ 2M_\ell^2 \cdot \|p - q\|_{\text{TV}}. \tag{12}$$

Proof. With respect to a dominating reference measure μ on $\mathcal{X}$, $\Sigma_\ell(p) - \Sigma_\ell(q) = \int \phi_\ell(x)\phi_\ell(x)^\top (p(x) - q(x))\,d\mu$. Applying the triangle inequality for the Frobenius norm under integration,

$$\big\|\Sigma_\ell(p) - \Sigma_\ell(q)\big\|_F \leq \int \|\phi_\ell(x)\|_2^2\,|p(x) - q(x)|\,d\mu \leq M_\ell^2 \int |p(x) - q(x)|\,d\mu = 2M_\ell^2 \cdot \|p - q\|_{\text{TV}}.$$

□

Combining Lemma 3 with the Pinsker-type inequality $\|p - q\|_{\mathrm{TV}} \leq \sqrt{2\,\mathrm{JSD}(p, q)}$ yields the JSD bound referenced in Section 5.2:

$$\left\|\Sigma_\ell(q) - \Sigma_\ell(p_\theta)\right\|_F \;\leq\; 2\sqrt{2}\, M_\ell^2 \cdot \sqrt{\mathrm{JSD}(q, p_\theta)}. \tag{13}$$

This relates covariance bias to a JSD measurement, motivating our use of JSD as the alignment metric in Section 5.2.

B.4. Summary of Assumptions

For ease of reference, we collect the assumptions invoked in this appendix.

- **(Boundedness, used throughout.)** $M_\ell := \sup_x \|\phi_\ell(x)\|_2 < \infty$ (9). Holds via layer normalization in modern transformers; can be relaxed to a fourth-moment bound $\mathbb{E}_p[\|\phi_\ell\|^4] < \infty$ for the JSD bound (13).
- **(K-FAC factorization.)** $F_\ell \approx A_\ell \otimes G_\ell$, where A_ℓ is positive definite and G_ℓ is positive semi-definite (10).

C. Experimental Setup and Baselines

C.1. Models, Editors, and Hyperparameters

Models and editing layers. We evaluate three open-weight instruction-tuned models with different training corpora: OLMo-2-7B-Instruct (Groeneveld et al., 2024), whose pretraining data (OLMo-Mix-1124) is publicly released (OLMo Team et al., 2025); Llama-3.1-8B-Instruct (Grattafiori et al., 2024); and Qwen3-8B (Bai et al., 2023), both trained on proprietary corpora. For each model, the target MLP layers for editing are identified via causal tracing (see Appendix C.5): layers 9–14 for OLMo-2, 3–8 for Llama-3.1, and 12–17 for Qwen3.

Editing methods. We use two representative locate-then-edit methods spanning both families of C usage: MEMIT (Meng et al., 2023), which incorporates C as a soft regularization penalty, and AlphaEdit (Fang et al., 2025), which uses C to define a hard null-space projection P_0. Both are evaluated on the COUNTERFACT association dataset (Meng et al., 2022a) under two regimes: *sequential editing*, in which edits are applied one at a time and the modified weights persist into the next edit (10 to 5K cumulative edits), and *batch editing*, in which a fixed number of edits are applied jointly (1K–5K for MEMIT, 5K–20K for AlphaEdit, reflecting each method's typical operating range).

Editing Dataset and Evaluation Framework. All knowledge editing experiments are conducted on the COUNTERFACT dataset (Meng et al., 2022a), a standard benchmark specifically designed to evaluate the efficacy and specificity of factual edits in large language models. To ensure rigorous reproducibility and standardized metric computation, our evaluation pipeline is built upon the `EasyEdit` (Wang et al., 2023) and `KnowEdit` (Zhang et al., 2024) frameworks. Specifically, for the application of covariance-constrained updates and the computation of all edit metrics (Efficacy, Generalization, Specificity, and Portability), we follow the standardized protocols and prompt templates established by these libraries.

C variants. We compare three preservation targets:

- $C_{\textbf{Wiki}}$: the conventional baseline, estimated from 100K samples of the HuggingFace `wikipedia` corpus used by ROME and MEMIT.
- $C_{\textbf{MOIR}}$: self-generated C following Algorithm 1 with single random-token seeding ($\langle \texttt{rand} \rangle \times 1$) as our main method with $N = 100$K, matching the conventional sample size, unless otherwise noted (e.g., $\langle \texttt{bos} \rangle$).
- $C_{\textbf{OLMoMix}}$ (OLMo-2 only): C computed from the actual OLMo-Mix-1124 pretraining mixture (which we refer to as the oracle) as a weighted sum of component-wise covariances (construction detailed in Appendix A.2).

C.2. Implementation and Reproducibility Details

To ensure full reproducibility of our results, we detail the exact computational procedures, hyperparameters, and hardware configurations used in our experiments.

MOIR Generation and C Computation. MOIR generates the preservation data by sampling continuations from the target model itself. For each model, we generate 100,000 text samples using untruncated ancestral sampling (`temperature=1.0`, `top_p=1.0`). Each sample is seeded with a single randomly sampled vocabulary token (`<rand>`×1) and extended to a maximum of 256 new tokens. No filtering, deduplication, or post-processing is applied. This yields approximately 17M tokens for OLMo-2, 24M for Llama-3.1, and 25M for Qwen3.

From this cached text, we compute the uncentered covariance matrix $C = \mathbb{E}[\mathbf{k}\mathbf{k}^\top]$ at each target MLP layer using batched forward passes (12,288 tokens per batch). Activations are collected at the input to `mlp.down_proj`, accumulated in `float32`, and stored. The resulting C matrices have dimensions $11{,}008 \times 11{,}008$ for OLMo-2, $14{,}336 \times 14{,}336$ for Llama-3.1, and $12{,}288 \times 12{,}288$ for Qwen3. For the conventional baselines, we use 100,000 articles from the 2020-05-01 English Wikipedia dump ($\sim$54.8M tokens).

Extended Editing Hyperparameters. All models are loaded in `float16`, but the actual parameter editing is performed in `float64` to ensure strict numerical stability during matrix inversions and SVD.

- **MEMIT:** We use a regularization weight $\lambda = 15{,}000$, λ is selected from the grid $\{0.5K, 1.5K, 5K, 15K, 30K\}$, 25 gradient steps for value optimization (learning rate $= 0.5$, weight decay $=$ 1e-3), and a KL factor of 0.0625.
- **AlphaEdit:** The null-space projection threshold τ is selected from the grid $\{0.02, 0.005, 0.001, 0.0005\}$ based on per-model optimal performance on a held-out validation split.

Compute Resources. All experiments, including self-generation, covariance estimation, and sequential/batch editing, are conducted on NVIDIA A100 (80GB) GPUs. Each job is allocated a single GPU, 4 CPU cores, and 60GB of system memory, managed via SLURM on an HPC cluster. Estimating C_{MOIR} requires a one-time forward pass over the 100K self-generated tokens, which takes approximately 15 minutes per model on a single A100, and the resulting matrix is reused across all subsequent edits.

C.3. Evaluation Metrics

ROME/MEMIT convention (Meng et al., 2022a; 2023). These metrics compare the *sequence-level* likelihood of the new target against the original.

