Norm Anchors Make Model Edits Last

Mingda Liu*
Institute of Science Tokyo
puppetsasya@gmail.com

Zhenghan Zhu*
Institute of Science Tokyo

Ze'an Miao*
Institute of Science Tokyo

Katsuki Fujisawa
Institute of Science Tokyo
fujisawa.k.2110@m.isct.ac.jp

Abstract

Sequential Locate-and-Edit (L&E) model editing can fail abruptly after many edits. We identify and formalize this failure as a positive *norm-feedback loop*, in which solved value vectors and edited MLP weights progressively amplify each other, degrading edit quality and eventually collapsing model capabilities. Our analysis shows that this feedback can yield approximately exponential norm growth under standard L&E dynamics, and can remain unresolved by existing increment-level regularizers or update clamps. We propose **Norm-Anchor Scaling (NAS)**, a plug-in stabilizer that breaks this loop by rescaling each solved value vector to an original-model reference norm. Across multiple LLM backbones, datasets, and L&E editors, NAS extends the usable editing horizon by more than **4**$\times$ and improves long-run editing performance by **72.2%** on average, while preserving single-edit efficacy, with only a **one-line modification and negligible computational overhead**. The code is available at https://github.com/SasyaTitech/NAS.

1 Introduction

Large language models (LLMs) store substantial factual knowledge [Petroni et al., 2019, Roberts et al., 2020], but deployed models inevitably encounter outdated, incorrect, or newly emerging facts [Lin et al., 2022, De Cao et al., 2021, Dhingra et al., 2022]. Correcting such errors by full retraining is prohibitively expensive [De Cao et al., 2021], while broad fine-tuning can modify unrelated behavior in hard-to-predict ways [Zhang and Wu, 2024]. **Knowledge editing** addresses this problem by locally modifying a model so that a targeted prompt produces a desired new fact while unrelated behaviors are preserved [De Cao et al., 2021, Mitchell et al., 2021].

Among editing approaches, **Locate-and-Edit (L&E)** [Meng et al., 2023a,b] methods have become a widely used in-weight editing paradigm. Given a factual update, they identify an FFN site and apply a localized low-rank update to the output projection so that a target key maps to a new value. This structure makes L&E computationally efficient and often more localized than large-subset fine-tuning [Wang et al., 2024b, Yao et al., 2023].

A practical editor, however, must be reusable across a stream of updates [Gupta et al., 2024]. In this standard sequential editing setting, stability is typically assessed by whether the editor can maintain edit quality, preserve unrelated and general-purpose behavior, and avoid abrupt collapse as edits accumulate [Meng et al., 2023b, Zhang et al., 2025]. Under long-stream evaluations, many existing L&E editors exhaust their usable editing lifespan within only a few thousand sequential edits, making it a central bottleneck to extend the effective horizon of in-weight editing.

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

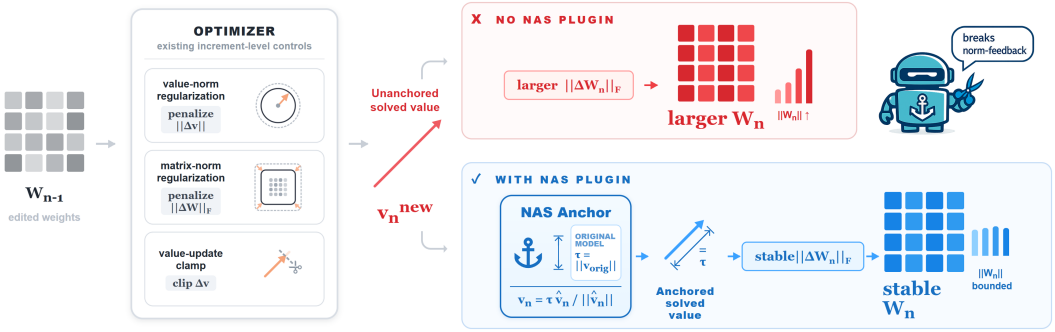


Figure 1: **Mechanism illustration.** NAS anchors the solved value before writing, breaking the norm-feedback loop left unresolved by increment-level controls.

Prior work has observed that such long-run degradation often accompanies abnormal growth in edited-parameter norms, and we see the same signature in our setting (Fig. 8 in Appendix B.3). We go beyond this phenomenon by formalizing the L&E update dynamics and identifying a self-reinforcing *norm-feedback instability*: past edits enlarge the edited matrix, causing later edits to solve value vectors at a larger scale, whose rank-one writes further amplify the same matrix. This analysis derives conditions under which the edited-matrix norm grows approximately exponentially, matching the log-linear norm trajectories observed in our experiments. The analysis also explains why existing increment-level norm controls, such as update regularization or clipping, can still leave the feedback unresolved: they constrain the update step but do not anchor the final solved value repeatedly written into the model (Fig. 1).

We view this diagnosis as the main contribution of the paper. It also points to a small fix: if the loop runs through the norm of the solved value vector, fixing that norm should break it. **Norm-Anchor Scaling (NAS)** does this. Before each write, NAS rescales the solved value vector to a reference norm measured once on the original model, and keeps its direction. Under the same scaling assumptions, the target-layer weight norm then stays below a finite bound over the edit sequence. Empirically, compared to non-NAS updates, NAS confines the target-layer norm drift to a much tighter band around its pre-edit value (Fig. 5), and substantially suppresses the drift of hidden representations (Fig. 3).

NAS is a **one-line drop-in with negligible computational overhead** (Appendix C.18). Extensive experiments against a broad set of baselines, including stability-oriented L&E editors with their own norm-related or statistical controls, show that NAS significantly delays the degradation point (by more than $4\times$ on average) and improves sequential editing performance: the average editing success increases from **51.9 to 89.3 (+37.4 percentage points; +72.2% relative)**, while preserving single-edit efficacy under stress test. Under our long-horizon *atomic sequential editing* (one request per step) on the full ZsRE (**19,086** edits) and CounterFact (**20,877** edits) streams, NAS is the *only* evaluated editing strategy that does not exhibit a clear degradation.

2 Preliminaries

2.1 Knowledge Editing Setup

Factual knowledge in LLMs is often described as a triple (s, r, o) (subject, relation, object), which can be queried by a prompt $x = \mathrm{Prompt}(s, r)$ such that the model should generate o. A knowledge editing request specifies a target object $o^\star$ for the same prompt x, and aims to modify a localized set of parameters so that the edited model assigns high probability to $o^\star$ under x while preserving unrelated behaviors.

2.2 Transformer FFN as Key–Value Memory

We focus on the FFN module at a fixed layer l. Following the key–value interpretation of FFNs [Geva et al., 2021], we view the intermediate activation as a *key* k that matches factual patterns, while the FFN output is a retrieved *value* v that encodes the information to be written into the hidden state. Let

$W_{\text{out}}^{(l)}$ denote the FFN output projection; then

$$v \;=\; W_{\text{out}}^{(l)}\, k. \tag{1}$$

For simplicity, we write $W \coloneqq W_{\text{out}}^{(l)}$ henceforth.

2.3 Locate-and-Edit Paradigm

Locate-and-Edit (L&E) methods update a localized FFN matrix W to rewrite a specific factual association. After selecting an editing site, we obtain a key vector $k_n^\star$ that represents the query (e.g., the pair (s, r)) at that site. The pre-edit value is

$$v^{\text{old}} \;\coloneqq\; W k_n^\star. \tag{2}$$

Overall update objective. The core objective is to modify W so that the target key $k_n^\star$ maps to a desired value v^{new}, while keeping the original input–output behavior of W on typical keys as unchanged as possible:

$$\min_{W'} \; \mathbb{E}_{k\sim\mathcal{D}} \left\|W'k - Wk\right\|^2 \quad \text{s.t.} \quad W'k_n^\star = v^{\text{new}}, \tag{3}$$

where $\mathcal{D}$ denotes a text-induced distribution over FFN keys.

Closed-form rank-one update of W. Let $C \coloneqq \mathbb{E}_{k\sim\mathcal{D}}[kk^\top]$ be the (pre-computed) second-moment matrix of keys. Following prior L&E methods, the constrained problem (3) admits a closed-form rank-one solution [Meng et al., 2023a]. Define the per-edit update ΔW as

$$\Delta W \;\coloneqq\; \left(v^{\text{new}} - v^{\text{old}}\right) \frac{(C^{-1}k_n^\star)^\top}{(k_n^\star)^\top C^{-1} k_n^\star}. \tag{4}$$

Then the updated matrix is

$$W' \;=\; W + \Delta W, \tag{5}$$

which enforces $W'k_n^\star = v^{\text{new}}$ while minimizing the expected disturbance $\mathbb{E}_{k\sim\mathcal{D}} \left\|W'k - Wk\right\|^2$.

Computing the target value v^{new}. The target value for the new fact is obtained by optimizing an additive correction $\Delta \in \mathbb{R}^d$ under the edit prompt $x = \text{Prompt}(s, r)$:

$$\Delta \in \arg\min_{\Delta\in\mathbb{R}^d} \mathcal{L}_{\text{NLL}}\big(o^\star \mid x;\, v^{\text{old}} + \Delta\big), \tag{6}$$

where $\mathcal{L}_{\text{NLL}}$ is evaluated with the FFN output at the editing site replaced by $v^{\text{old}} + \Delta$. The target value is then defined as

$$v^{\text{new}} \coloneqq v^{\text{old}} + \Delta. \tag{7}$$

2.4 Sequential Editing

Sequential editing applies a stream of requests $\{(s_n, r_n, o_n^\star)\}_{n=1}^T$ to the same base model. Focusing on the edited matrix W, the parameter trajectory follows

$$W_0 = W_{\text{base}}, \qquad W_n = W_{n-1} + \Delta W_n, \qquad n = 1, \ldots, T. \tag{8}$$

For the n-th request, the pre-edit value is given by

$$v_n^{\text{old}} \;=\; W_{n-1} k_n^\star. \tag{9}$$

and the closed-form per-edit update is

$$\Delta W_n \;\coloneqq\; \left(v_n^{\text{new}} - v_n^{\text{old}}\right) \frac{(C^{-1}k_n^\star)^\top}{(k_n^\star)^\top C^{-1} k_n^\star}. \tag{10}$$

In the next section, we analyze the induced dynamics of $\|W_n\|^2$ and relate them to the norm statistics of v_n^{old} and v_n^{new}.

3 Analysis and Method

3.1 Computing $\|W_n\|^2$

In this section, we conduct both the analysis and all sequential-editing experiments under the Euclidean metric. We focus on the most basic locate-and-edit (L&E) update without additional statistical constraints, and thus set $C = I$ for clarity. The general $C \neq I$ case follows analogously; see Appendix B.5. Under this setting, Eq. (10) becomes:

$$\Delta W_n = \frac{\left(v_n^{\text{new}} - v_n^{\text{old}}\right) k_n^{\star\top}}{\|k_n^\star\|^2}. \tag{11}$$

Combining Eqs. (8) and (11) yields an explicit recursion for $\|W_n\|^2$.

lemma 3.1. *The squared norm of the edited weight matrix satisfies*

$$\|W_n\|^2 = \|W_{n-1}\|^2 + \frac{\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2}{\|k_n^\star\|^2}. \tag{12}$$

The proof is deferred to Appendix B.2.

Scaling assumptions. The analysis uses two relations that we measure rather than derive.

Assumption A1. $\|k_n^\star\|^{-2} \approx K$ for a constant $K > 0$ across edit steps.

Assumption A2. Given W_{n-1}, the squared value norms are approximately linear in $\|W_{n-1}\|^2$:

$$\mathbb{E}\left[\|v_n^{\text{new}}\|^2 \mid W_{n-1}\right] \approx s_{\text{new}}\|W_{n-1}\|^2 + b_{\text{new}}, \quad \mathbb{E}\left[\|v_n^{\text{old}}\|^2 \mid W_{n-1}\right] \approx s_{\text{old}}\|W_{n-1}\|^2 + b_{\text{old}},$$

with $s_{\text{new}} > s_{\text{old}} > 0$.

Appendix B.3 checks both relations on our backbones (Figs. 6 and 7; linear fits with $R^2 \geq 0.986$). Prop. 3.2 and Cor. 3.3 hold to the extent that these relations hold.

Exponential growth of $\|W_n\|^2$. Under Lemma 3.1 and Assumptions A1–A2, taking expectation on both sides of Eq. (12) yields the following consequence.

Proposition 3.2. *Under Assumptions A1 and A2, there exist constants $\alpha \in \mathbb{R}$ and $R > 1$ such that*

$$\mathbb{E}\|W_n\|^2 \approx R^n\,\mathbb{E}\|W_0\|^2 + \alpha\,(R^n - 1). \tag{13}$$

Consequently, $\mathbb{E}\|W_n\|^2$ grows exponentially with the number of edits.

The proof is deferred to Appendix B.3.

Verification via curve fitting. We fit $\|W_n\|$ as a function of n and observe a clear exponential trend (Fig. 2). In particular, a linear fit on $\log(\cdot)$-scaled trajectories corroborates the predicted exponential growth.

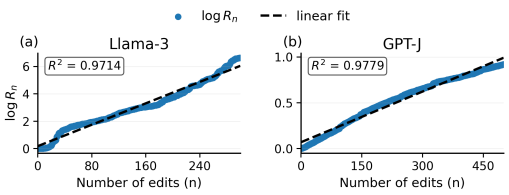


Figure 2: **Log-linear growth of weight-norm ratio under sequential editing.** We measure $\log R_n$ as a function of edit step n ($R_n = \|W_n\|/\|W_0\|$) for Llama-3 and GPT-J using MEMIT. Linear fits (dashed) achieve high R^2, supporting an approximately exponential increase of R_n with the number of edits.

3.2 Method

Motivation. Section 3.1 suggests that under unconstrained locate-and-edit (L&E) updates, the edited weight norm can grow rapidly with the number of edits. Empirically, $\|v_n^{\text{new}}\|^2$ increases with the current state (e.g., $\|W_{n-1}\|^2$; Fig. 7), forming a positive feedback loop: larger $\|W\|^2$ induces larger target values, which in turn further amplify $\|W\|^2$ through the rank-one updates.

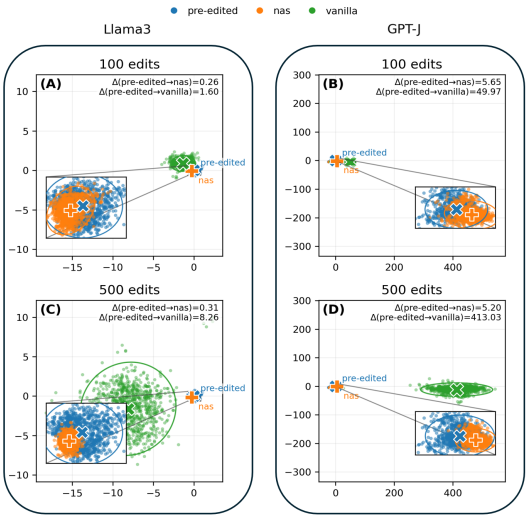


Figure 3: Hidden representation drift under sequential L&E updates. We probe the target-module representations on 1,000 held-out factual prompts for *pre-edited* (blue), *vanilla* (green), and *vanilla*+NAS (orange) after 100 and 500 edits, and visualize them in a shared 2D PCA space (PCA fit on *pre-edited*). Cross markers indicate state-wise means; ellipses denote 95% confidence regions. We report $\Delta(pre \to \cdot) = \|\mu_\cdot - \mu_{pre}\|_2$ (centroid distance in the original hidden space).

Norm anchoring via rescaling. We break this feedback by explicitly controlling the magnitude of the injected value vector. Let

$$\tau \;:=\; \left(\mathbb{E}\|v^{\text{new}}\|^2\right)^{1/2} > 0 \;\text{ measured on an original (unedited) model,} \tag{14}$$

where the expectation is estimated by performing pilot edits on N randomly sampled facts and averaging the resulting $\|v^{\text{new}}\|^2$ (Implementation details and ablation of N in Appendix A.3).

For each edit step n, suppose the base editor produces an unconstrained target value $\hat{v}_n^{\text{new}}$. We then rescale it as

$$v_n^{\text{new}} \;\leftarrow\; \frac{\tau}{\|\hat{v}_n^{\text{new}}\|}\hat{v}_n^{\text{new}}, \tag{15}$$

which enforces $\|v_n^{\text{new}}\| = \tau$ by construction. Intuitively, this introduces negative feedback: as the system drifts and the unconstrained $\|\hat{v}_n^{\text{new}}\|^2$ grows, the rescaling factor shrinks accordingly.

Implication for weight-norm dynamics. We next show that norm anchoring removes the divergence predicted by the unconstrained analysis.

Corollary 3.3. *Under Assumption A1 and the v^{old} relation in Assumption A2, if Eq.* (15) *is applied at every step n, then there exist constants $\beta > 0$, $r \in (0,1)$ such that*

$$\mathbb{E}\|W_n\|^2 \approx r^n\,\mathbb{E}\|W_0\|^2 + \beta\,(1 - r^n). \tag{16}$$

Consequently, $\mathbb{E}\|W_n\|^2$ remains bounded as $n \to \infty$.

The proof is deferred to Appendix B.4.

Behavior as $n \to \infty$. The limit $\beta = (\tau^2 - b_{\text{old}})/s_{\text{old}}$ is the weight-norm level at which the expected increment in Lemma 3.1 is zero, i.e., $\tau^2 = s_{\text{old}}\beta + b_{\text{old}}$. The condition $\beta > \mathbb{E}\|W_0\|^2$ is equivalent to $\tau^2 > \mathbb{E}[\|v^{\text{old}}\|^2 \mid W_0]$, the same observation that makes norms grow under vanilla editing (Fig. 7). With the anchor active from the first edit, $\mathbb{E}\|W_n\|^2$ therefore rises and levels off at β; it does not fall below $\mathbb{E}\|W_0\|^2$. Reaching β does not stop editing: each write is still $(v_n^{\text{new}} - v_n^{\text{old}})k_n^{\star\top}/\|k_n^\star\|^2$ with both value norms close to τ, so its size does not depend on n. This is different from stabilizers whose updates shrink as edits accumulate (Section 5). Cor. 3.3 concerns the stability of the write dynamics. It does not bound forgetting of earlier edits, which we measure in Section 4. Derivations are in Appendix B.4.

Hidden representation drift. To probe activation stability under long sequential updates, we sample target-layer representations on 1,000 *held-out* factual prompts that are disjoint from the edit stream, from three model states: *pre-edited*, *vanilla* L&E, and *vanilla*+NAS. Here, *vanilla* denotes MEMIT without NAS. At 100 and 500 cumulative edits, we project representations into a shared 2D PCA space. Figure 3 shows a clear failure mode of vanilla L&E: as edits accumulate, representations

Table 1: RQ1: Long-Horizon sequential model editing results. Reported values are mean±std for each metric. For each model: Pre-edited is separated by a rule; Non-LE baselines appear above the next rule; LE-family methods appear below. Best in **bold**, second-best in underlined.

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3	Pre-edited	7.18±0.13	9.39±0.09	89.79±0.21	568.53±1.11	3.55±0.04	36.11±0.24	35.06±0.37	32.09±0.30
	FT	68.69±0.30	61.99±0.21	40.91±0.38	325.90±0.41	1.44±0.05	8.58±0.04	6.96±0.05	17.30±0.23
	UltraEdit	60.06±0.20	52.90±0.18	43.88±0.14	558.79±0.70	11.70±0.34	77.61±0.13	75.15±0.33	**50.26±0.13**
	RLEdit	65.38±0.29	49.18±0.26	47.86±0.30	596.19±0.11	12.18±0.33	81.59±0.35	78.77±0.17	27.03±0.35
	MEMIT	50.95±0.14	50.95±0.19	49.05±0.23	244.21±0.29	4.86±0.13	0.95±0.03	0.22±0.06	0.22±0.06
	PRUNE	49.88±0.21	49.87±0.13	50.13±0.27	495.37±0.18	2.77±0.11	0.00±0.00	0.00±0.00	0.00±0.00
	RECT	49.91±0.15	49.91±0.38	50.09±0.28	336.42±0.24	4.07±0.09	0.00±0.00	0.00±0.00	0.00±0.00
	AlphaEdit	66.27±0.34	58.75±0.20	49.98±0.17	555.21±1.06	4.12±0.12	22.30±0.32	20.34±0.15	1.57±0.03
	NAS	**97.95±0.32**	**82.71±0.19**	**60.44±0.23**	**619.72±0.41**	**31.88±0.32**	**93.17±0.35**	**88.30±0.33**	32.15±0.13
Qwen2.5	Pre-edited	12.58±0.17	15.26±0.25	86.24±0.28	338.63±0.28	6.87±0.09	34.83±0.14	33.76±0.16	38.60±0.20
	FT	88.45±0.14	**69.30±0.26**	42.18±0.13	532.51±0.20	2.70±0.03	21.15±0.35	10.23±0.19	3.42±0.10
	UltraEdit	63.03±0.21	55.78±0.23	40.34±0.33	575.56±0.27	18.33±0.17	65.76±0.19	62.81±0.23	28.19±0.17
	RLEdit	75.50±0.27	53.83±0.36	46.09±0.16	605.33±0.25	14.29±0.35	93.75±0.33	85.37±0.20	35.61±0.20
	MEMIT	51.27±0.25	51.28±0.10	48.73±0.14	524.64±0.11	0.78±0.02	0.00±0.00	0.00±0.00	0.00±0.00
	PRUNE	53.16±0.17	53.17±0.14	46.83±0.15	531.26±0.22	0.16±0.05	0.00±0.00	0.00±0.00	0.00±0.00
	RECT	51.92±0.16	52.03±0.29	48.03±0.13	512.85±1.16	0.37±0.03	0.00±0.00	0.01±0.00	0.00±0.00
	AlphaEdit	84.59±0.10	58.41±0.23	68.96±0.38	**626.15±0.60**	**33.45±0.38**	**97.25±0.21**	86.17±0.29	40.16±0.10
	NAS	**90.44±0.29**	55.05±0.27	**73.21±0.22**	624.10±0.28	30.63±0.37	96.84±0.23	**88.10±0.30**	**42.52±0.34**
GPT-J	Pre-edited	15.12±0.11	17.55±0.35	83.71±0.21	622.18±0.23	29.68±0.16	27.23±0.16	26.43±0.31	27.23±0.26
	FT	57.33±0.25	54.90±0.22	44.75±0.23	576.65±0.10	3.09±0.07	11.85±0.27	10.26±0.17	0.51±0.01
	UltraEdit	56.29±0.19	49.42±0.14	51.21±0.26	498.91±0.58	14.98±0.18	65.76±0.35	62.81±0.36	**28.19±0.29**
	RLEdit	78.10±0.35	54.86±0.18	48.78±0.34	579.45±0.33	16.76±0.36	70.51±0.26	66.69±0.34	24.00±0.29
	MEMIT	49.47±0.13	49.54±0.27	50.54±0.37	509.95±0.43	4.71±0.14	2.01±0.06	0.14±0.00	0.14±0.01
	PRUNE	48.86±0.22	49.01±0.38	50.94±0.16	448.54±1.00	4.74±0.03	1.06±0.02	0.03±0.02	0.03±0.01
	RECT	48.79±0.25	48.74±0.30	51.26±0.35	240.35±0.16	4.75±0.11	0.00±0.00	0.00±0.00	0.00±0.00
	AlphaEdit	62.65±0.37	54.57±0.21	54.73±0.20	508.65±0.34	2.90±0.12	86.13±0.23	79.43±0.23	21.84±0.10
	NAS	**98.73±0.12**	**90.68±0.19**	**60.75±0.22**	**584.59±1.19**	**40.37±0.29**	**92.08±0.33**	**85.04±0.10**	22.24±0.19

exhibit a substantial centroid shift and markedly increased dispersion relative to the pre-edited distribution, and the drift amplifies sharply from 100 to 500 edits. In contrast, NAS-stabilized updates keep the representation cloud tightly aligned with the pre-edited distribution. This observation suggests that sequential degradation is accompanied by an out-of-distribution (OOD) shift in target-layer activations, which can cause downstream layers to operate outside their pre-training regime and thereby trigger broad behavioral failures. We next quantify these effects under long-horizon sequential editing in Section 4.

4 Experiments

In this section, we evaluate NAS and answer the following research questions (RQs):

- **RQ1.** Across multiple backbone models and datasets under long-horizon *atomic sequential editing*, does NAS achieve superior editing performance and stronger stability compared to existing methods?
- **RQ2.** During long-horizon sequential editing, does NAS better preserve the model's general capability than existing methods, thereby mitigating degradation on general-purpose tasks?
- **RQ3.** As a plug-and-play component that requires only *a single line of code* to integrate, can NAS suppress norm explosion when attached to different Locate-and-Edit (L&E) baselines, while simultaneously improving editing performance and overall model behavior?
- **RQ4.** Does Norm Anchoring Trade Single-Edit Efficacy for Long-Horizon Stability?

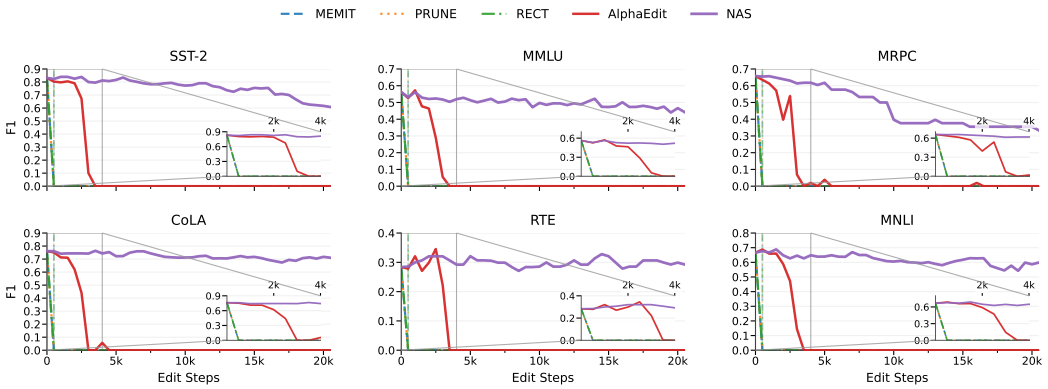


Figure 4: RQ2: GLUE performance (F1) during sequential editing. NAS preserves base-task performance substantially longer than baselines. Insets zoom into the early-edit region (0–4k); inset x-ticks are shown at 2k and 4k for readability.

4.1 Experimental Setup

Additional details on the setup are in Appendix A.

Backbone Models. Our main experiments use Llama3-8B [Grattafiori et al., 2024], Qwen2.5-7B [Qwen et al., 2025], and GPT-J [Wang and Komatsuzaki, 2021]. As additional backbones, we also conduct supplementary evaluations on GPT2-XL [Radford et al., 2019] (Appendix C.9).

Datasets and Metrics. We adopt two widely used knowledge editing benchmarks, CounterFact [Meng et al., 2023a] and ZsRE [Levy et al., 2017], and follow standard protocols to report *Efficacy*, *Generalization*, *Specificity*, *Fluency*, *Consistency*, and *Score*. To characterize long-horizon failure under sequential editing, we additionally report the *collapse point* (CP@60) (see Appendix A.2.2). Beyond the main CounterFact and ZsRE evaluations, we report auxiliary benchmarks in the appendix. WikiBigEdit [Thede et al., 2025] provides a large-scale stream of real-world Wikidata edits for longer-horizon evaluation and additional L&E ablations (Appendix C.11). MQuAKE-CF-3k-v2 [Zhong et al., 2023] probes counterfactual multi-hop editing (Appendix C.4), while WikiRecent and WikiBio from KnowEdit [Zhang et al., 2024] provide shorter recent and biographical factual-update streams for breadth checks (Appendix C.5).