- **Efficacy Score (ES).** The fraction of edits for which the model assigns higher average token probability to the new target o^* than to the original o:

$$\text{ES} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\left[\underbrace{-\frac{1}{T}\sum_{t=1}^{T}\log P_\theta(o_t^* \mid x_i, o_{<t}^*)}_{\text{NLL}(o^*|x_i)} \;<\; \underbrace{-\frac{1}{S}\sum_{s=1}^{S}\log P_\theta(o_s \mid x_i, o_{<s})}_{\text{NLL}(o|x_i)}\right] \tag{14}$$

  where T and S are the token lengths of o^* and o, respectively. Each NLL is length-normalized.

- **Paraphrase Score (PS).** Same as ES but evaluated on paraphrased prompts x':

$$\text{PS} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}[\,\text{NLL}(o^* \mid x_i') < \text{NLL}(o \mid x_i')\,] \tag{15}$$

- **Neighborhood Score (NS).** For neighborhood prompts x_n (semantically related facts that should *not* change), we check whether *every* target token is correctly predicted via argmax:

$$\text{NS} = \frac{1}{|\mathcal{N}|}\sum_{x_n\in\mathcal{N}}\prod_{s=1}^{S}\mathbf{1}[\,\arg\max P_\theta(\cdot \mid x_n, o_{<s}) = o_s\,] \tag{16}$$

EasyEdit/KnowEdit convention (Wang et al., 2023; Zhang et al., 2024). These metrics use *token-level argmax matching* under teacher forcing.

- **Efficacy (token-level accuracy).** The average fraction of target tokens correctly predicted by argmax:

$$\text{Efficacy} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}[\,\arg\max P_\theta(\cdot \mid x_i, o_{<t}^*) = o_t^*\,] \tag{17}$$

  Note: this averages over tokens (partial credit), whereas ROME's `targets_correct` requires *all* tokens to match.

- **Generalization (token-level accuracy on paraphrases).**

$$\text{Generalization} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T}\mathbf{1}[\,\arg\max P_\theta(\cdot \mid x_i', o_{<t}^*) = o_t^*\,] \tag{18}$$

- **Specificity.** Identical to NS above (argmax match on neighborhood prompts). The two conventions agree on this metric.

- **Fluency***: baseline-normalized fluency, defined as $\widetilde{f} = f_{\text{edited}}/f_{\text{baseline}}$, where f is the mean token log-probability on generated continuations. Normalization ensures the score is comparable across models.

- **Portability**: fraction of generation-based portability probes answered correctly (`portability.generation_acc` in the EasyEdit framework).

We report two composite scores that summarize editing quality and capability preservation, respectively. Both are defined as the harmonic mean (HM) of their constituent metrics, which penalizes low outliers more than the arithmetic mean and better reflects practical failure modes.

Edit Score (S_e).

$$S_e = \text{HM}(\text{Efficacy},\ \text{Generalization},\ \text{Specificity},\ \text{Fluency}^*,\ \text{Portability}) \tag{19}$$

Preservation Score (S_p).

$$S_p = \text{HM}(\text{MMLU},\ \text{GSM8K},\ \text{HellaSwag},\ \text{WinoGrande},\ \text{ARC-C},\ \text{HumanEval}) \tag{20}$$

All six metrics are raw accuracy values (0–1), evaluated as described in Table 3.

Table 3. Preservation benchmark descriptions and evaluation metrics. All benchmarks are evaluated on the full test split unless noted otherwise.

Benchmark	**Domain**	**Metric**	**Tool**
MMLU (Hendrycks et al., 2021)	General knowledge	accuracy (`acc`)	lm-eval-harness
GSM8K (Cobbe et al., 2021)	Math reasoning	strict-match exact accuracy	lm-eval-harness
HellaSwag (Zellers et al., 2019)	Commonsense / NLU	length-normalized accuracy (`acc_norm`)	lm-eval-harness
WinoGrande (Sakaguchi et al., 2021)	Commonsense reasoning	accuracy (`acc`)	lm-eval-harness
ARC-Challenge (Clark et al., 2018)	Science reasoning	length-normalized accuracy (`acc_norm`)	lm-eval-harness
HumanEval (Chen et al., 2021)	Code generation	pass@1 (execution-based)	custom sandbox

HellaSwag and ARC-Challenge use length-normalized accuracy (`acc_norm`) to correct for varying completion lengths among multiple-choice options, following the standard protocol in `lm-evaluation-harness` (Gao et al., 2024). GSM8K uses strict string matching of the final numerical answer rather than flexible extraction, which provides a more conservative estimate. HumanEval evaluates functional correctness by executing generated code against unit tests. All 164 problems are evaluated with a single completion per problem (`n_samples=1`) using nucleus sampling (`temperature=0.2`, `top_p=0.95`). The same generation settings are used for both baseline and post-edit evaluation.

Why harmonic mean? If any single capability collapses to near zero (e.g., GSM8K $\rightarrow$ 0.2% under C_{Wiki}), the harmonic mean drops sharply, reflecting the practical reality that a model missing one core capability is not "mostly preserved." The arithmetic mean would mask such failures by averaging over the surviving metrics.

C.4. Pre-edit Baseline Performance

Table 4. Pre-edit baseline performance. MMLU, HellaSwag, WinoGrande, ARC-Challenge report accuracy; GSM8K reports exact-match; HumanEval reports pass@1.

Model	MMLU	GSM8K	HellaSwag	WinoGrande	ARC-Challenge	HumanEval
OLMo2-7B Base	60.5	68.5	80.5	74.6	57.2	13.4
OLMo2-7B-Instruct	59.2	77.3	83.3	71.4	58.8	39.0
Llama-3.1-8B Base	64.1	50.4	79.3	74.4	55.0	37.8
Llama-3.1-8B-Instruct	68.4	77.8	79.5	73.6	55.9	61.0
Qwen3-8B Base	74.8	84.5	78.6	72.8	56.8	63.4
Qwen3-8B	73.0	88.2	74.9	67.7	56.5	62.2

Table 4 reports the pre-edit performance of all six model configurations across six benchmarks. These scores serve as the reference point for measuring locality degradation after knowledge editing.

C.5. Causal Tracing for Target Layer Selection

We perform causal tracing (Meng et al., 2022a) on all six model configurations (three architectures $\times$ base/instruct) to identify which MLP layers are most critical for factual recall, thereby justifying the per-model choice of editing target layers.

Protocol. Following the three-step procedure of Meng et al. (2022a): (1) a *clean run* records the model's probability P_{clean} for the correct answer; (2) a *corrupted run* adds Gaussian noise ($\sigma = 3\times$ embedding std, following the original ROME protocol) to the subject token embeddings, yielding $P_{\text{corrupted}}$; (3) a series of *restored runs* each replace the hidden state at a single (layer, token position) with its clean-run value, yielding P_{restored}. The indirect effect is IE $= (P_{\text{restored}} - P_{\text{corrupted}})/(P_{\text{clean}} - P_{\text{corrupted}})$. A sample is valid when $P_{\text{clean}} - P_{\text{corrupted}} > 0.01$. We evaluate 100 CounterFact prompts per model and align results into a fixed window of[5 before | 3 subject slots | 10 after] before averaging. Using 3σ noise (scaled to each model's embedding statistics) enables fair comparison across architectures.

Results. Table 5 reports the top-10 layers by indirect effect at the last subject token for all six models. For each model, we select the contiguous range of 7 layers that maximizes the sum of IE as the MEMIT editing target (Table 6). Several patterns emerge:

- **Qwen3-8B** exhibits the sharpest factual concentration, with IE values exceeding 0.8 across layers 8–17 and a peak of 1.11 at layer 14 in the instruct variant. This suggests highly localized factual storage, favorable for precise editing.
- **OLMo2-7B** distributes recall broadly across layers 8–14 (peak IE ≈ 0.34 at layer 11), with base and instruct variants showing similar profiles.
- **Llama-3.1-8B** localizes recall in early layers 2–8 (peak at layer 4), consistent with prior ROME/MEMIT results on Llama-family models.
- Base and instruct variants of the same architecture share similar peak locations, though instruct models sometimes show sharper peaks (e.g., Llama-3.1 instruct at layers 3–5).

Table 5. Top-10 layers by indirect effect (IE) at the last subject token for all six model configurations (100 CounterFact prompts, noise $= 3\sigma$). **Bold** layers indicate the selected contiguous edit target range.

Rank	OLMo2 Base		OLMo2 Inst		Llama3 Base		Llama3 Inst		Qwen3 Base		Qwen3	
	L	IE	L	IE	L	IE	L	IE	L	IE	L	IE
1	**11**	**.340**	**12**	**.308**	**4**	**.283**	**4**	**.338**	**8**	**.831**	**14**	**1.11**
2	**12**	**.322**	**11**	**.298**	**3**	**.282**	**5**	**.334**	**9**	**.717**	**15**	**1.02**
3	**10**	**.316**	**13**	**.274**	**6**	**.266**	**3**	**.318**	**14**	**.695**	**16**	**.849**
4	**13**	**.307**	**10**	**.257**	**5**	**.249**	**6**	**.223**	**15**	**.687**	**13**	**.848**
5	**8**	**.307**	**9**	**.251**	**7**	**.233**	**7**	**.157**	7	.682	**12**	**.801**
6	**14**	**.301**	**14**	**.219**	**2**	**.225**	**8**	**.123**	**11**	**.642**	8	.808
7	**9**	**.294**	**8**	**.200**	**8**	**.220**	0	.132	**13**	**.641**	**17**	**.787**
8	7	.275	3	.185	10	.216	1	.113	**10**	**.625**	9	.745
9	16	.245	5	.179	11	.215	2	.102	16	.621	**11**	**.703**
10	15	.237	15	.177	0	.199	9	.094	4	.621	10	.697

Table 6. Summary of causal tracing results. Edit targets are the contiguous 6-layer range maximizing the sum of IE. Bold rows indicate the instruct models used in our experiments.

Model	Layers	Valid	Peak Layer	Peak IE	Edit Target (6 layers)
GPT-2 XL (prior work)	48	–	$\sim$20	–	17–22
GPT-J-6B (prior work)	28	–	$\sim$6	–	3–8
OLMo2-7B Base	32	52/100	11	0.340	8–13
OLMo2-7B-Instruct	**32**	**51/100**	**12**	**0.308**	**9–14**
Llama-3.1-8B Base	32	62/100	4	0.283	2–7
Llama-3.1-8B-Instruct	**32**	**58/100**	**4**	**0.338**	**3–8**
Qwen3-8B Base	36	48/100	8	0.831	6–11
Qwen3-8B	**36**	**41/100**	**14**	**1.115**	**12–17**

Figure 8 visualizes the layer-wise IE profiles for all six models, and Figure 9 compares base vs. instruct variants.

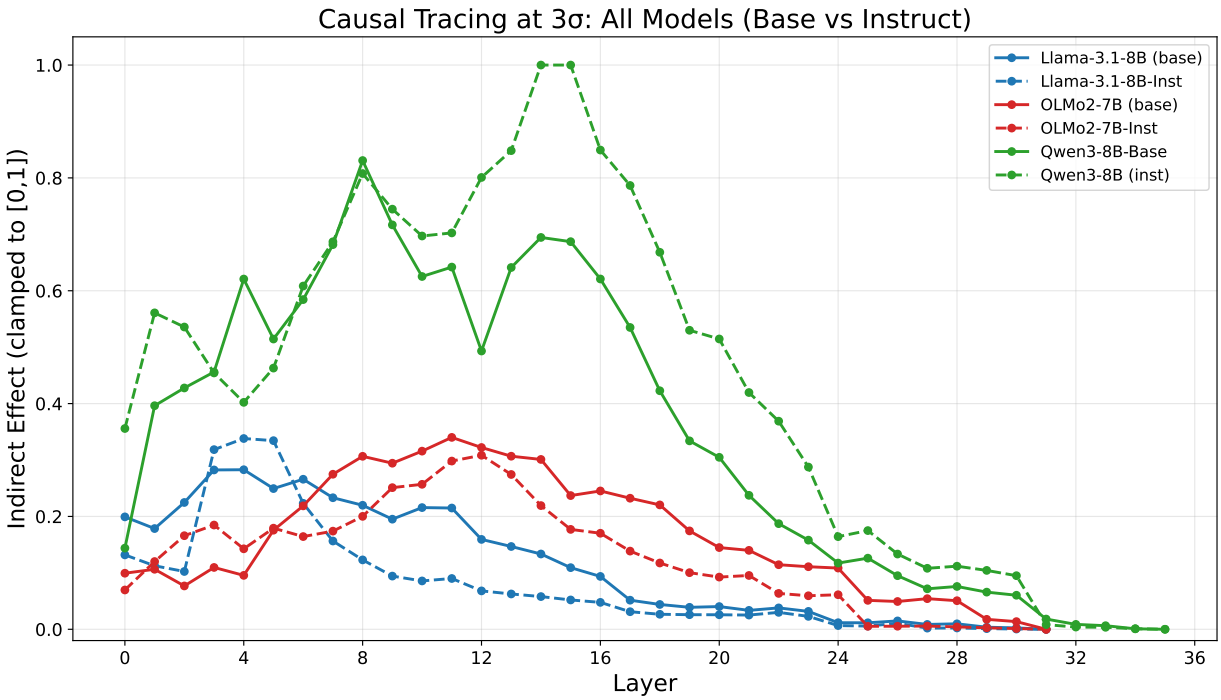


Figure 8. Layer-wise indirect effect at the last subject token for all six model configurations (100 CounterFact prompts, noise $= 3\sigma$). Shaded regions indicate the selected edit target layers. Qwen3 shows the strongest factual concentration; Llama-3.1 localizes recall earliest; OLMo2 distributes recall most broadly.

Justification for Orthogonal Constraints. While causal tracing isolates layers critical for factual recall, these specific MLP weights do not store facts in isolation. Factual recall circuits and complex out-of-domain subspaces (e.g., code, math) are highly entangled within the same weight matrices. Consequently, modifying these layers to inject a fact inherently risks destroying the overlapping capability manifolds, which necessitates the strict orthogonal projection (via C) employed by MEMIT and AlphaEdit.