Baselines. We evaluate a diverse set of baselines spanning both *non-L&E* and *L&E* paradigms. For non-L&E in-weight editors, we include FT [Zhu et al., 2020], as well as the recent lifelong editors UltraEdit [Gu et al., 2025] and RLEdit [Li et al., 2025]. For L&E methods, we include MEMIT [Meng et al., 2023b], PRUNE [Ma et al., 2024], RECT [Gu et al., 2024], and AlphaEdit [Fang et al., 2024]. In Appendix C.10, we report results for recent competitive L&E methods ENCORE [Gupta et al., 2025b] and LyapLock [Wang et al., 2025b], as official hyperparameters for our main backbones are unavailable. We also report representative *backbone-frozen* editors in Appendix C.10 , including GRACE [Hartvigsen et al., 2023], MELO [Yu et al., 2023], WISE [Wang et al., 2024a], and MEMOIR [Wang et al., 2025a].

General language benchmarks. To assess how different editing methods affect the base capabilities of the models, we measure performance on standard language understanding benchmarks during large number of sequential edits. We include GLUE-style tasks [Wang et al., 2019] such as SST [Socher et al., 2013], MRPC [Dolan and Brockett, 2005], RTE [Bentivogli et al., 2009], CoLA [Warstadt et al., 2019], and MNLI [Williams et al., 2018], and MMLU [Hendrycks et al., 2021] (see Appendix A.5 for details).

4.2 Long-horizon Sequential Editing(RQ1)

We evaluate editors under *atomic sequential editing* on the *full* CounterFact (20,877 edits) and ZsRE (19,086 edits) streams: each request edits exactly one fact and immediately commits a weight update before proceeding to the next. Main results are summarized in Table 1; results on GPT-2XL are provided in Appendix C.9.

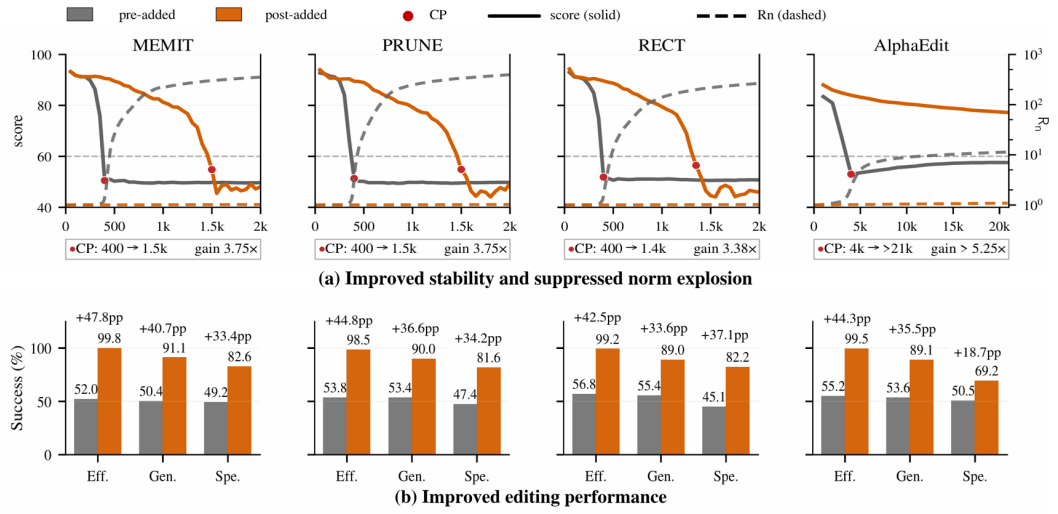


Figure 5: RQ3: **NAS improves stability and editing performance under long-horizon sequential editing.** **(a)** Dual-axis trajectories: solid lines show post-edit score, dashed lines show R_n; the red dot marks the collapse point (CP). **(b)** Editing quality at baselines' CP, reported as Eff./Gen./Spe. Comparing pre-added (gray) vs. post-added (orange).

Observation 1: NAS yields the strongest long-horizon sequential edit performance. Aggregating results over *both* CounterFact and ZsRE and all three backbones, and comparing each metric to the *strongest baseline under the same evaluation block*, NAS improves edit performance by **+11.5pp** (Eff.) and **+9.9pp** (Gen.) on average, with specificity essentially unchanged (within 0.3pp on average). This matches the smaller representation drift under NAS (Fig. 3). NAS also surpasses recent lifelong-focused editors such as ULTRAEDIT and RLEDIT on most metrics, often by a wide margin. The advantage is most pronounced on GPT-J, where NAS boosts CounterFact efficacy from **78.1** to **98.7** (**+20.6pp**) and generalization from **54.9** to **90.7** (**+35.8pp**) over the strongest baseline, while also improving specificity (**54.7** $\rightarrow$ **60.8**, **+6.0pp**).

Observation 2: NAS keeps generations fluent and improves consistency. Averaged over the three backbones, NAS improves fluency by **+8.9** and consistency by **+13.5** over the best baseline in the same block; on Llama3-8B, consistency rises from **12.2** to **31.9**.

Batched granularity. Table 1 edits one fact per step. MEMIT, PRUNE, RECT, and AlphaEdit also support batched updates and were tuned for them, so we rerun the L&E family with 500 facts per step on the same streams (Table 2). NAS† (AlphaEdit+NAS under this protocol) is best at every checkpoint, and its lead over AlphaEdit grows with stream length: on CounterFact from $+2.1$ to $+7.1$ points on Llama3-8B and from $+4.9$ to $+17.9$ on GPT-J. Batching slows the norm-feedback loop but does not remove it. Editors evaluated under their own official configurations (GRACE, WISE, MEMOIR, ENCORE, LyapLock) are reported in Appendix C.10.

4.3 General Capability Retention (RQ2)

To test whether long-horizon sequential editing compromises a model's *general capability*, we track F1 on *GLUE-style tasks* as edits accumulate. We perform sequential L&E updates over the *entire* CounterFact stream (20,877 edits) for five representative editors (MEMIT, PRUNE, RECT, AlphaEdit, and NAS), and plot the capability trajectories throughout the editing process (Fig. 4; insets zoom into 0–4k edits). We observe consistent trends on additional backbones (Qwen2.5 and GPT-J), with full results in Appendix C.6.

Observation: NAS extends the capability-preserving horizon by $\geq 5\times$. The prior L&E baselines fall to near-zero GLUE-style performance within the first few thousand edits (typically $\leq$4k). NAS keeps non-trivial performance over all 20,877 edits; most tasks stay stable, and MRPC declines more but does not approach zero.

Table 2: Sequential editing with 500 facts per step. CounterFact: Score (harmonic mean of probability-based Eff./Gen./Spe.) after 10k and 20k edits. ZsRE: raw-accuracy Eff./Gen./Spe. after the full stream (19,086 edits). NAS† = AlphaEdit+NAS. –: not run.

	CounterFact (Score)				ZsRE (Eff./Gen./Spe.)	
	Llama3-8B		GPT-J		Llama3-8B	GPT-J
Method	@10k	@20k	@10k	@20k	full	full
MEMIT	58.59	53.83	–	–	1.9/1.9/1.2	–
PRUNE	61.08	55.80	–	–	–	–
RECT	64.98	57.11	–	–	–	–
AlphaEdit	79.59	70.77	79.77	62.45	86.5/82.1/38.2	56.0/48.7/21.1
NAS†	**81.68**	**77.83**	**84.65**	**80.30**	**90.5/86.4/40.6**	**82.6/73.4/30.1**

4.4 Plug-in Gains in Stability and Performance (RQ3)

We evaluate NAS as a plug-in for L&E editors by attaching NAS to four representative L&E baselines with only a *one-line* integration. (MEMIT, PRUNE, RECT, and AlphaEdit) and run long-horizon sequential editing on CounterFact. Main results are reported on Llama3-8B, and we replicate the same study on GPT-J in Appendix C.8. Unless otherwise stated, we report editing quality at a shared reference point—the collapse point (CP) of each corresponding base editor.

Observation 1: NAS suppresses norm explosion and delays collapse across base editors. Figure 5(a) shows dual-axis trajectories during sequential editing, where solid lines track the post-edit score and dashed lines track the norm statistic R_n. Across all four editors, the base variants exhibit rapid growth in R_n followed by a sharp collapse in score. After integrating NAS, the explosive growth is suppressed and the collapse point is substantially pushed back: $3.75\times$ for MEMIT and PRUNE (CP: 400 $\rightarrow$ 1.5k), $3.38\times$ for RECT (400 $\rightarrow$ 1.4k), and $> 5.25\times$ for AlphaEdit (4k $\rightarrow$ $>$21k). This indicates that NAS acts as a robust stabilization plug-in across heterogeneous L&E update rules.

Observation 2: NAS improves editing quality for each base editor. NAS-augmented editors achieve substantially higher post-edit success on Eff./Gen./Spe. at the reference point, as shown in Fig. 5(b). Averaged across editors and metrics, NAS improves success by **+37.4 percentage points** (51.9% $\rightarrow$ 89.3%), corresponding to a **+72.2% relative** increase. The gains hold for efficacy, generalization, and specificity alike.

We provide plug-in general-capability results at the same reference point in Appendix C.7.

4.5 Single-Edit Plasticity under Norm Anchoring (RQ4)

A natural concern is that NAS may improve long-horizon stability simply by weakening each individual edit. To test this, we evaluate independent single edits on 1,000 CounterFact examples, where each edit starts from the original base model and is evaluated in isolation. We compare each base editor with and without NAS under the same editing configuration. As shown in Table 3, NAS does not systematically reduce single-edit efficacy.

We further stress-test the strongest intervention cases by selecting the top 25% examples whose solved value vectors are most strongly downscaled by NAS. Even in this hard subset, AlphaEdit+NAS remains neutral to slightly better, while MEMIT+NAS shows only a small score drop. These results indicate that the long-horizon gains of NAS are not explained by a broad stability–plasticity trade-off.

Appendix C reports further checks: anchor mis-specification, non-stationary edit orders, short streams and retention (App. C.14), multi-hop editing (App. C.4), long-form edits on UnKEBench (App. C.15), composition with SPHERE (App. C.16), clamp sweeps (App. C.12), and switching the anchor off and on mid-stream (App. C.13).

5 Related Work

Model editing methods can be broadly divided into *in-weight* editors, which overwrite pretrained parameters, and *backbone-frozen* editors, which store edits in auxiliary modules or external mem-

Table 3: Single-edit plasticity under NAS. Full absolute numbers are in Appendix C.1.

Editor	Split	NAS effect	ΔEff.	ΔGen.	ΔSpe.	ΔScore
MEMIT	All	On – Off	0.00	+1.80	−0.06	+0.70
AlphaEdit	All	On – Off	+1.00	+2.90	−0.18	+1.33
MEMIT	Top-25%	On – Off	−0.80	−2.80	+0.16	−1.17
AlphaEdit	Top-25%	On – Off	0.00	0.00	+0.64	+0.24

ory [Wang et al., 2024b]. Non-L&E in-weight methods include FT [Zhu et al., 2020], MEND [Mitchell et al., 2021], UltraEdit [Gu et al., 2025], and RLEdit [Li et al., 2025], which directly optimize, predict, or learn parameter updates without an explicit locate-and-edit step. Backbone-frozen methods such as GRACE [Hartvigsen et al., 2023], MELO [Yu et al., 2023], WISE [Wang et al., 2024a], and MEMOIR [Wang et al., 2025a] avoid overwriting the backbone by routing edits through adapters or memory, but require additional inference-time components.

Our work focuses on *Locate-and-Edit (L&E)* methods, which perform localized updates at causal FFN sites. ROME rewrites individual facts with rank-one updates [Meng et al., 2023a], while MEMIT extends this idea to mass editing with structured multi-layer updates [Meng et al., 2023b]. Recent sequential L&E variants, including AlphaEdit, PRUNE, RECT, ENCORE, and LyapLock, introduce subspace constraints, perturbation control, relative-change restrictions, norm regularization, or long-run preservation objectives to reduce interference and delay degradation [Fang et al., 2024, Ma et al., 2024, Gu et al., 2024, Gupta et al., 2025b, Wang et al., 2025b]. NAS is complementary to these approaches: it does not change the locate step, the editor objective, or add inference-time modules, and only rescales the final solved value vector written by each edit.

Norm growth in sequential editing. Norm growth during sequential L&E editing has been reported before. Gupta et al. [2025a] show that the Frobenius norm of the edited matrices grows steadily and that the activations of the edited layer shift with it; they do not give a mechanism or a fix. ENCORE [Gupta et al., 2025b], from the same group, responds with early stopping and a norm regularizer. HSE [Fang et al., 2025] proves that the weight norm converges under its own update rule (its Cor. 2); the proof relies on a replay term whose updates shrink as edits accumulate and on a constraint proportional to the current weights. Our analysis is about the standard unconstrained L&E update: Prop. 3.2 explains why it diverges, and NAS ties the written value to a statistic of the original model rather than of the current one, so the size of each write does not shrink (RQ4). Table 4 summarizes the differences. Adding NAS to ENCORE raises its WikiBigEdit score from 3.83 to 63.50 (Appendix C.11).

Table 4: NAS compared with prior norm-related work on sequential editing.

Work	Explains standard L&E	Constraint reference	Write size as n grows
Gupta et al. [2025a]	no (empirical study)	no intervention	–
ENCORE [Gupta et al., 2025b]	no	current weights	not analyzed
HSE [Fang et al., 2025]	no (proof for HSE only)	current weights	shrinks
NAS (ours)	yes (Prop. 3.2)	original model	does not shrink

6 Conclusion, Limitation and Future Work

We identify a positive norm-feedback loop in sequential L&E editing and propose NAS, a vector-level norm-control method that breaks this loop by anchoring the solved value vector rather than only regularizing update increments. This simple intervention substantially improves editing lifespan and long-run performance with negligible overhead. A limitation is that NAS uses a fixed original-model reference norm, which may be suboptimal when edit difficulty, locality requirements, or the edit distribution changes substantially; future work should explore adaptive or conditional anchoring while preserving the plug-in simplicity of NAS.