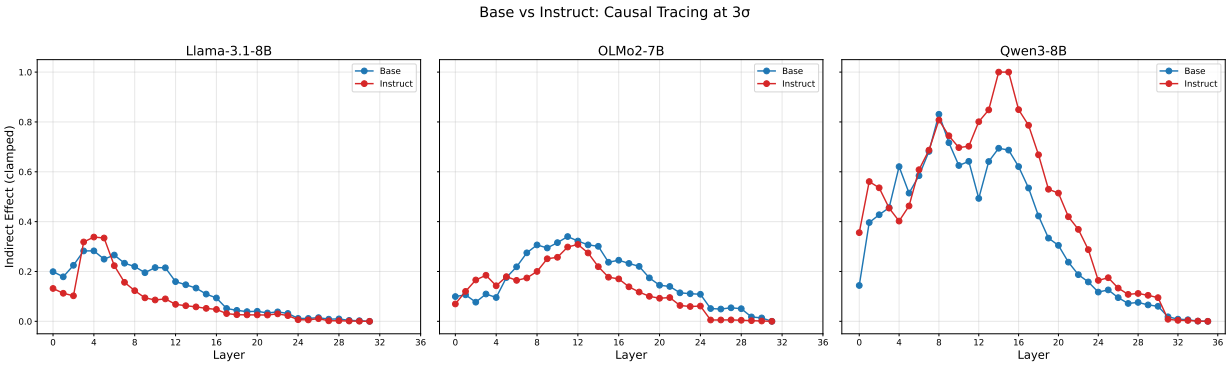


Figure 9. Base vs. instruct comparison for each architecture. Instruction tuning preserves the general shape of the factual recall profile but can shift or sharpen the peak (e.g., Qwen3 shifts from layer 8 to layer 14 after instruction tuning).

D. Additional Distributional Analyses

This section provides extended empirical evidence supporting the distributional claims made in Section 4 and Section 5. We visualize the token-level diversity induced by random-prefix seeding and compare the per-document negative log-likelihood (NLL) across various corpora.

D.1. Self-generated Token Distribution

As discussed in Section 3, ⟨bos⟩-only seeding suffers from severe mode collapse, whereas random-prefix seeding successfully diversifies the generated trajectories. Figure 1 illustrates this pipeline, and Tables 7 and 8 provide qualitative examples of the generated continuations under both regimes.

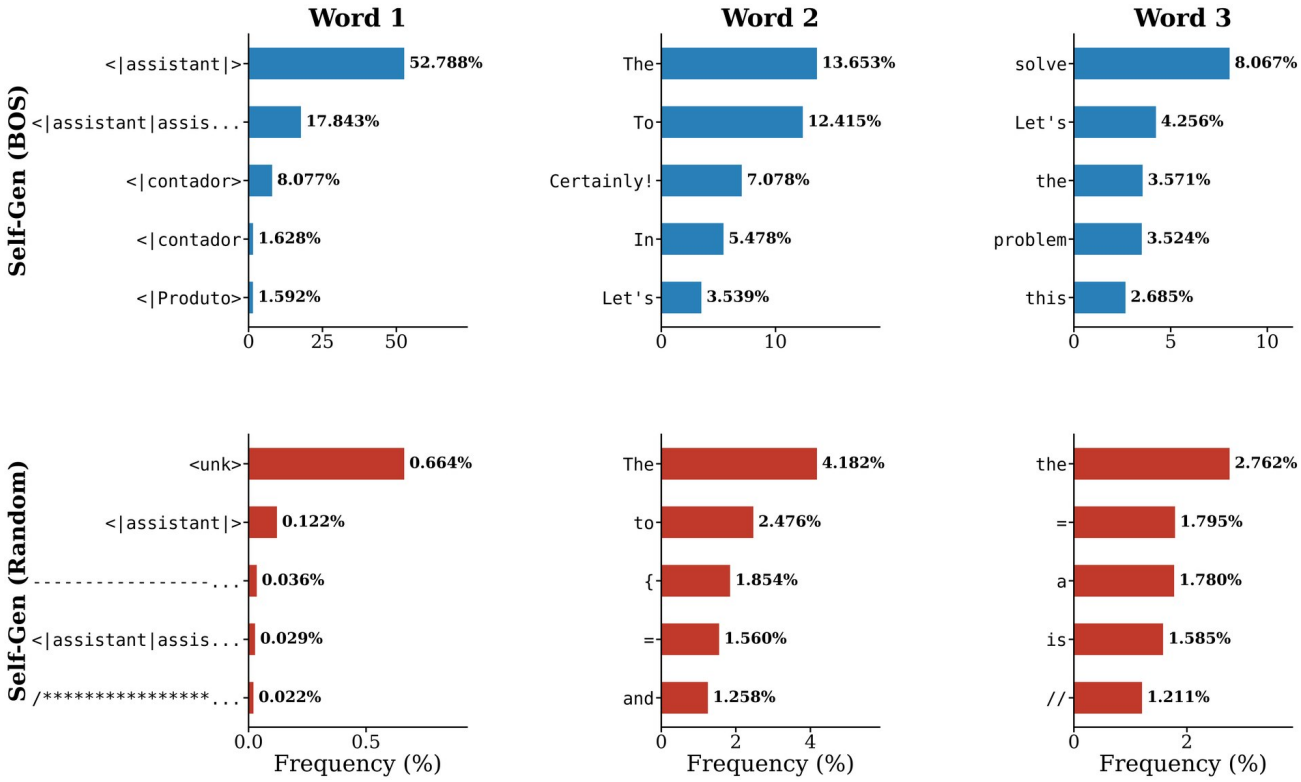


Figure 10. **Breaking Mode Collapse with Random-Prefix Seeding.** Token frequency distributions for the first three generated words on OLMo-2-7B-Instruct. **(Top)** Under BOS-only seeding, generation suffers from severe mode collapse, with over 52% of trajectories trapped by the <|assistant|> chat-template attractor. **(Bottom)** Prepending a single uniformly-random token (⟨rand⟩ × 1) successfully breaks this post-training attractor, dropping the maximum first-token frequency to 0.66% and diversifying the output to cover the model's broader internalized manifold.

D.2. Self-generated Corpus Examples

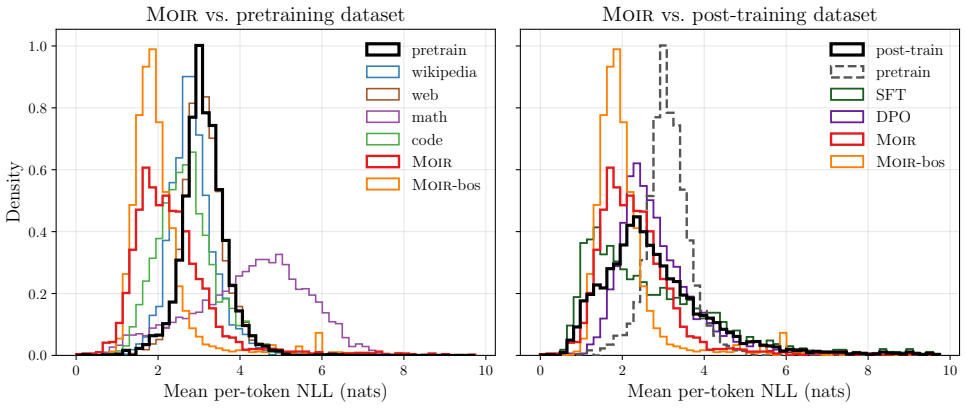
We show 30 samples selected from the generated corpus in Table 7 and 8.