Acknowledgments and Disclosure of Funding

This work was supported by the Fujisawa Laboratory, Department of Mathematical and Computing Science, School of Computing, Institute of Science Tokyo. The authors declare no competing interests.

References

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference, TAC 2009, Gaithersburg, Maryland, USA, November 16-17, 2009*. NIST, 2009. URL https://tac.nist.gov/publications/2009/additional.papers/RTE5_overview.proceedings.pdf.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.522. URL https://aclanthology.org/2021.emnlp-main.522/.

Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. Everything is Editable: Extend Knowledge Editing to Unstructured Data in Large Language Models. In *International Conference on Learning Representations*, 2025.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022. doi: 10.1162/tacl_a_00459. URL https://aclanthology.org/2022.tacl-1.15/.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://aclanthology.org/I05-5002/.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Shi Jie, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. *arXiv preprint arXiv:2410.02355*, 2024.

Quntian Fang, Zhen Huang, Zhiliang Tian, Minghao Hu, Dongsheng Li, Yiping Yao, Xinyue Fang, Menglong Lu, and Guotong Geng. Hippocampal-like Sequential Editing for Continual Knowledge Updates in Large Language Models. In *Advances in Neural Information Processing Systems*, 2025.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories, 2021. URL https://arxiv.org/abs/2012.14913.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783. AI@Meta.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue, 2024. URL https://arxiv.org/abs/2401.04700.

Xiaojie Gu, Ziying Huang, Jia-Chen Gu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in language models, 2025. URL https://arxiv.org/abs/2505.14679.

Akshat Gupta, Anurag Rao, and Gopala Krishna Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:266999650.

Akshat Gupta, Christine Fang, Atahan Ozdemir, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. Norm Growth and Stability Challenges in Localized Sequential Knowledge Editing, 2025a. URL https://arxiv.org/abs/2502.19416.

Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. Lifelong knowledge editing requires better regularization, 2025b. URL https://arxiv.org/abs/2502.01636.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors, 2023. URL https://arxiv.org/abs/2211.11031.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AnyEdit: Edit Any Knowledge Encoded in Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *CoRR*, abs/1706.04115, 2017. URL http://arxiv.org/abs/1706.04115.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models, 2025. URL https://arxiv.org/abs/2502.05759.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Qingyuan Liu, Jia-Chen Gu, Yunzhi Yao, Hong Wang, and Nanyun Peng. Energy-Regularized Sequential Model Editing on Hyperspheres. In *International Conference on Learning Representations*, 2026.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. *arXiv preprint arXiv:2405.16821*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023a. URL https://arxiv.org/abs/2202.05262.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer, 2023b. URL https://arxiv.org/abs/2210.07229.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. *CoRR*, abs/2110.11309, 2021. URL https://arxiv.org/abs/2110.11309.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250/.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL https://aclanthology.org/2020.emnlp-main.437/.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170/.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Tom Hartvigsen. WikiBigEdit: Understanding the Limits of Lifelong Knowledge Editing in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL https://arxiv.org/abs/1804.07461.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Ke Wang, Yiming Qin, Nikolaos Dimitriadis, Alessandro Favero, and Pascal Frossard. Memoir: Lifelong model editing with minimal overwrite and informed retention for llms, 2025a. URL https://arxiv.org/abs/2506.07899.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models, 2024a. URL https://arxiv.org/abs/2405.14768.

Peng Wang, Biyu Zhou, Xuehai Tang, Jizhong Han, and Songlin Hu. Lyaplock: Bounded knowledge preservation in sequential large language model editing, 2025b. URL https://arxiv.org/abs/2505.15702.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey, 2024b. URL https://arxiv.org/abs/2310.16218.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019. doi: 10.1162/tacl_a_00290. URL https://aclanthology.org/Q19-1040/.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101/.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.632. URL https://aclanthology.org/2023.emnlp-main.632/.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora, 2023. URL https://arxiv.org/abs/2312.11795.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and

Huajun Chen. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

Taolin Zhang, Haidong Kang, Dongyang Li, Qizhou Chen, Chengyu Wang Xiaofeng He, and Richang Hong. Queueedit: Structural self-correction for sequential model editing in llms, 2025. URL https://arxiv.org/abs/2506.17864.

Xiao Zhang and Ji Wu. Dissecting learning and forgetting in language model finetuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tmsqb6WpLz.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.971. URL https://aclanthology.org/2023.emnlp-main.971/.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. Modifying memories in transformer models. *CoRR*, abs/2012.00363, 2020. URL https://arxiv.org/abs/2012.00363.

A Experimental Setup and Implement Details

In this section, we describe the datasets and evaluation metrics used in our experiments, and detail the base-model capability evaluation, experimental setup, and baseline methods.

A.1 Datasets

- **CounterFact.** CounterFact [Meng et al., 2023a] is a benchmark designed specifically for *factual knowledge editing*. Each example specifies a factual triple in the form of a *subject* and *relation* (e.g., "*Paris* is the capital of France") together with a *counterfactual target object* to be written into the model (e.g., replacing France with a new object). The benchmark provides multiple prompt variants to query the edited knowledge (e.g., a direct rewrite prompt and paraphrases), as well as semantically related "neighborhood" prompts used to check locality/specificity. It is therefore commonly used to measure whether an editor succeeds on the intended rewrite while preserving behavior on nearby but non-target facts.
- **ZsRE.** ZsRE [Levy et al., 2017] is a widely used factual probing dataset derived from relation extraction in a *question-answering* format. Each instance corresponds to a (subject, relation) query rendered as a natural-language question (often with paraphrased variants), and the model is expected to generate the correct answer entity. In model editing, ZsRE is typically used to evaluate whether an edit can reliably update the answer to a targeted question while generalizing across question paraphrases, making it a complementary benchmark to CounterFact for assessing efficacy and generalization under diverse query formulations.
- **WikiBigEdit.** WikiBigEdit [Thede et al., 2025] is a large-scale *lifelong* knowledge-editing benchmark constructed from real-world *Wikidata snapshot diffs*, yielding a chronological stream of factual changes with over 500K question–answer pairs. Each example is grounded in a Wikidata (subject, relation, object) triplet and provides an *update* question plus rephrases/persona variants to test generalization; it further includes dedicated *locality* prompts (with expected answers) and optional *multi-hop* questions to probe whether edits preserve unrelated knowledge and reasoning behavior.

A.2 Metrics

We report standard knowledge-editing metrics on ZsRE and CounterFact, and additionally introduce a long-horizon stability indicator based on the CounterFact score. We also report WikiBigEdit metrics following the official benchmark protocol, namely *Edit Success* (ES), *Generalization Success* (GS), and *Locality Success* (LS). For each edit instance i, let f_θ denote the (edited) language model, (s_i, r_i) the edit prompt constructed from the subject–relation pair, and o_i the desired post-edit target output. We use $N((s_i, r_i))$ to denote a set of paraphrased (semantically equivalent) prompts, and $O((s_i, r_i))$ to denote a set of locality prompts used to probe non-target behavior. In sequential-editing experiments, metrics are evaluated at a set of discrete checkpoints (shared across all methods for a fixed base model), which enables us to define the *collapse point* (CP) as the first checkpoint where the CounterFact score falls below a predefined threshold. Our definitions largely follow common practice in prior work (e.g., [Meng et al., 2023a,b, Zhang et al., 2025, Fang et al., 2024]), but we restate them here for completeness.

A.2.1 ZsRE Metrics

Efficacy. On ZsRE, efficacy is measured as the average top-1 accuracy on the edited questions:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i)) \right\}. \tag{17}$$

Generalization. Generalization evaluates whether the edit transfers to paraphrased queries. We compute the average top-1 accuracy over the rephrased prompt set $N((s_i, r_i))$:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid N((s_i, r_i))) \right\}. \tag{18}$$

Specificity (Locality). Specificity measures whether predictions on locality prompts remain correct with respect to their dataset-provided locality answers o_i^c. We evaluate this by checking whether the model's top-1 prediction on $O((s_i, r_i))$ matches the (original) pre-edit answer o_i^c:

$$\mathbb{E}_i \left\{ o_i^c = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i))) \right\}. \tag{19}$$

A.2.2 CounterFact Metrics

On COUNTERFACT, the evaluation is likelihood-based. Let o_c^i be the model's original (pre-edit) output for the edit prompt, and o_i the desired new object to be written. We then define:

Efficacy (Rewrite Success). We count an edit as successful on the rewrite prompt if the new target is assigned higher probability than the original output:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid (s_i, r_i)] > \mathbb{P}_{f_\theta}\left[o_c^i \mid (s_i, r_i)\right] \right]. \tag{20}$$

Generalization (Paraphrase Success). We analogously evaluate paraphrase generalization on the rephrased prompt set $N((s_i, r_i))$:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid N((s_i, r_i))] > \mathbb{P}_{f_\theta}\left[o_c^i \mid N((s_i, r_i))\right] \right]. \tag{21}$$

Specificity (Neighborhood/Locality Success). Specificity is computed on neighborhood prompts $O((s_i, r_i))$ (distinct but semantically related subjects) by requiring the edited model to favor the correct/original completion over the edited target:

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i \mid O((s_i, r_i))] < \mathbb{P}_{f_\theta}\left[o_c^i \mid O((s_i, r_i))\right] \right]. \tag{22}$$

Score (Harmonic Mean). In addition to reporting *Efficacy*, *Generalization*, and *Specificity* separately, we also summarize CounterFact performance with a single *overall score* defined as their harmonic mean:

$$\text{Score} \;\coloneqq\; \text{H}(\text{Eff.}, \text{Gen.}, \text{Spe.}) \;=\; \frac{3}{\frac{1}{\text{Eff.}} + \frac{1}{\text{Gen.}} + \frac{1}{\text{Spe.}}}. \tag{23}$$

Here, Eff., Gen., and Spe. are the dataset-level success rates defined above (Eqs. 20–22).

Collapse Point (CP@60). Let $\mathcal{T}_m = \{t_1, t_2, \ldots\}$ be the evaluation checkpoints for base model m, and let $\text{Score}(t)$ denote the CounterFact score at checkpoint t. We define the collapse point as the earliest checkpoint where the score falls below a threshold:

$$\text{CP@60} \;\coloneqq\; \min\{\, t \in \mathcal{T}_m \;:\; \text{Score}(t) \le 60 \}. \tag{24}$$

Due to discrete evaluation, the true collapse (if any) occurs within the interval $(t^-, \text{CP@60}]$, where t^- is the immediate predecessor of CP@60 in $\mathcal{T}_m$.

Fluency (Generation Entropy). To detect degenerate generations with excessive repetition, we compute an n-gram entropy score:

$$-\frac{2}{3}\sum_k g_2(k)\log_2 g_2(k) \;-\; \frac{4}{3}\sum_k g_3(k)\log_2 g_3(k), \tag{25}$$

where $g_n(\cdot)$ denotes the empirical frequency distribution of n-grams in the model output.

Consistency (Reference Score). Consistency evaluates whether the model's generated description remains aligned with a reference text. Given a subject s, we compute the cosine similarity between the TF-IDF vector of the model-generated text and that of a reference Wikipedia passage about the corresponding object.

A.2.3 WikiBigEdit Metrics

On WIKIBIGEDIT, the evaluation is accuracy-based on question–answer prompts. Let (s_i, r_i) denote the update query for edit instance i, and o_i the desired post-edit target output. We use $N((s_i, r_i))$ to denote a set of generalization queries (e.g., rephrases/persona variants), and $O((s_i, r_i))$ to denote a set of locality probes paired with edit i, whose answers should remain unchanged (denoted by o_i^c).

ES (Edit Success). We measure edit success as the average top-1 accuracy on the update queries:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i)) \right\}. \tag{26}$$

GS (Generalization Success). Generalization evaluates whether the edit transfers to the associated generalization set $N((s_i, r_i))$:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid N((s_i, r_i))) \right\}. \tag{27}$$

LS (Locality Success). Locality evaluates whether predictions on paired locality probes $O((s_i, r_i))$ match their original (pre-edit) answers o_i^c:

$$\mathbb{E}_i \left\{ o_i^c = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i))) \right\}. \tag{28}$$

A.3 Implementation details

Table 5: Ablation on hyperparameter N. We report runtime (seconds) and the raw-mean estimate of $\|v^{\text{new}}\|$.

Model	N	**Time (s)**	**Raw** $\|v^{\text{new}}\|$
Llama3-8B	100	47.15 ± 5.68	5.298 ± 0.040
	300	143.33 ± 1.95	5.309 ± 0.019
	500	239.13 ± 4.05	5.320 ± 0.020
	1000	469.34 ± 6.81	5.325 ± 0.015
	2000	936.79 ± 8.51	5.322 ± 0.007
GPT-J-6B	100	13.07 ± 1.57	63.248 ± 1.102
	300	41.43 ± 1.37	62.872 ± 0.634
	500	67.84 ± 2.54	62.946 ± 0.561
	1000	135.06 ± 1.78	63.075 ± 0.412
	2000	268.83 ± 2.99	63.038 ± 0.179

Estimating the anchor norm. NAS requires a single scalar anchor norm $\tau := \left(\mathbb{E}\|v^{\text{new}}\|^2\right)^{1/2}$ measured on an original (unedited) model. In all experiments, we estimate this expectation by performing pilot edits on $N = 1000$ randomly sampled facts and computing the empirical mean of the resulting $\|v^{\text{new}}\|^2$ values. We use the same editing site and the same base editor configuration as in the main sequential-editing runs.

Integration with base editors. NAS is implemented as a lightweight post-processing step on the target value vector produced by the base editor. When inserting NAS into an editor, we only introduce two NAS-specific hyperparameters: (i) the pilot sample size N (fixed to 1000 throughout this paper), and (ii) a boolean flag `use_nas` indicating whether to enable the rescaling rule. All other hyperparameters and logic—including the computation of $\tilde{v}_n^{\text{new}}$, optimization settings, and update rules—are inherited *unchanged* from the attached editor. Unless otherwise stated, when we refer to "NAS" as an independent editing method, it denotes **AlphaEdit** equipped with NAS (i.e., AlphaEdit + NAS).

Sensitivity to N. We ablate the hyperparameter N over $\{100, 300, 500, 1000, 2000\}$ with 5 restarts. Across both Llama3-8B and GPT-J-6B, the raw-mean estimate of $\|v^{\text{new}}\|$ remains highly stable as N increases (e.g., Llama3-8B varies only from 5.298 at N=100 to 5.322 at N=2000, and GPT-J-6B stays within 62.87–63.25), with variability across restarts generally shrinking for larger N. In contrast, the runtime increases approximately linearly with N (e.g., on Llama3-8B from 47.15s at N=100 to 936.79s at N=2000), indicating that larger N primarily trades compute for marginally reduced estimator variance rather than changing the central tendency.

Compute resources. All experiments were conducted on a single NVIDIA H100 GPU. For in-weight Locate-and-Edit (L&E) editors (e.g., ROME/MEMIT/PRUNE/RECT/AlphaEdit), running sequential editing on our primary base models typically requires at least 48 GB of GPU memory due to the model size and the per-edit optimization/update procedure. Importantly, enabling NAS does not introduce additional GPU-memory overhead.

A.4 Baselines

All baselines are evaluated using the **official implementations and recommended hyperparameters released by the authors**. The only exception is PRUNE, for which we use the implementation provided in the **AlphaEdit** codebase [Fang et al., 2024].

- **FT (Fine-Tuning). [Zhu et al., 2020]** A straightforward baseline that directly fine-tunes the model on the edited fact via gradient-based optimization. While often effective at rewriting the target behavior, standard fine-tuning can introduce broader parameter changes and thus may cause unintended side effects on unrelated knowledge.
- **MEMIT. [Meng et al., 2023b]** A scalable in-weight editing method designed for efficiently injecting many factual updates. It generalizes single-edit rank-constrained updates to multi-edit settings by composing a structured weight update (often distributed across layers), enabling large batches or long streams of edits with improved efficiency.
- **PRUNE. [Ma et al., 2024]** A regularization-based editor that constrains edit-induced perturbations to preserve model fidelity under sequential updates. It controls the geometry (e.g., update conditioning) of the applied change to suppress overly aggressive directions, thereby mitigating degradation and side effects over long edit horizons.
- **RECT. [Gu et al., 2024]** An editing method that restricts the relative change of weights during an update, typically by sparsifying or truncating the weight delta so that only the most relevant subset of parameters is modified. This strategy retains the base model's general capabilities while still achieving high edit efficacy on the target knowledge.
- **AlphaEdit. [Fang et al., 2024]** A constrained in-weight editor that shapes the update to reduce interference with pre-existing knowledge. It enforces an approximate invariance constraint (e.g., a null-space projection) so that the update primarily affects the target association while minimizing unintended changes to other facts.
- **UltraEdit. [Gu et al., 2025]** A training- and memory-free editing method that computes the weight update in a single step from a hidden state and its gradient, without any additional fine-tuning. UltraEdit also employs a continual feature normalization strategy to adapt to distribution shifts across sequential edits, enabling ultra-fast and scalable knowledge updates with minimal interference.
- **RLEdit. [Li et al., 2025]** A reinforcement learning-based editor that formulates sequential model editing as a decision process for a hypernetwork. By treating editing success as a reward and optimizing the hypernetwork over entire edit sequences, RLEdit adapts to dynamic model changes and achieves superior editing efficacy and efficiency in lifelong settings.
- **GRACE. [Hartvigsen et al., 2023]** A lifelong editing approach that stores new knowledge in auxiliary key–value style adapters rather than modifying backbone parameters. Each edit is written into a dedicated latent adapter, and a retrieval mechanism activates the corresponding adapter for relevant inputs, supporting a large number of edits with reduced interference on original knowledge.
- **MELO. [Yu et al., 2023]** A plug-in editor that realizes edits via dynamically activated low-rank adapters (e.g., LoRA-style modules). New facts are stored in separate adapter modules that are selectively invoked for matching inputs, providing modularity and strong isolation of edits without changing the core model weights.
- **WISE. [Wang et al., 2024a]** A dual-memory approach that separates original knowledge retention from edited knowledge storage. It writes updates into an auxiliary "side" memory and uses a routing mechanism to combine or choose between the base model and the edited memory at inference time, aiming to reduce interference and improve stability under continual updates.

- **LyapLock. [Wang et al., 2025b]** A sequential L&E wrapper that formulates editing as a long-term constrained optimization problem and uses Lyapunov optimization (virtual queues) to enforce cumulative knowledge-preservation constraints while remaining compatible with existing editors.
- **ENCORE. [Gupta et al., 2025b]** A regularization-oriented L&E extension that combines Most-Probable Early Stopping (MPES) with Frobenius-norm regularization on MEMIT-style closed-form updates to mitigate over-optimization and norm growth in long edit streams.
- **MEMOIR. [Wang et al., 2025a]** A backbone-frozen lifelong editor that writes edits into a residual memory module via sparse, sample-dependent masked updates (TopHash) and selectively retrieves relevant memory at inference to reduce interference.

A.5 General Capability Evaluation

Beyond edit-centric metrics, we also monitor whether long streams of edits erode the base model's *general-purpose* language understanding and reasoning ability. This type of collateral degradation under (lifelong) sequential editing has been repeatedly observed in recent evaluations of model editing at scale, motivating routine reporting of downstream benchmark performance alongside editing success [Fang et al., 2024, Zhang et al., 2025]. Following this common practice, we evaluate the edited model on five representative tasks from GLUE [Wang et al., 2019] and on MMLU [Hendrycks et al., 2021] at a set of checkpoints during the edit stream.

- **SST-2 (SST).** A sentence-level sentiment classification task derived from movie reviews. The model predicts whether a review expresses *positive* or *negative* sentiment, probing basic lexical and compositional understanding as well as robustness to stylistic variation.
- **MRPC.** The Microsoft Research Paraphrase Corpus is a sentence-pair task that asks whether two sentences are paraphrases (i.e., semantically equivalent). It serves as a compact probe of semantic similarity judgments and sensitivity to small meaning changes.
- **RTE.** Recognizing Textual Entailment is a binary natural language inference (NLI) task. Given a premise and a hypothesis, the model predicts whether the hypothesis is entailed by the premise, providing a probe of logical/semantic inference under limited supervision.
- **CoLA.** The Corpus of Linguistic Acceptability asks whether a sentence is grammatically acceptable. While GLUE commonly reports MCC for CoLA, we compute F1 on the acceptability label to keep a unified scale across tasks; this benchmark primarily serves as a probe of syntactic/grammatical sensitivity.
- **MNLI.** Multi-Genre Natural Language Inference is a three-way NLI classification task (entailment/neutral/contradiction) spanning diverse text domains. It is a stronger, broader probe of inference capability than RTE and is commonly used as a barometer of general sentence-pair understanding.
- **MMLU.** Measuring Massive Multitask Language Understanding is a multiple-choice benchmark covering a broad set of academic and professional subjects. Each example presents a question with a fixed set of answer options, testing both factual knowledge and multi-step reasoning in a standardized format.

B Proofs and Derivations

B.1 Definitions and Basic Identities

Our analysis studies the dynamical evolution of $\|v_n^{\mathrm{old}}\|^2$, $\|v_n^{\mathrm{new}}\|^2$, and $\|W_n\|_F^2$ under sequential edits. We first recall the definition of the Frobenius norm and inner product, together with several identities that will be used repeatedly.

Definition B.1 (Frobenius norm)**.** For any real matrix $A \in \mathbb{R}^{m\times n}$ with entries $(A)_{ij} = a_{ij}$, the Frobenius norm is

$$\|A\|_F^2 \;\coloneqq\; \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2. \tag{29}$$

Equivalently, $\|A\|_F$ is the Euclidean norm of $\mathrm{vec}(A) \in \mathbb{R}^{mn}$. For convenience, we use $\|A\|^2$ to denote $\|A\|_F^2$ when the meaning is clear.

Definition B.2 (Frobenius inner product)**.** For any $A, B \in \mathbb{R}^{m\times n}$ with entries $(A)_{ij} = a_{ij}$ and $(B)_{ij} = b_{ij}$, the Frobenius inner product is

$$\langle A, B\rangle_F \;\coloneqq\; \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}\, b_{ij}. \tag{30}$$

This coincides with the Euclidean inner product between $\mathrm{vec}(A)$ and $\mathrm{vec}(B)$, and in particular

$$\|A\|_F^2 \;=\; \langle A, A\rangle_F. \tag{31}$$

lemma B.3 (Useful identities)**.** *For any $A, B \in \mathbb{R}^{m\times n}$, $u \in \mathbb{R}^m$, and $v \in \mathbb{R}^n$, the following hold:*

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + 2\langle A, B\rangle_F, \tag{32}$$

$$\|uv^\top\|_F^2 = \|u\|^2\,\|v\|^2, \tag{33}$$

$$\langle A, uv^\top\rangle_F = u^\top A v. \tag{34}$$

B.2 Proof for Lemma 3.1

Recall that under the convention $C = I$, Eq. (10) can be written as Eq. (11):

$$\Delta W_n \;= \frac{\big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big){k_n^\star}^\top}{\|k_n^\star\|^2}. \tag{35}$$

Starting from Eq. (8) and applying the Frobenius identity (32) (Lemma B.3), we have

$$\begin{aligned}\|W_n\|^2 &= \|W_{n-1} + \Delta W_n\|^2 \\ &= \|W_{n-1}\|^2 + \|\Delta W_n\|^2 + 2\langle W_{n-1}, \Delta W_n\rangle_F.\end{aligned} \tag{36}$$

We now expand the last two terms in Eq. (36). First, by Eq. (33) (Lemma B.3) and (11),

$$\begin{aligned}\|\Delta W_n\|^2 &= \left\| \frac{\big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big){k_n^\star}^\top}{\|k_n^\star\|^2} \right\|^2 \\ &= \frac{\|v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\|^2}{\|k_n^\star\|^2}.\end{aligned} \tag{37}$$

Second, by Eq. (34) (Lemma B.3) and (11),

$$\begin{aligned}\langle W_{n-1}, \Delta W_n\rangle_F &= \left\langle W_{n-1}, \frac{\big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big){k_n^\star}^\top}{\|k_n^\star\|^2} \right\rangle_F \\ &= \frac{1}{\|k_n^\star\|^2}\left\langle W_{n-1}, \big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big){k_n^\star}^\top \right\rangle_F \\ &= \frac{1}{\|k_n^\star\|^2}\big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big)^\top W_{n-1} k_n^\star.\end{aligned} \tag{38}$$

Substituting Eqs. (37) and (38) into Eq. (36) yields

$$\|W_n\|^2 = \|W_{n-1}\|^2 + \frac{\|v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\|^2}{\|k_n^\star\|^2} + \frac{2}{\|k_n^\star\|^2}\big(v_n^{\mathrm{new}} - v_n^{\mathrm{old}}\big)^\top W_{n-1} k_n^\star. \tag{39}$$

Recall that $v_n^{\mathrm{old}} = W_{n-1}k_n^\star$, and define $\Delta_n \coloneqq v_n^{\mathrm{new}} - v_n^{\mathrm{old}}$. Then Eq. (39) becomes

$$\|W_n\|^2 = \|W_{n-1}\| + \frac{1}{\|k_n^\star\|^2}\Big(\|\Delta_n\|^2 + 2\langle \Delta_n, v_n^{\mathrm{old}}\rangle\Big). \tag{40}$$

Finally, since $v_n^{\mathrm{new}} = v_n^{\mathrm{old}} + \Delta_n$,

$$\|\Delta_n\|^2 + 2\langle \Delta_n, v_n^{\mathrm{old}}\rangle = \|v_n^{\mathrm{new}}\|^2 - \|v_n^{\mathrm{old}}\|^2. \tag{41}$$

Plugging Eq. (41) into Eq. (40) gives exactly Eq. (12), completing the proof.

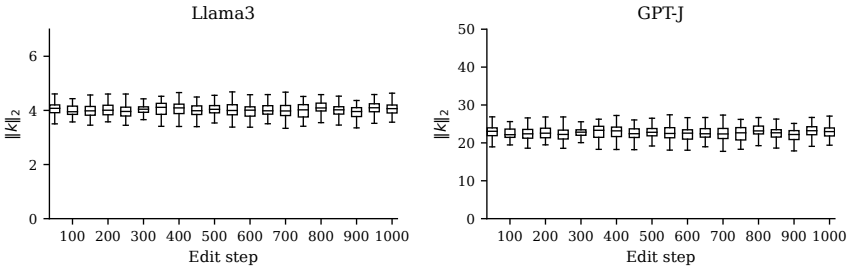


Figure 6: $\|k_n^\star\|^2$ **stability across edit steps.** Boxplots of $\|k_n^\star\|$ over edit steps show small fluctuation and tight concentration, supporting the approximation $\|k_n^\star\|^{-2} \approx K$ used in the analysis.

B.3 Proof for Prop. 3.2

Starting from Lemma 3.1 (Eq. (12)), we take conditional expectation given the current weight W_{n-1}:

$$\mathbb{E}\big[\|W_n\|^2 \,\big|\, W_{n-1}\big] = \|W_{n-1}\|^2 + \mathbb{E}\left[\frac{\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2}{\|k_n^\star\|^2} \,\middle|\, W_{n-1}\right]. \tag{42}$$

Empirically, $\|k_n^\star\|$ varies little across edit steps (Fig. 6). We therefore treat $\|k_n^\star\|^{-2}$ as approximately constant and set

$$K \;:=\; \text{the empirical mean of } \; \|k_n^\star\|^{-2} \text{ over the sampled checkpoints and sampled edits/keys.} \tag{43}$$

Substituting $\|k_n^\star\|^{-2} \approx K$ into (42) yields

$$\mathbb{E}\big[\|W_n\|^2 \,\big|\, W_{n-1}\big] \;\approx\; \|W_{n-1}\|^2 + K \cdot \mathbb{E}\big[\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2 \,\big|\, W_{n-1}\big]. \tag{44}$$

Next, Fig. 7 shows an approximately linear scaling between value-vector norms and the edited weight norm over the observation range:

$$\mathbb{E}\big[\|v_n^{\text{new}}\|^2 \,|\, W_{n-1}\big] \approx s_{\text{new}} \|W_{n-1}\|^2 + b_{\text{new}}, \tag{45}$$

$$\mathbb{E}\big[\|v_n^{\text{old}}\|^2 \,\big|\, W_{n-1}\big] \approx s_{\text{old}} \|W_{n-1}\|^2 + b_{\text{old}}, \tag{46}$$

where $s_{\text{new}} > s_{\text{old}} > 0$.