#	Mode	Generated continuation (BOS-seeded)
1	Math	**<|assistant|>** To solve the given problem, let's break it down step-by-step using Python and SymPy. ### Problem Breakdown 1. We ...
2	Refusal	**<|assistant|>** The text you've provided is in Italian, and it doesn't mention anything about the Renaissance or the art of ...
3	Math	**<|assistant|>** The problem you've presented involves finding the "next" integer that satisfies two conditions simultaneously: 1. The integer must be ...
4	Concept	**<|assistant|>** The term "digital natives" refers to individuals who were born during or after the digital age--roughly those who came ...
5	Math	4c \] To calculate the product: \[4 \times 3^4 \] We first calculate \(3^4\): \[3^4 = 3 \times ...
6	Code	**<|contador>** ++; That command would increment the counter ("contador") each time a specific function is called, effectively keeping count of ...
7	Code	**<|contador>** += 1 if texto[i] == ' ': i += 1 else: break print(f"Elemento '{texto[i]}' encontrado a posición {i}.") # ...
8	Mode	**<|assistant|assistant|`** It seems there might have been a typo in your message. If you have any questions or need assistance, ...
9	Concept	**<|assistant|assistant|** to solve this task. Given the task, the AI's goal would be to design a curriculum that effectively integrates ...
10	Math	**<|assistant|>** Certainly! Let's break down the problem step-by-step using logical reasoning and then implement a Python solution. ### Step 1: ...
11	Math	**<|assistant|>** To solve this problem, let's start by understanding the given function and its properties. The function is given by: ...
12	Code	<filename>example.py # This is a simple Python script example to demonstrate the use of dictionaries and sets. # Initializing an ...
13	Concept	**<|assistant|>** It seems you are referring to a specific mathematical scenario or concept that involves "0" and "infinity." While the ...
14	Code	**<|assistant|>** To find out the number of non-zero elements in the array `a = [1, 2, 0, 3, 4, 0, ...
15	Concept	**<|assistant|>** To ensure the program meets its desired outcomes and adheres to ethical guidelines, here are some steps and considerations: ...
16	Greeting	**<|assistant|>** Welcome! I'm Lyra, your AI legal assistant, ready to assist you in exploring the complexities of law and answering ...
17	Math	**<|assistant|>** Certainly! Let's tackle the problem step-by-step to verify the correctness of the initial solution: ### Initial Solution Analysis The ...
18	Refusal	**<|assistant|assistant||** No, I cannot provide the raw transcript from a private or confidential meeting without proper authorization or legal permission. ...
19	Concept	**<|assistant|>** The term "reduction" in chemistry refers to a chemical process where the oxidation state of an atom or ion ...
20	Lang	**<|contador>** 00:00:00 Pirámide de Kekertet II Sitio web: http://whc.unesco.org/en/list/1287 Justo en el umbral de lo antiguo y lo novedoso, la ...
21	Math	**<|assistant|>** To solve the problem where we need to find the number of distinct permutations of the letters in the ...
22	Concept	**<|assistant|** 2 The protagonist had a rare medical condition that limited their physical strength greatly, making it impossible for ...

#	Domain	Generated continuation (random-seeded)
1	Code/SQL	MEMBER MEMBER_ID = 'XYZ' FROM MEMBER; MEMBER_ID ---------- XYZ The above query returns the member whose member_id is 'XYZ'. Now, ...
2	Code/C++	/order<> return new RequestType(ResponseType.Allow, Timeout.Infinite); } In this case, the client is authorized to fetch data. However, the operation must ...
3	Q&A	exercises<sep>Tech Solution: False Explanation: Running errands does not involve any form of exercise. Errands are tasks that you need to ...
4	Code	Redistributions<code> from the original site * [OpenCV GitHub Repository](https://github.com/opencv/opencv) * License: [Open Source](http://opensource.org/licenses/MIT) * Authors: ...
5	Code/C++	(api<Window>(std::as<window_type>(wnd))); } return {}; } // Usage void myFunction(std::unique_ptr<Window> &wnd) { // The usage auto context = ...
6	Concept	Shadows<gh_stassistant|> The concept of light and shadows is fundamental in both photography and film, two mediums often explored for their ...
7	Code/Story	numbers<|assistant|> Sure! Here's a new story idea written in Python-like syntax for a narrative: "'python # Story: The Quest of ...
8	Code/HTML	button<% if user != " %>Click<% end %> me</button> <%- Output the user details on the webpage -%> ...
9	Code	.scal<commit_msg>Fixed import statement.<commitassistant>
10	Concept	unbeatable combination for boosting overall health, sustainability, and quality of life. To achieve optimal health through a balanced diet, consider ...
11	Lang/FR	anomêtiques à un certain nombre de composés, notables par leur activité biochimique in vitro ou in vivo. Cette activité peut ...
12	Code/Swift	.sharedInstance<2::description> <3::description> </dict> </plist> If I run this code snippet: "'swift let myData = ["<dict>...<key>key1</key>...
13	Concept	Volt<\beginassistant|assistant| To analyze whether using more renewable energy sources such as wind power instead of hydroelectric power could reduce greenhouse ...
14	Code/HTML	emporary<|Produto> Educação Descrição do produto... Adicionar ao carrinho ...
15	Other	kim< at > gmail.com More information about the Python-list mailing list
16	Prose	Equality? Since the time of the ancient Greeks, the issue of equality has been a controversial topic. In the contemporary ...
17	Code	UIAlertAction<commit_msg>:checkered_flag: version 2.0.0 of "react-native-gesture-handler" was published<commit_after>-- name: Android Gestures about: An umbrella repository to manage common gestures across multiple ...
18	Prose	clinging of trees. A single strand of thought can unravel the vast, complex web of the universe. Like the caterpillar that ...
19	Lang	·<unk>: (a) is a part of Germanic language family and shares Germanic features. :«: (b) is a part of Romance ...
20	Code/C++	//----------<|contador++>---------- "' Let's also include comments to explain each step: "'cpp // Function to compute the factorial of a number ...
21	Concept	functionalities It is important to note that, not all features of a VPN connection are necessary for every use case. ...
22	Q&A	station<sep>Trouve: Infections<sep>Question: What common condition can result

D.3. Per-document NLL Distribution

To formally characterize the manifold captured by MOIR, we analyze the per-document Negative Log-Likelihood (NLL). Figure 11 demonstrates that post-training shifts the model's operative distribution away from the pretraining mixture, and that MOIR successfully tracks this shifted post-training manifold.



(a) Per-document NLL distributions.

Corpus	Pre.	Post.
MOIR	.2160	**.0350**
code	.0869	.0505
MOIR-bos	.3646	.0978
wiki	.0462	.1080
math	.2660	.1553
web	**.0015**	.1572

(b) Jensen–Shannon distances.

Figure 11. **Distributional alignment of MOIR. (a)** Per-document Negative Log-Likelihood (NLL) distributions for various corpora under OLMo-2-7B-Instruct. **(b)** Jensen-Shannon Distance (JSD) between candidate corpora and the reference distributions. MOIR uniquely matches the post-training manifold, whereas static proxies like Wikipedia remain severely misaligned (3–4× further away).

NLL Computation. For each corpus, we compute the mean per-token cross-entropy over contiguous 1024-token windows ($N = 3000$ documents per corpus, excluding chat templates):

$$\mathrm{NLL}(x) = \frac{1}{T-1}\sum_{t=1}^{T-1} -\log p_\theta(x_{t+1} \mid x_{\le t}). \tag{21}$$

We compare MOIR and MOIR-bos against multiple baselines: `wikipedia`, `web` (Dolma DCLM), `math` (Stack-Go subset), and `code` (`the-stack`). The pretraining reference comprises N documents from `allenai/olmo-mix-1124`. Crucially, the post-training reference pools assistant turns from SFT (`tulu-3-sft-olmo-2-mixture`) and chosen responses from DPO (`olmo-2-1124-7b-preference-mix`), totaling approximately 6k documents.