Subtracting Eq. (46) from Eq. (45) gives

$$\mathbb{E}\big[\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2 \,\big|\, W_{n-1}\big] \approx (s_{\text{new}} - s_{\text{old}}) \|W_{n-1}\|^2 + (b_{\text{new}} - b_{\text{old}}). \tag{47}$$

For notational convenience, we introduce a unified reparametrization of the empirical linear fits by the two constants (ρ, γ):

$$\rho(s_{\text{new}}, s_{\text{old}}; K) := 1 + K(s_{\text{new}} - s_{\text{old}}), \qquad \gamma(s_{\text{new}}, b_{\text{new}}, s_{\text{old}}, b_{\text{old}}) := \frac{b_{\text{new}} - b_{\text{old}}}{s_{\text{new}} - s_{\text{old}}}. \tag{48}$$

Then Eq. (47) can be rewritten as

$$\mathbb{E}[\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2 \mid W_{n-1}] \approx (s_{\text{new}} - s_{\text{old}}) \, (\|W_{n-1}\|^2 + \gamma), \quad \textit{where}\, \gamma = \gamma(s_{\text{new}}, b_{\text{new}}, s_{\text{old}}, b_{\text{old}}) \tag{49}$$

Substituting Eq. (49) into Eq. (44) yields

$$\mathbb{E}[\|W_n\|^2 \mid W_{n-1}] \approx \|W_{n-1}\|^2 + K(s_{\text{new}} - s_{\text{old}}) \, (\|W_{n-1}\|^2 + \gamma) = \rho \, \|W_{n-1}\|^2 + (\rho - 1)\gamma, \quad \textit{where}\, \rho = \rho(s_{\text{new}}, s_{\text{old}}; K). \tag{50}$$

Taking expectation on both sides and using the tower property,

$$\mathbb{E}\|W_n\|^2 \approx \rho \, \mathbb{E}\|W_{n-1}\|^2 + (\rho - 1)\gamma. \tag{51}$$

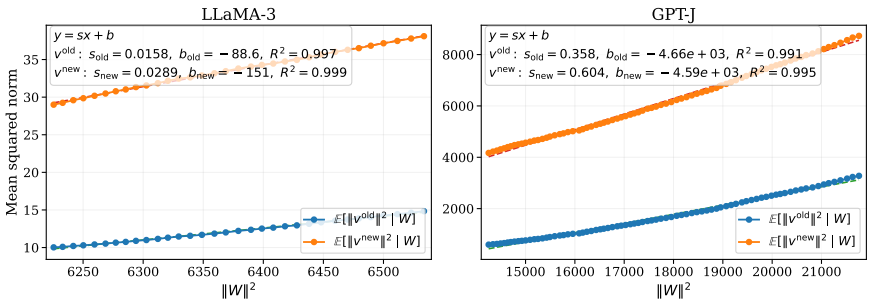


Figure 7: **Linear scaling between value-vector norms and the edited weight norm (baseline).** For sequential editing, we plot $\mathbb{E}\left[\|v^{\text{new}}\|^2 \,\middle|\, W\right]$ and $\mathbb{E}\left[\|v^{\text{old}}\|^2 \,\middle|\, W\right]$ versus $\|W\|^2$ at multiple checkpoints (averaged over sampled edits/keys). Dashed lines are linear fits; insets report fitted slopes and R^2. The fitted slope for v^{new} is consistently larger than that for v^{old}.

Since $K > 0$ and $s_{\text{new}} - s_{\text{old}} > 0$, we have $\rho > 1$. Solving Eq. (51) gives

$$\mathbb{E}\|W_n\|^2 \approx \rho^n \, \mathbb{E}\|W_0\|^2 + \gamma(\rho^n - 1). \tag{52}$$

In the main text, we denote the baseline (divergent) regime by (α, R), which is the specialization of (γ, ρ) under the empirical fits:

$$R := \rho(s_{\text{new}}, s_{\text{old}}; K), \qquad \alpha := \gamma(s_{\text{new}}, b_{\text{new}}, s_{\text{old}}, b_{\text{old}}). \tag{53}$$

Using notations above, Eq. (52) can be rewritten as:

$$\mathbb{E}\|W_n\|^2 \approx R^n \, \mathbb{E}\|W_0\|^2 + \alpha(R^n - 1), \tag{54}$$

which is exactly Eq. (12). In particular, $R > 1$ indicates an approximately exponential growth trend in $\mathbb{E}\|W_n\|^2$ under the baseline empirical fits. We will introduce an alternative parametrization (β, r) for the NAS (stable) regime in the next subsection.

B.4 Proof for Cor. 3.3

We now analyze our method, which introduces a negative feedback by enforcing a constant target norm for the post-edit value vector. Specifically, we impose

$$\|v_n^{\text{new}}\|^2 := \tau^2, \qquad \tau > 0. \tag{55}$$

Equivalently,

$$\mathbb{E}[\|v_n^{\text{new}}\|^2 \mid W_{n-1}] = \tau^2. \tag{56}$$

Using the same linear-fit notation as in Appendix B.3, this corresponds to the specialization

$$s_{\text{new}} = 0, \qquad b_{\text{new}} = \tau^2. \tag{57}$$

Meanwhile, within the same observation range (Fig. 9), we empirically find an approximately stable linear scaling for the pre-edit value norm:

$$\mathbb{E}\|v_n^{\text{old}}\|^2 \approx s_{\text{old}} \, \mathbb{E}\|W_{n-1}\|^2 + b_{\text{old}}. \tag{58}$$

Recall the unified parametrization in Appendix B.3 (Eq. (48)). Under the NAS specialization $s_{\text{new}} = 0$ and $b_{\text{new}} = \tau^2$, we have

$$\rho = \rho(0, s_{\text{old}}; K) = 1 - K s_{\text{old}}, \qquad \gamma = \gamma(0, \tau^2, s_{\text{old}}, b_{\text{old}}) = \frac{\tau^2 - b_{\text{old}}}{0 - s_{\text{old}}}. \tag{59}$$

Following the same derivation as in Appendix B.3, substituting $\|v_n^{\text{new}}\|^2 \mapsto \tau^2$ and using Eq. (58) yields

$$\mathbb{E}\|W_n\|^2 \approx \rho \, \mathbb{E}\|W_{n-1}\|^2 + (\rho - 1)\gamma. \tag{60}$$

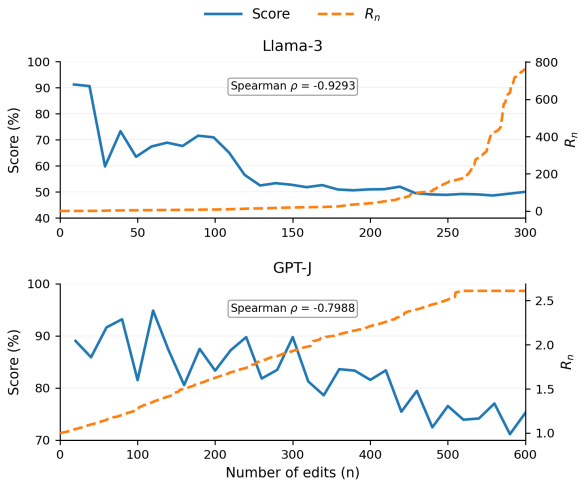


Figure 8: **Norm explosion accompanies sequential editing collapse.** During sequential MEMIT editing, we track edit success (blue, left axis) and normalized weight growth $R_n := \|W_n\|/\|W_0\|$ (orange, right axis) versus edit step n. For both Llama-3 and GPT-J, increasing R_n coincides with deteriorating editing performance; Spearman ρ is shown in each panel.

Solving Eq. (60) gives

$$\mathbb{E}\|W_n\|^2 \approx \rho^n \,\mathbb{E}\|W_0\|^2 + \gamma(\rho^n - 1). \tag{61}$$

In the main text, we reparametrize the NAS (stable) regime by (β, r) to highlight boundedness:

$$r := \rho(0, s_{\text{old}}; K) = 1 - K s_{\text{old}}, \qquad \beta := -\gamma(0, \tau^2, s_{\text{old}}, b_{\text{old}}) = \frac{\tau^2 - b_{\text{old}}}{s_{\text{old}}}. \tag{62}$$

Substituting Eq. (62) into Eq. (61) yields

$$\mathbb{E}\|W_n\|^2 \approx r^n \,\mathbb{E}\|W_0\|^2 + \beta(1 - r^n). \tag{63}$$

which is exactly Eq. (16). In our experiments, the fitted slope satisfies $s_{\text{old}} > 0$ and the estimated product $K s_{\text{old}}$ lies in $(0, 1)$ over the observation range, hence $0 < r = 1 - K s_{\text{old}} < 1$. Moreover, we typically observe $b_{\text{old}} < 0$ and set $\tau > 0$, so $\beta = \frac{\tau^2 - b_{\text{old}}}{s_{\text{old}}} > 0$. Therefore, Eq. (63) suggests that $\mathbb{E}\|W_n\|^2$ remains uniformly bounded over n.

Fixed point and direction of approach. Eq. (60) can be written as

$$\mathbb{E}\|W_n\|^2 - \beta \;\approx\; r\left(\mathbb{E}\|W_{n-1}\|^2 - \beta\right), \tag{64}$$

so β is the unique fixed point and the distance to it shrinks by a factor $r \in (0, 1)$ per edit. The same β follows from setting the expected increment of Lemma 3.1 to zero: $K(\tau^2 - s_{\text{old}}\beta - b_{\text{old}}) = 0$. Because $s_{\text{old}} > 0$,

$$\beta > \mathbb{E}\|W_0\|^2 \iff \tau^2 > s_{\text{old}}\,\mathbb{E}\|W_0\|^2 + b_{\text{old}} \approx \mathbb{E}\big[\|v^{\text{old}}\|^2 \mid W_0\big], \tag{65}$$

i.e., the anchor is larger than the typical pre-edit value norm of the original model. This is what we measure (Figs. 7 and 9), and it is also the reason vanilla editing grows the norm in the first place. Starting from W_0, Eq. (64) then gives a monotone increase toward β from below. If NAS is switched on only after the norm has grown past β, the same recursion gives a monotone decrease toward β. In the opposite regime $\tau^2 \le \mathbb{E}[\|v^{\text{old}}\|^2 \mid W_0]$ the vanilla loop does not start, and NAS has no effect in expectation.

Size of individual writes at the fixed point. At β only the *expected* norm increment is zero. Each write still satisfies

$$\|\Delta W_n\|_F = \frac{\|v_n^{\text{new}} - v_n^{\text{old}}\|}{\|k_n^\star\|} \le \frac{\tau + \|v_n^{\text{old}}\|}{\|k_n^\star\|}, \tag{66}$$

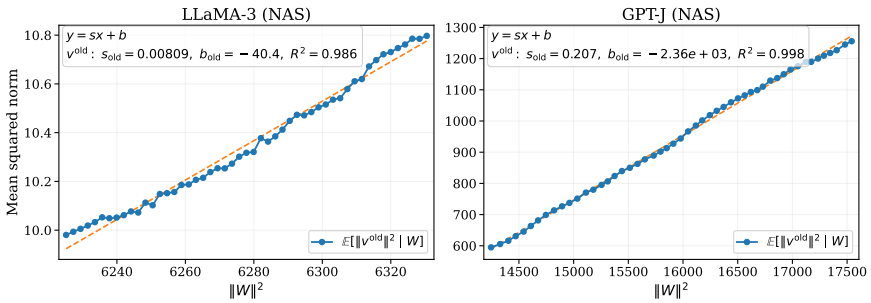


Figure 9: **Linear scaling between the pre-edit value norm and the edited weight norm (NAS).** Under norm anchoring, we plot $\mathbb{E}[\|v_n^{\text{old}}\|^2 \mid W]$ versus $\|W\|^2$ at multiple checkpoints (averaged over sampled edits/keys). Dashed lines are linear fits; insets report fitted slopes and R^2.

which contains no factor that decays with n; the content of the edit is carried by the direction of v_n^{new}. Edits are also purely additive, $W_n = W_0 + \sum_{i \le n} \Delta W_i$, and nothing in the update multiplies W by r. The term $r^n \, \mathbb{E}\|W_0\|^2$ in Eq. (63) only says that the norm level forgets its initial value; it does not mean that the base weights decay.