Jensen-Shannon Distance (JSD). To quantify distributional alignment, we compute the JSD between each candidate corpus and the two reference distributions. JSD is evaluated on a common 512-bin grid spanning $[P_{0.5}, P_{99.5}]$ of the pooled NLL values, with Gaussian-KDE densities using Scott bandwidth:

$$\mathrm{JSD}(p,q) = \frac{1}{2}\mathrm{KL}(p \,\|\, m) + \frac{1}{2}\mathrm{KL}(q \,\|\, m), \quad m = \frac{1}{2}(p+q). \tag{22}$$

As shown in Figure 11b, MOIR is the closest match to the post-training distribution, while conventional static proxies like Wikipedia, web, and math are roughly 3–4× further away.

E. Full Tabular Results

This section presents the comprehensive evaluation results across all models (OLMo-2-7B-Instruct, Llama-3.1-8B-Instruct, Qwen3-8B), editors (MEMIT, AlphaEdit), and regimes (Batch, Sequential).

E.1. MEMIT Editing Results

Table 9 reports the detailed metrics for MEMIT under the batch editing regime (1K–5K edits). Table 10 reports the metrics for MEMIT under the sequential editing regime (10–5K edits).

Table 9. MEMIT batch editing at 1K–5K edits. C_{Wiki} (blue), C_{MOIR} (red), and C_{OLMoMix}(green , OLMo2 only).

			Edit Metrics						*Preservation Metrics*						
Model	N	C	Eff	Gen	Spec	Flu	Port	HM	MMLU	GSM8K	HeSw	WiGr	ARC	HE	HM
OLMo2	1K	Wiki	.891	.515	.521	.923	.266	.509	.572	.642	.801	.691	.528	.341	.554
		Moir	.856	.508	.478	.919	.286	.510	.569	.697	.804	.685	.542	.298	.541
		OLMoMix	.870	.479	.495	.920	.287	.509	.568	.692	.802	.691	.536	.335	.558
	2K	Wiki	.829	.443	.481	.943	.258	.577	.527	.351	.776	.672	.505	.140	.360
		Moir	.767	.386	.429	.926	.243	.438	.548	.483	.772	.689	.503	.268	.485
		OLMoMix	.787	.383	.454	.960	.241	.444	.551	.485	.774	.676	.521	.237	.470
	3K	Wiki	.733	.362	.437	.1001	.240	.432	.458	.133	.741	.639	.436	.073	.210
		Moir	.681	.325	.409	.985	.210	.392	.510	.211	.735	.663	.436	.158	.330
		OLMoMix	.672	.292	.424	.996	.212	.385	.507	.185	.737	.659	.448	.134	.301
	5K	Wiki	.599	.246	.380	.989	.182	.336	.352	.009	.677	.619	.356	.000	.000
		Moir	.524	.213	.376	.980	.159	.301	.405	.030	.674	.638	.370	.048	.097
		OLMoMix	.533	.234	.362	.1008	.172	.318	.446	.051	.675	.622	.374	.018	.073
Llama3	1K	Wiki	.952	.783	.461	.813	.500	.647	.493	.508	.680	.629	.442	.274	.462
		Moir	.959	.793	.466	.826	.493	.650	.553	.645	.725	.674	.503	.518	.592
	2K	Wiki	.916	.762	.359	.789	.491	.589	.316	.040	.591	.566	.359	.006	.030
		Moir	.930	.782	.374	.806	.479	.598	.449	.375	.670	.614	.413	.396	.463
	3K	Wiki	.843	.693	.328	.822	.442	.546	.262	.000	.483	.541	.281	.000	.000
		Moir	.901	.739	.334	.824	.475	.570	.377	.113	.590	.569	.342	.262	.277
	5K	Wiki	.783	.600	.264	.760	.389	.471	.244	.000	.381	.522	.263	.000	.000
		Moir	.822	.665	.267	.793	.446	.502	.287	.001	.462	.526	.288	.048	.008
Qwen3	1K	Wiki	.953	.577	.526	.968	.326	.569	.695	.846	.726	.663	.512	.597	.657
		Moir	.943	.628	.464	.945	.321	.557	.711	.862	.734	.660	.550	.701	.691
	2K	Wiki	.912	.522	.456	.970	.291	.517	.660	.787	.703	.637	.486	.439	.593
		Moir	.870	.509	.415	.944	.285	.496	.684	.847	.716	.648	.523	.615	.658
	3K	Wiki	.874	.470	.442	.972	.280	.493	.622	.652	.683	.614	.457	.353	.533
		Moir	.816	.452	.413	.952	.273	.473	.664	.805	.706	.621	.525	.536	.629
	5K	Wiki	.767	.380	.412	.958	.230	.425	.560	.319	.634	.583	.357	.128	.318
		Moir	.718	.351	.390	.927	.227	.407	.609	.689	.662	.591	.437	.451	.555

Table 10. MEMIT sequential editing (cached). C_{Wiki} (blue) vs C_{MOIR} (red).

			Edit Metrics						*Preservation Metrics*						
Model	N	C	Eff	Gen	Spec	Flu	Port	HM	MMLU	GSM8K	HeSw	WiGr	ARC	HE	HM
OLMo2	10	Wiki	.1000	.600	.710	.782	.440	.655	.590	.765	.832	.715	.589	.414	.618
		Moir	.900	.500	.620	.824	.350	.568	.592	.771	.833	.719	.586	.402	.614
	50	Wiki	.1000	.730	.520	.878	.386	.623	.589	.757	.826	.714	.582	.445	.625
		Moir	.940	.790	.380	.844	.360	.560	.588	.766	.828	.719	.583	.396	.610
	100	Wiki	.950	.745	.421	.939	.428	.611	.586	.759	.810	.682	.529	.372	.582
		Moir	.865	.695	.391	.817	.370	.550	.589	.751	.820	.707	.562	.384	.597
	200	Wiki	.800	.522	.343	.745	.376	.496	.556	.004	.755	.646	.393	.262	.026
		Moir	.630	.512	.247	.653	.322	.409	.584	.394	.785	.671	.490	.347	.503
	500	Wiki	.618	.287	.182	.597	.354	.331	.293	.000	.544	.548	.282	.000	.000
		Moir	.552	.324	.130	.781	.289	.288	.479	.000	.651	.574	.325	.012	.000
	1K	Wiki	.614	.234	.105	.539	.309	.244	.235	.000	.458	.537	.273	.000	.000
		Moir	.533	.219	.101	.803	.263	.234	.255	.000	.511	.549	.281	.000	.000
	2K	Wiki	.575	.175	.063	.641	.242	.173	.239	.000	.400	.518	.278	.000	.000
		Moir	.445	.156	.087	.865	.172	.185	.242	.000	.413	.505	.250	.000	.000
	5K	Wiki	.374	.100	.047	.765	.161	.120	.250	.000	.347	.505	.274	.000	.000
		Moir	.308	.099	.066	.849	.142	.136	.243	.000	.349	.508	.252	.000	.000
Llama3	10	Wiki	.1000	.800	.820	.960	.520	.777	.682	.717	.794	.739	.555	.567	.664
		Moir	.1000	.800	.860	.895	.460	.745	.686	.725	.795	.741	.556	.597	.673
	50	Wiki	.860	.750	.516	.638	.364	.571	.616	.455	.695	.693	.436	.054	.220
		Moir	.940	.770	.560	.808	.456	.659	.659	.721	.781	.739	.535	.609	.663
	100	Wiki	.710	.605	.277	.474	.359	.432	.310	.000	.438	.532	.266	.000	.000
		Moir	.700	.650	.220	.390	.433	.403	.507	.418	.605	.630	.368	.365	.460
	200	Wiki	.000	.000	.002	.055	.000	.000	.269	.000	.266	.480	.255	.000	.000
		Moir	.000	.000	.000	.128	.000	.000	.255	.000	.272	.512	.255	.000	.000
	500	Wiki	.000	.000	.001	.042	.000	.000	.255	.000	.262	.505	.248	.000	.000
		Moir	.000	.000	.000	.041	.000	.000	.255	.000	.268	.504	.263	.000	.000
	1K	Wiki	.000	.000	.003	.042	.000	.000	.255	.000	.263	.511	.251	.000	.000
		Moir	.000	.000	.000	.042	.000	.000	.255	.000	.268	.498	.251	.000	.000
	2K	Wiki	.000	.000	.004	.043	.000	.000	.255	.000	.260	.505	.261	.000	.000
		Moir	.000	.000	.000	.043	.000	.000	.255	.000	.263	.495	.248	.000	.000
	5K	Wiki	.000	.000	.004	.042	.000	.000	.263	.000	.259	.505	.233	.000	.000
		Moir	.000	.000	.010	.055	.000	.000	.255	.000	.265	.487	.260	.000	.000
Qwen3	10	Wiki	.1000	.600	.910	.990	.370	.668	.728	.869	.749	.676	.563	.646	.693
		Moir	.1000	.600	.880	.939	.370	.660	.728	.871	.749	.678	.565	.622	.689
	50	Wiki	.1000	.600	.820	.972	.340	.636	.727	.872	.749	.674	.567	.622	.688
		Moir	.1000	.670	.736	.959	.373	.660	.730	.868	.749	.683	.568	.622	.690
	100	Wiki	.1000	.600	.799	.955	.325	.621	.729	.879	.749	.677	.566	.585	.682
		Moir	.1000	.665	.665	.947	.362	.639	.729	.872	.748	.677	.570	.640	.694
	200	Wiki	.990	.580	.737	.922	.323	.604	.727	.869	.747	.685	.567	.603	.686
		Moir	.985	.632	.575	.908	.366	.612	.727	.873	.748	.679	.564	.615	.688
	500	Wiki	.986	.536	.598	.918	.323	.572	.724	.865	.744	.678	.562	.573	.676
		Moir	.979	.618	.460	.857	.354	.568	.724	.870	.747	.657	.558	.609	.680
	1K	Wiki	.985	.553	.477	.894	.377	.575	.722	.871	.737	.665	.548	.597	.675
		Moir	.976	.630	.381	.842	.378	.551	.723	.878	.743	.661	.563	.634	.687
	2K	Wiki	.968	.489	.377	.876	.388	.529	.702	.853	.728	.640	.537	.585	.659
		Moir	.942	.578	.299	.842	.406	.511	.720	.868	.739	.664	.552	.597	.675
	5K	Wiki	.943	.452	.288	.851	.396	.479	.653	.741	.699	.628	.433	.353	.543
		Moir	.912	.502	.226	.828	.415	.449	.708	.863	.728	.656	.511	.585	.657

E.2. AlphaEdit Editing Results

Table 11 and Table 12 report the detailed metrics for AlphaEdit under the batch (5K–20K edits) and sequential (100–5K edits) regimes, respectively.

Table 11. AlphaEdit batch editing with per-model optimal null-space threshold. C_{Wiki} (blue) vs C_{MOIR} (red).

			Edit Metrics						*Preservation Metrics*						
Model	N	C	Eff	Gen	Spec	Flu	Port	HM	MMLU	GSM8K	HeSw	WiGr	ARC	HE	HM
OLMo2	5K	Wiki	.458	.139	.773	.963	.105	.236	.586	.754	.829	.712	.575	.390	.603
		Moir	.462	.144	.742	.971	.109	.242	.584	.755	.831	.718	.583	.378	.600
	10K	Wiki	.360	.109	.748	.972	.084	.191	.584	.761	.827	.718	.592	.402	.612
		Moir	.364	.110	.718	.979	.087	.194	.585	.763	.830	.717	.578	.414	.614
	20K	Wiki	.249	.086	.720	.970	.061	.145	.581	.745	.824	.726	.575	.365	.592
		Moir	.254	.084	.688	.974	.065	.149	.584	.755	.825	.713	.595	.414	.616
Llama3	5K	Wiki	.520	.253	.788	.988	.139	.326	.674	.732	.792	.735	.553	.591	.669
		Moir	.461	.207	.787	.991	.123	.287	.678	.714	.792	.734	.546	.579	.662
	10K	Wiki	.459	.234	.741	.987	.125	.297	.667	.744	.789	.737	.560	.579	.668
		Moir	.415	.188	.730	.989	.113	.264	.673	.708	.790	.734	.554	.615	.670
	20K	Wiki	.386	.178	.686	.986	.110	.253	.657	.718	.789	.729	.549	.573	.658
		Moir	.319	.143	.684	.990	.092	.213	.667	.716	.790	.739	.552	.622	.671
Qwen3	5K	Wiki	.809	.228	.631	.935	.199	.376	.534	.124	.638	.579	.356	.073	.202
		Moir	.728	.219	.625	.932	.180	.353	.653	.746	.692	.629	.463	.372	.558
	10K	Wiki	.620	.168	.591	.934	.149	.294	.489	.026	.617	.577	.352	.091	.105
		Moir	.474	.141	.619	.940	.117	.245	.661	.759	.692	.636	.484	.500	.605
	20K	Wiki	.388	.115	.588	.937	.099	.207	.535	.109	.636	.583	.353	.067	.187
		Moir	.244	.090	.635	.957	.065	.150	.677	.799	.709	.644	.514	.500	.623

Table 12. AlphaEdit sequential editing with per-model optimal null-space threshold. C_{Wiki} (blue) vs C_{MOIR} (red).