B.5 General Case for $C \neq I$

Lemma 3.1 (General C Case). For the general C case, Eq. (12) in Lemma 3.1 becomes

$$\boxed{\|\tilde{W}_n\|^2 = \|\tilde{W}_{n-1}\|^2 + \frac{\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2}{\|\tilde{k}_n\|^2}}, \tag{67}$$

where

$$\tilde{W}_n \;:=\; W_n C^{1/2}, \qquad \Delta\tilde{W}_n \;:=\; \Delta W_n C^{1/2}, \qquad \tilde{k}_n \;:=\; C^{-1/2} k_n^\star. \tag{68}$$

Proof: Consider Eq. (10):

$$\Delta W_n = \frac{(v_n^{\text{new}} - v_n^{\text{old}})(C^{-1}k_n^\star)^\top}{{k_n^\star}^\top C^{-1} k_n^\star}. \tag{69}$$

For the denominator, note that

$${k_n^\star}^\top C^{-1} k_n^\star = (C^{-1/2}k_n^\star)^\top (C^{-1/2}k_n^\star) = \tilde{k}_n^\top \tilde{k}_n = \|\tilde{k}_n\|^2. \tag{70}$$

For the numerator, observe that

$$(C^{-1}k_n^\star)^\top = \big(C^{-1/2}(C^{-1/2}k_n^\star)\big)^\top = (C^{-1/2}\tilde{k}_n)^\top = \tilde{k}_n^\top (C^{-1/2})^\top = \tilde{k}_n^\top C^{-1/2}. \tag{71}$$

Therefore,

$$\Delta W_n = \frac{\big(v_n^{\text{new}} - v_n^{\text{old}}\big)\, \tilde{k}_n^\top C^{-1/2}}{\left\|\tilde{k}_n\right\|^2}. \tag{72}$$

Since $W_n = W_{n-1} + \Delta W_n$, multiplying both sides by $C^{1/2}$ gives

$$\tilde{W}_n = \tilde{W}_{n-1} + \Delta\tilde{W}_n, \tag{73}$$

where

$$\Delta\tilde{W}_n = \Delta W_n C^{1/2} = \frac{\big(v_n^{\text{new}} - v_n^{\text{old}}\big)\, \tilde{k}_n^\top}{\left\|\tilde{k}_n\right\|^2}. \tag{74}$$

Eqs. (73) and (74) have the same form as Eqs. (8) and (11) under $C = I$. Thus, by replacing $(W_n, k_n^\star)$ with $(\tilde{W}_n, \tilde{k}_n)$ in Appendix B.2, we obtain

$$\|\tilde{W}_n\|^2 = \|\tilde{W}_{n-1}\|^2 + \frac{\|v_n^{\text{new}}\|^2 - \|v_n^{\text{old}}\|^2}{\|\tilde{k}_n\|^2}, \tag{75}$$

which completes the proof.

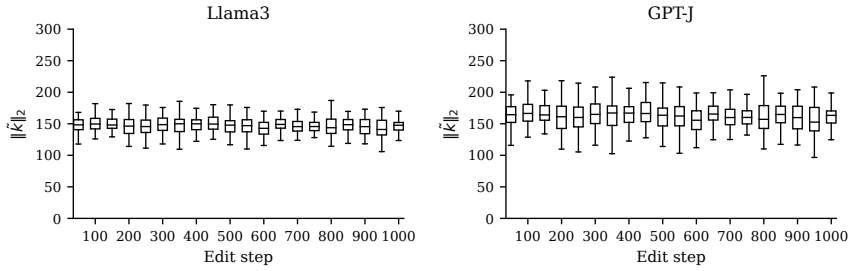


Figure 10: $\|\tilde{k}_n\|^2$ **stability across edit steps.** Boxplots of $\|\tilde{k}_n\|$ over edit steps show small fluctuation and tight concentration, supporting the approximation $\|\tilde{k}_n\|^{-2} \approx \tilde{K}$ used in the analysis.

Propositon 3.2 (General C Case). For general C case, Eq. (3.2) in Prop. 3.2 becomes

$$\boxed{\mathbb{E}\|\tilde{W}_n\|^2 \approx R^n\,\mathbb{E}\|\tilde{W}_0\|^2 + \alpha(R^n - 1)}, \tag{76}$$

where $R > 1$. The derivation is identical to Appendix B.3, by replacing $W_n, k_n^\star$ with the "tilde" versions defined in Eq. (68).

Proof: Recall that the proof of Prop.3.2 directly begins with Eq. (12), which has the same form as Eq. (67) above. Therefore we can very well reuse the derivation in Prop.3.2 line-by-line. The only caveat here is whether the empirical observations in Fig. 6, Fig. 7, still hold under the "tilde" transformation. Concretely, the two approximations used in Appendix B.3 should be replaced by:

1. $\|\tilde{k}_n\|^2$ **Stability (Fig. 10).** Replace
$$\|k_n^\star\|^{-2} \approx K$$
by
$$\|\tilde{k}_n\|^{-2} \approx \tilde{K},$$
i.e., $\|\tilde{k}_n\|^{-2}$ remains stable around a constant during sequential edits.
2. **Linear fits (Fig. 11).** Replace
$$\mathbb{E}\big[\|v_n^{\text{new}}\|^2 \,\big|\, \|W_{n-1}\|^2\big] \approx s_{\text{new}}\|W_{n-1}\|^2 + b_{\text{new}}, \qquad \mathbb{E}\big[\|v_n^{\text{old}}\|^2 \,\big|\, \|W_{n-1}\|^2\big] \approx s_{\text{old}}\|W_{n-1}\|^2 + b_{\text{old}}$$
by the corresponding relations for the "tilde" versions:
$$\mathbb{E}\Big[\|v_n^{\text{new}}\|^2 \,\Big|\, \|\tilde{W}_{n-1}\|^2\Big] \approx s_{\text{new}}\|\tilde{W}_{n-1}\|^2 + b_{\text{new}}, \qquad \mathbb{E}\Big[\|v_n^{\text{old}}\|^2 \,\Big|\, \|\tilde{W}_{n-1}\|^2\Big] \approx s_{\text{old}}\|\tilde{W}_{n-1}\|^2 + b_{\text{old}}.$$

Under these empirical observations for the "tilde" versions, we can define (α, R) (resp. (β, r)) for the general C case. And the derivation in Appendix B.3 can be reused line-by-line, yielding Eq. (76).

Corollary 3.3 (General C Case). For general C case, Eq. (3.3) in Prop. 3.3 becomes

$$\boxed{\mathbb{E}\|\tilde{W}_n\|^2 \approx r^n\,\mathbb{E}\|\tilde{W}_0\|^2 + \beta(1 - r^n)}, \tag{77}$$

where $\beta > 0, \quad r \in (0, 1)$.

Proof: Similar to **Proposition 3.2 (General C Case)**, to reuse the derivation in Appendix B.4, it suffices to verify that the after-NAS empirical fit used in Appendix B.4 remains valid under the transformation in Eq. (68). Specifically, the approximately stable relation in the observation range,

$$\mathbb{E}\|v_n^{\text{old}}\|^2 \approx s_{\text{old}}\,\mathbb{E}\|W_{n-1}\|^2 + b_{\text{old}},$$

should be replaced in the general C case by: (Fig. 12)

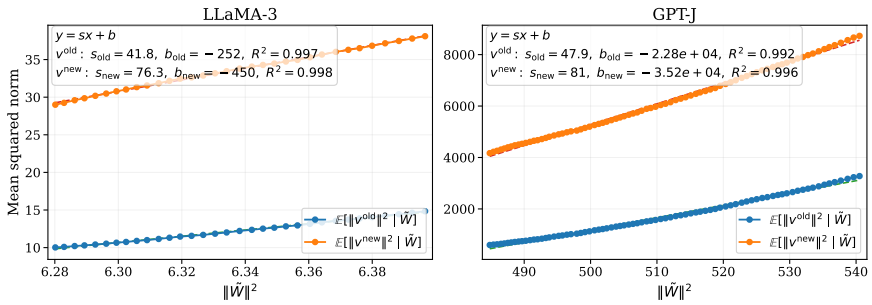


Figure 11: **Linear scaling between value-vector norms and the edited weight norm (baseline, tilde space).** For sequential editing, we plot $\mathbb{E}\left[\|v^{\text{new}}\|^2 \,\middle|\, \tilde{W}\right]$ and $\mathbb{E}\left[\|v^{\text{old}}\|^2 \,\middle|\, \tilde{W}\right]$ versus $\|\tilde{W}\|^2$ at multiple checkpoints (averaged over sampled edits/keys). Dashed lines are linear fits; insets report fitted slopes and R^2. The fitted slope for v^{new} is consistently larger than that for v^{old}.

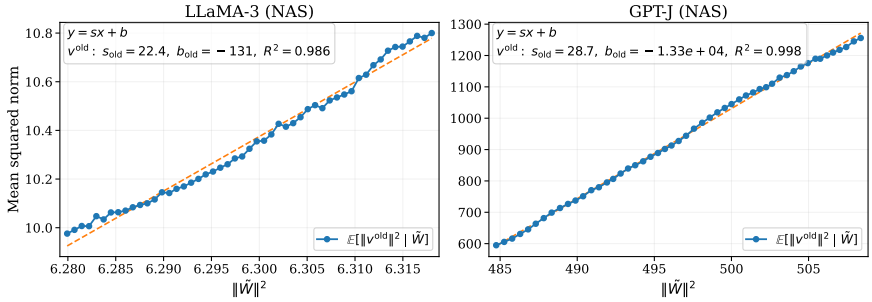


Figure 12: **Linear scaling between the pre-edit value norm and the edited weight norm (NAS, tilde space).** Under norm anchoring, we plot $\mathbb{E}[\|v_n^{\text{old}}\|^2 \mid \tilde{W}]$ versus $\|\tilde{W}\|^2$ at multiple checkpoints (averaged over sampled edits/keys). Dashed lines are linear fits; insets report fitted slopes and R^2.

$$\mathbb{E}\|v_n^{\text{old}}\|^2 \approx s_{\text{old}}\,\mathbb{E}\|\tilde{W}_{n-1}\|^2 + b_{\text{old}}.$$

Since the above "tilde" empirical phenomenon also holds, the remaining computations in Appendix B.4 can be reused line-by-line, yielding the corresponding conclusion of Cor. 3.3 for the general C case.

B.6 Per-Edit Interference under NAS

Cor. 3.3 bounds the expected weight norm. Here we turn it into a bound on how far a single write can move the edited layer on typical inputs. We use the general rank-one write of Eq. (10) with $C \succ 0$ and the notation $\tilde{k}_n = C^{-1/2}k_n^\star$ of Appendix B.5.

Corollary B.4 (Bounded per-edit interference)**.** *Let $C \succ 0$ and let NAS be applied at every step, so that $\|v_n^{\text{new}}\| = \tau$. Then*

$$\mathbb{E}_{k\sim\mathcal{D}}\|\Delta W_n k\|^2 = \|\Delta W_n C^{1/2}\|_F^2 = \frac{\|v_n^{\text{new}} - v_n^{\text{old}}\|^2}{\|\tilde{k}_n\|^2} \le \frac{\left(\tau + \|v_n^{\text{old}}\|\right)^2}{\|\tilde{k}_n\|^2}. \tag{78}$$

If, in addition, $\|\tilde{k}_n\|^{-2} \approx \tilde{K}$ (Assumption A1 in the transformed space, Appendix B.5) and the v^{old} relation of Assumption A2 holds, then

$$\mathbb{E}\,\|\Delta W_n C^{1/2}\|_F^2 \;\lesssim\; 2\tilde{K}\Big(\tau^2 + s_{\text{old}}\max\big(\mathbb{E}\|W_0\|^2, \beta\big) + b_{\text{old}}\Big) \quad \textit{for all } n. \tag{79}$$

Proof. Since $C = \mathbb{E}_{k\sim\mathcal{D}}[kk^\top]$, $\mathbb{E}_k\|\Delta W_n k\|^2 = \operatorname{Tr}(\Delta W_n C\,\Delta W_n^\top) = \|\Delta W_n C^{1/2}\|_F^2$. From Eq. (10), $\Delta W_n C^{1/2} = (v_n^{\text{new}} - v_n^{\text{old}})\,u_n^\top$ with $u_n = C^{1/2}C^{-1}k_n^\star/(k_n^{\star\top}C^{-1}k_n^\star) = \tilde{k}_n/\|\tilde{k}_n\|^2$. A rank-one matrix $ab^\top$ has $\|ab^\top\|_F = \|a\|\,\|b\|$, and $\|u_n\| = 1/\|\tilde{k}_n\|$, which gives the equality in Eq. (78). The inequality is the triangle inequality with $\|v_n^{\text{new}}\| = \tau$. For Eq. (79), use $(a+b)^2 \le 2a^2 + 2b^2$ and $\mathbb{E}\|v_n^{\text{old}}\|^2 \approx s_{\text{old}}\mathbb{E}\|W_{n-1}\|^2 + b_{\text{old}}$. By Eq. (64), $\mathbb{E}\|W_{n-1}\|^2$ lies between $\mathbb{E}\|W_0\|^2$ and β for every n, so it is at most $\max(\mathbb{E}\|W_0\|^2, \beta)$. □

Table 6: Independent single-edit results on 1,000 CounterFact examples with Llama-3-8B. Each edit starts from the original base model and is evaluated in isolation. NAS does not systematically reduce single-edit efficacy.

Editor	NAS	Rewrite Succ.	Paraphrase Succ.	Locality Succ.	Score
MEMIT	Off	98.20	80.30	88.68	88.46
MEMIT	On	98.20	82.10	88.62	89.16
AlphaEdit	Off	98.00	81.00	86.74	88.03
AlphaEdit	On	99.00	83.90	86.56	89.36

Effect on an earlier edit. For $C = I$, the write at step n moves the output of an earlier key $k_i^\star$ by

$$\Delta W_n k_i^\star = \big(v_n^{\text{new}} - v_n^{\text{old}}\big) \frac{\langle k_n^\star, k_i^\star \rangle}{\|k_n^\star\|^2}, \qquad \|\Delta W_n k_i^\star\| \le \big(\tau + \|v_n^{\text{old}}\|\big) \frac{|\langle k_n^\star, k_i^\star \rangle|}{\|k_n^\star\|^2}. \tag{80}$$

The scale of this disturbance is set by the write signal. Under NAS it stays of order τ for the whole stream; without the anchor it grows with the norm-feedback loop.

Scope. Eqs. (78)–(80) bound how far one write moves the layer. They are not a theorem about downstream accuracy or about retention of earlier edits; retention is measured in Appendix C.14. In particular, NAS adds no mechanism against the gradual rank-one interference between edits in Eq. (80). Editors that protect earlier edits explicitly, such as the accumulated previous-edit term in AlphaEdit, keep that protection when NAS is added, because NAS only changes the norm of the solved value.