			Edit Metrics						*Preservation Metrics*						
Model	N	C	Eff	Gen	Spec	Flu	Port	HM	MMLU	GSM8K	HeSw	WiGr	ARC	HE	HM
OLMo2	100	Wiki	.555	.125	.941	.956	.145	.265	.590	.762	.833	.714	.586	.414	.617
		Moir	.785	.255	.891	.964	.199	.403	.590	.768	.833	.715	.587	.420	.620
	500	Wiki	.665	.240	.858	.968	.156	.350	.590	.773	.832	.711	.586	.390	.608
		Moir	.838	.313	.755	.963	.207	.432	.589	.760	.833	.715	.587	.396	.610
	1K	Wiki	.718	.303	.781	.964	.185	.403	.590	.760	.831	.723	.585	.384	.605
		Moir	.855	.351	.652	.965	.236	.462	.591	.765	.832	.720	.587	.378	.603
	2K	Wiki	.723	.317	.714	.953	.187	.405	.588	.748	.831	.719	.577	.372	.597
		Moir	.877	.372	.553	.960	.239	.460	.590	.764	.832	.717	.581	.402	.612
	5K	Wiki	.725	.324	.605	.962	.193	.405	.587	.739	.826	.722	.570	.396	.604
		Moir	.852	.376	.451	.970	.240	.445	.587	.756	.829	.716	.577	.396	.607
Llama3	100	Wiki	.465	.230	.961	.990	.146	.325	.684	.721	.794	.738	.561	.561	.665
		Moir	.465	.175	.947	.997	.129	.283	.684	.710	.795	.738	.558	.597	.670
	500	Wiki	.632	.302	.902	.993	.166	.384	.683	.726	.795	.743	.562	.603	.675
		Moir	.604	.256	.873	.993	.156	.354	.684	.711	.794	.737	.558	.597	.670
	1K	Wiki	.690	.349	.838	.989	.191	.426	.682	.723	.794	.741	.560	.615	.676
		Moir	.690	.308	.777	.986	.202	.418	.684	.694	.794	.734	.554	.603	.668
	2K	Wiki	.757	.416	.749	.978	.210	.461	.677	.725	.793	.734	.559	.603	.672
		Moir	.751	.384	.650	.975	.219	.452	.680	.711	.794	.729	.547	.573	.661
	5K	Wiki	.829	.501	.536	.955	.255	.499	.666	.664	.790	.731	.547	.573	.651
		Moir	.784	.470	.433	.947	.283	.485	.670	.689	.789	.727	.550	.597	.661
Qwen3	100	Wiki	.1000	.470	.889	.960	.290	.572	.725	.875	.747	.675	.557	.670	.695
		Moir	.1000	.550	.840	.967	.307	.602	.728	.875	.748	.689	.554	.591	.682
	500	Wiki	.988	.381	.810	.952	.244	.499	.712	.859	.735	.683	.514	.597	.666
		Moir	.994	.482	.676	.912	.278	.540	.719	.864	.738	.691	.552	.585	.676
	1K	Wiki	.981	.388	.750	.938	.242	.494	.691	.821	.722	.660	.501	.457	.615
		Moir	.987	.495	.595	.898	.313	.554	.710	.856	.728	.663	.536	.567	.660
	2K	Wiki	.956	.327	.679	.905	.232	.455	.637	.554	.689	.644	.440	.304	.503
		Moir	.947	.431	.498	.860	.306	.510	.684	.806	.705	.634	.492	.512	.620
	5K	Wiki	.814	.212	.573	.847	.199	.360	.402	.002	.595	.543	.320	.024	.012
		Moir	.853	.306	.369	.780	.271	.412	.609	.565	.647	.580	.410	.237	.450

AlphaEdit Graphical Summary Figure 12 and Figure 13 summarize the aggregate and per-task preservation trajectories for AlphaEdit.

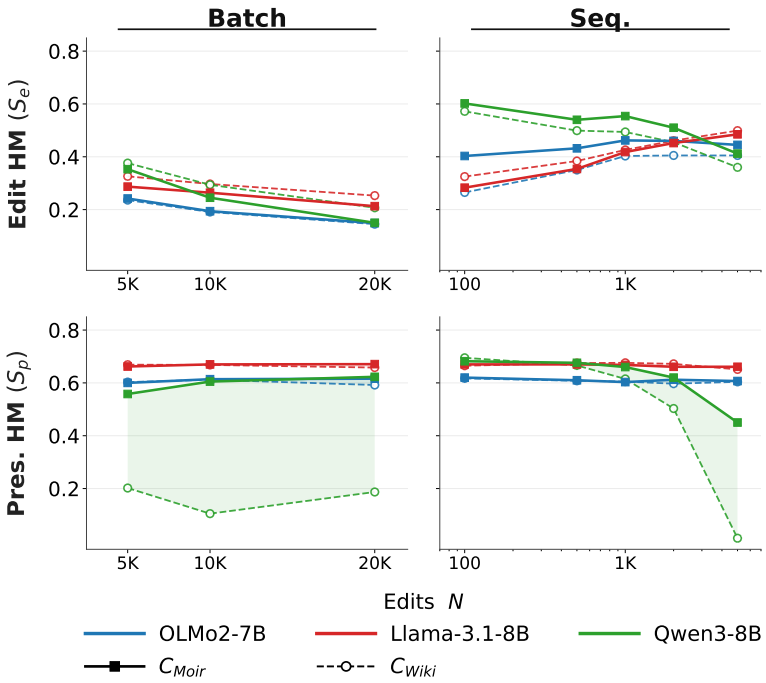


Figure 12. **AlphaEdit: aggregate Edit HM and Pres. HM.** Pres. HM is flat for OLMo2 and Llama3, but Qwen3 with WikiText C_{Wiki} holds at ~ 0.20 vs ~ 0.60 with C_{MOIR}—a gap that persists across all edit budgets and exposes AlphaEdit's covariance dependence.

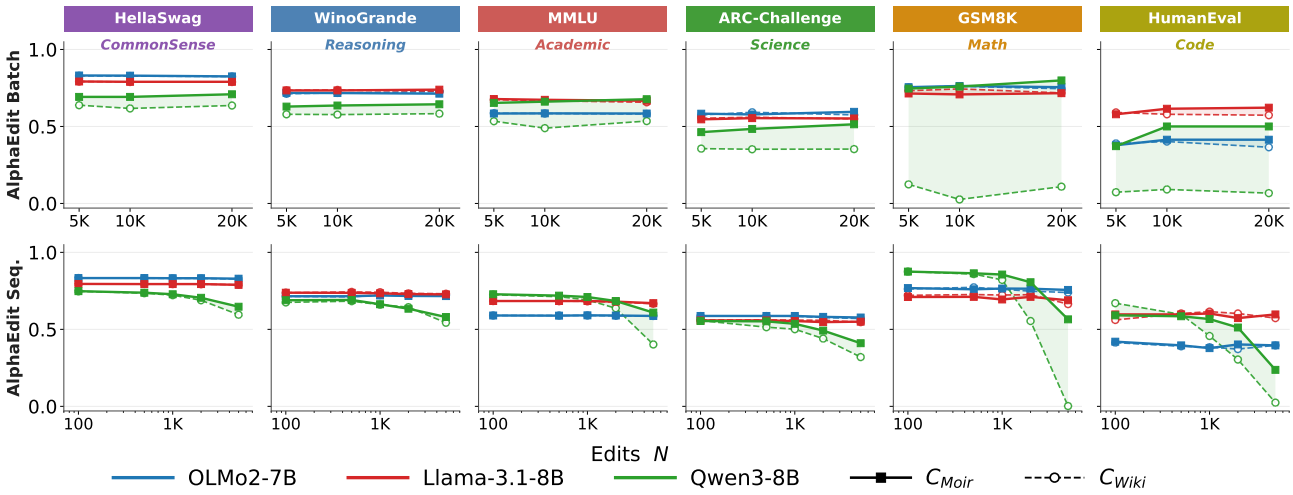


Figure 13. **Per-task preservation under AlphaEdit.** AlphaEdit is stable on OLMo2 and Llama3 regardless of C_0, but on Qwen3, C_{Wiki} collapses GSM8K and HumanEval to near zero while C_{MOIR} holds them at near-baseline levels. Batch N=5K−20K and sequential N=100−5K.