B.7 Multi-Layer Editors

MEMIT-style editors solve one target value at the last edited layer and spread the residual over a set of layers $\mathcal{L}$. NAS rescales the solved value before this allocation. Given the layerwise allocation, the write at each layer $l \in \mathcal{L}$ has the same rank-one form as Eq. (10), with that layer's share of the residual in place of $v_n^{\text{new}} - v_n^{\text{old}}$. Lemma 3.1 and Cor. B.4 therefore apply to each layer separately, with layer-specific constants. This is a conditional statement. A write at layer l also changes the keys seen by layers after l, and our results do not cover this cross-layer coupling. Empirically, the weight-norm ratio of every edited layer stays within $[1.006, 1.021]$ after 5,000 AlphaEdit+NAS edits on Llama3-8B (Table 17).

C Additional Experiment

C.1 Single-Edit Plasticity under Norm Anchoring

RQ4 examines whether the long-horizon gains of NAS are obtained by weakening individual edits. To isolate this question from sequential accumulation effects, we evaluate independent single edits: each edit starts from the original base model, applies one edit, and is then evaluated in isolation. We use 1,000 CounterFact examples on Llama-3-8B and compare each base editor with and without NAS under the same editing configuration.

Table 6 reports the full single-edit results. NAS does not systematically reduce single-edit efficacy. For MEMIT, the overall score increases from 88.46 to 89.16; for AlphaEdit, it increases from 88.03 to 89.36. The locality metric changes only marginally, suggesting that anchoring the final solved value vector does not introduce a broad degradation of edit locality in the single-edit setting.

We further test a harder subset where NAS applies the strongest magnitude correction. Specifically, for each base editor, we select the top 25% examples whose solved value vectors have the largest norms relative to the NAS anchor, i.e., the cases that receive the strongest downward rescaling under NAS. This subset directly probes whether norm anchoring sacrifices edit plasticity when the intervention is most active.

Table 7 shows that the effect remains limited even in this hard subset. For AlphaEdit, NAS is neutral to slightly beneficial, increasing the score from 91.03 to 91.27. For MEMIT, NAS incurs a modest score decrease from 92.98 to 91.81, mainly through paraphrase success, while rewrite success remains

very high and locality slightly improves. Thus, any single-edit plasticity cost is small and localized, and cannot explain the large long-horizon gains observed in the sequential setting.

Table 7: Single-edit results on the top 25% most strongly downscaled CounterFact examples. These are the cases whose solved value vectors have the largest norms relative to the NAS anchor. Even in this hard subset, NAS does not show a broad collapse of single-edit efficacy.

Editor	NAS	Eff.	Gen.	Loc.	Score
MEMIT	Off	100.00	90.80	88.88	92.98
MEMIT	On	99.20	88.00	89.04	91.81
AlphaEdit	Off	99.20	88.40	86.48	91.03
AlphaEdit	On	99.20	88.40	87.12	91.27

Overall, these single-edit analyses support the interpretation in RQ4: NAS's long-horizon improvements are not primarily driven by a broad stability–plasticity trade-off. Instead, anchoring the final solved value vector can improve sequential stability while leaving standard single-edit behavior largely intact.

C.2 Robustness to Anchor Mis-Specification

NAS uses an original-model reference norm as the anchor. A natural question is whether the method depends on a narrowly tuned anchor value. To test this, we intentionally bias the anchor estimate by selecting pilot subsets with lower or higher solved-value norms, and then run 4,000 sequential CounterFact [Meng et al., 2023a] edits under the same configuration.

Table 8: Robustness to anchor mis-specification. The target norm is intentionally estimated from lower- or higher-norm pilot subsets.

Anchor source	Target norm	Score@4000
Bottom 10%	4.6434	83.79
Bottom 25%	4.8226	83.88
Default	5.3161	83.44
Top 25%	5.7927	82.98
Top 10%	5.9504	83.07

The anchor varies from 4.6434 to 5.9504, while the final score changes by only 0.90 points. Thus, NAS does not rely on a precisely tuned anchor. The main effect is to keep solved value vectors within a bounded operating range, rather than to select one uniquely optimal target norm.

C.3 Robustness under Non-Stationary Edit Orders

A fixed anchor could be brittle if the edit stream is non-stationary, for example when easy and difficult edits appear in different phases. To stress-test this setting, we reorder the first 4,000 CounterFact edits using a difficulty proxy computed once on the original model: the native MEMIT write norm. We compare three orders: the default stream order, low-to-high difficulty, and high-to-low difficulty.

Table 9: Robustness under non-stationary edit orders. Scores are reported at a key horizon for each editor, together with the range across three edit orders. The later-horizon column reports scores under default / low-to-high / high-to-low orders.

Editor	NAS	Key horizon	Default	Low→High	High→Low	Range	Later horizon
AlphaEdit	Off	1500	81.76	82.77	54.70	28.07	@4000: 52.85 / 50.32 / 50.76
AlphaEdit	On	1500	86.25	84.91	87.68	2.77	@4000: 83.44 / 82.01 / 83.76
MEMIT	Off	300	85.87	84.88	51.52	34.35	@1000: 50.57 / 50.54 / 50.37
MEMIT	On	300	90.56	90.76	90.94	0.38	@1000: 80.32 / 78.98 / 68.59

NAS substantially reduces order sensitivity. For AlphaEdit, the cross-order spread at 1,500 edits decreases from 28.07 to 2.77 after enabling NAS; for MEMIT, the spread at 300 edits decreases from 34.35 to 0.38. Under the harshest high-to-low order, MEMIT+NAS still degrades by 1,000

edits, but remains well above the collapsed baseline. These results support the robustness of a fixed oridinal-model anchor while also motivating adaptive anchoring as a possible extension under severe non-stationarity.

C.4 Multi-Hop Counterfactual Editing

We also evaluate NAS on MQuAKE [Zhong et al., 2023], a counterfactual multi-hop editing benchmark. This setting probes whether NAS can stabilize sequential editing without causing an early under-editing penalty on harder reasoning-oriented edits.

Table 10: Sequential multi-hop editing results on MQuAKE-CF-3k-v2. "First < 5" denotes the first edit step where accuracy falls below 5%.

Method	Acc@100	Peak	Acc@3000	First < 5
NAS	16.68	19.87@900	17.46	–
AlphaEdit	16.17	19.28@700	0.04	1400
MEMIT	14.49	14.49@100	0.00	200
ROME	3.33	3.33@100	0.00	100

NAS shows no early under-editing penalty on this multi-hop benchmark: its Acc@100 is comparable to AlphaEdit and higher than MEMIT and ROME. More importantly, NAS maintains 17.46% accuracy at 3,000 edits, whereas AlphaEdit, MEMIT, and ROME collapse to near-zero accuracy. This supports the view that norm anchoring improves long-run stability without simply weakening the edit signal. We also run MEMIT+NAS on the same stream: it reaches 16.37% at 3,000 edits, against 0.00% for vanilla MEMIT, so the effect is not specific to AlphaEdit.

C.5 Additional Short-Stream KnowEdit Benchmarks

To complement the long-stream benchmarks used in the main text, we also evaluate NAS on two shorter KnowEdit-style benchmarks, WikiRecent and WikiBio [Zhang et al., 2024], using Llama-3-8B. These datasets are not intended to stress the ultra-long regime, but they provide additional breadth over more recent and biographical factual updates. Table 11 reports end-of-stream results.

Table 11: End-of-stream results on additional KnowEdit benchmarks. All numbers are percentages. WikiBio does not report paraphrase success under its protocol.

Dataset	Method	Rewrite	Paraphrase	Locality
WikiRecent (1266)	NAS	96.65	62.76	73.20
WikiRecent (1266)	AlphaEdit	95.07	61.80	70.77
WikiRecent (1266)	MEMIT	0.00	0.00	0.00
WikiRecent (1266)	PRUNE	0.00	0.00	0.00
WikiRecent (1266)	RECT	0.00	0.00	0.00
WikiBio test (306)	NAS	71.81	–	35.71
WikiBio test (306)	AlphaEdit	62.14	–	33.04
WikiBio test (306)	MEMIT	0.00	–	0.00
WikiBio test (306)	PRUNE	0.00	–	0.00
WikiBio test (306)	RECT	0.00	–	0.00

NAS remains stable to the end of both short streams and outperforms AlphaEdit on all reported metrics. The results suggest that the gains of NAS are not limited to the full CounterFact, ZsRE, or WikiBigEdit streams used in the main experiments.

C.6 Additional General Capability Experiment

C.6.1 Qwen2.5



Figure 13: GLUE F1 trajectories over sequential edits on Qwen (step 0 denotes the pre-edit baseline).

Appendix Fig. 13 reports the evolution of GLUE F1 on Qwen as the number of edits increases (step 0 is the pre-edit baseline). Across all six tasks, NAS yields consistently stronger retention of general language understanding compared with prior locate-and-edit editors, with the gap becoming most visible in the long-horizon regime where cumulative interference typically dominates. This behavior supports our central claim that NAS acts as a plug-in stabilizer for lifelong knowledge updating: by controlling the destabilizing drift induced by repeated localized rewrites, it substantially reduces collateral degradation on out-of-distribution evaluation suites such as GLUE.

C.6.2 GPT-J

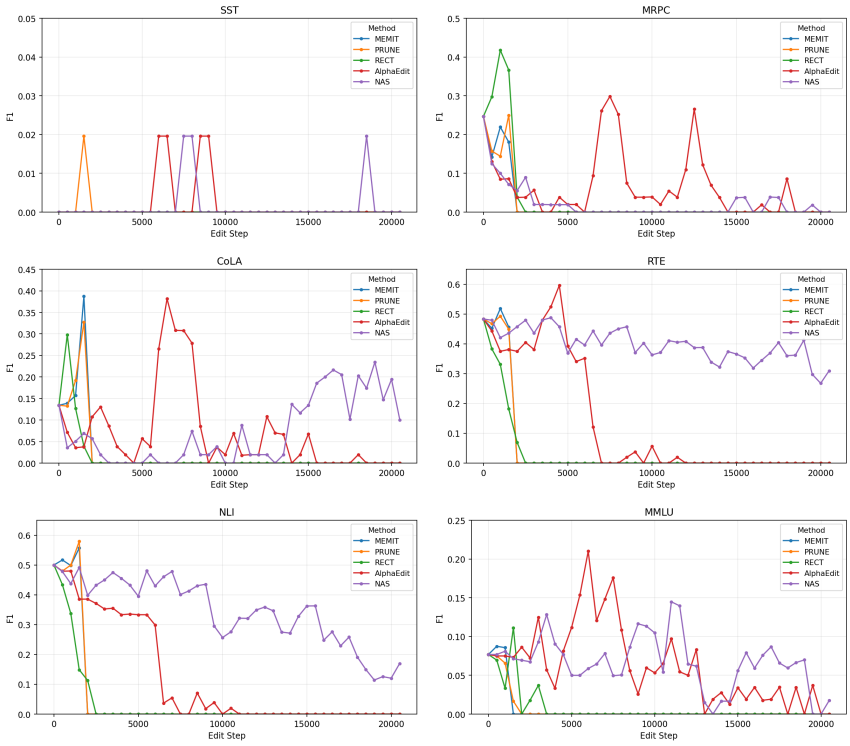


Figure 14: GLUE F1 trajectories over sequential edits on GPT-J (step 0 denotes the pre-edit baseline).

A similar trend is observed on GPT-J in Appendix Fig. 14, where the backbone is generally more fragile under repeated edits. Despite the lower absolute starting performance on some tasks, NAS markedly improves long-horizon stability, maintaining non-trivial GLUE performance in stages where competing editors often exhibit sharp degradation. The fact that the same plug-in provides clear benefits on both Qwen and GPT-J indicates that NAS captures a model-agnostic stabilization effect, rather than exploiting idiosyncrasies of a specific backbone or task, reinforcing its practicality for sequential deployment.

C.7 Plug-in General Capability Results

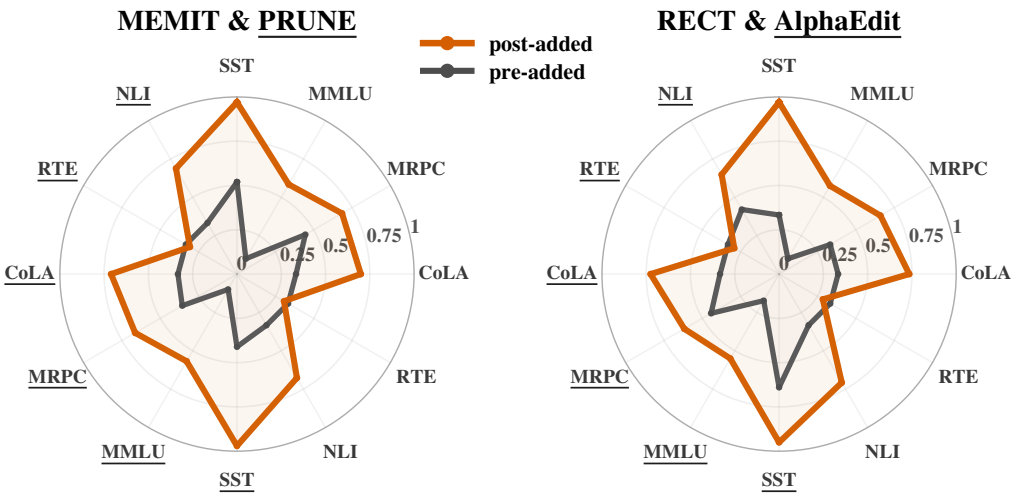


Figure 15: RQ3: **General capabilities comparison.** Radar plots compare pre-added (gray) and post-added (orange) performance on general capabilities (GLUE-style evaluation) for two method pairs: MEMIT & PRUNE (left) and RECT & AlphaEdit (right). For each radar, the first six axes correspond to the first method in the title, and the underlined six axes correspond to the second method.

Figure 15 compares GLUE-style general capability at the same reference point used in the plug-in study. For both method pairs (MEMIT & PRUNE; RECT & AlphaEdit), the NAS-augmented variants dominate the base editors on most axes; the only near-exception is RTE, where the gap is comparatively small. Overall, NAS improves long-horizon editability by mitigating instability, without sacrificing general capabilities.

C.8 Plug in Experiment on GPT-J.

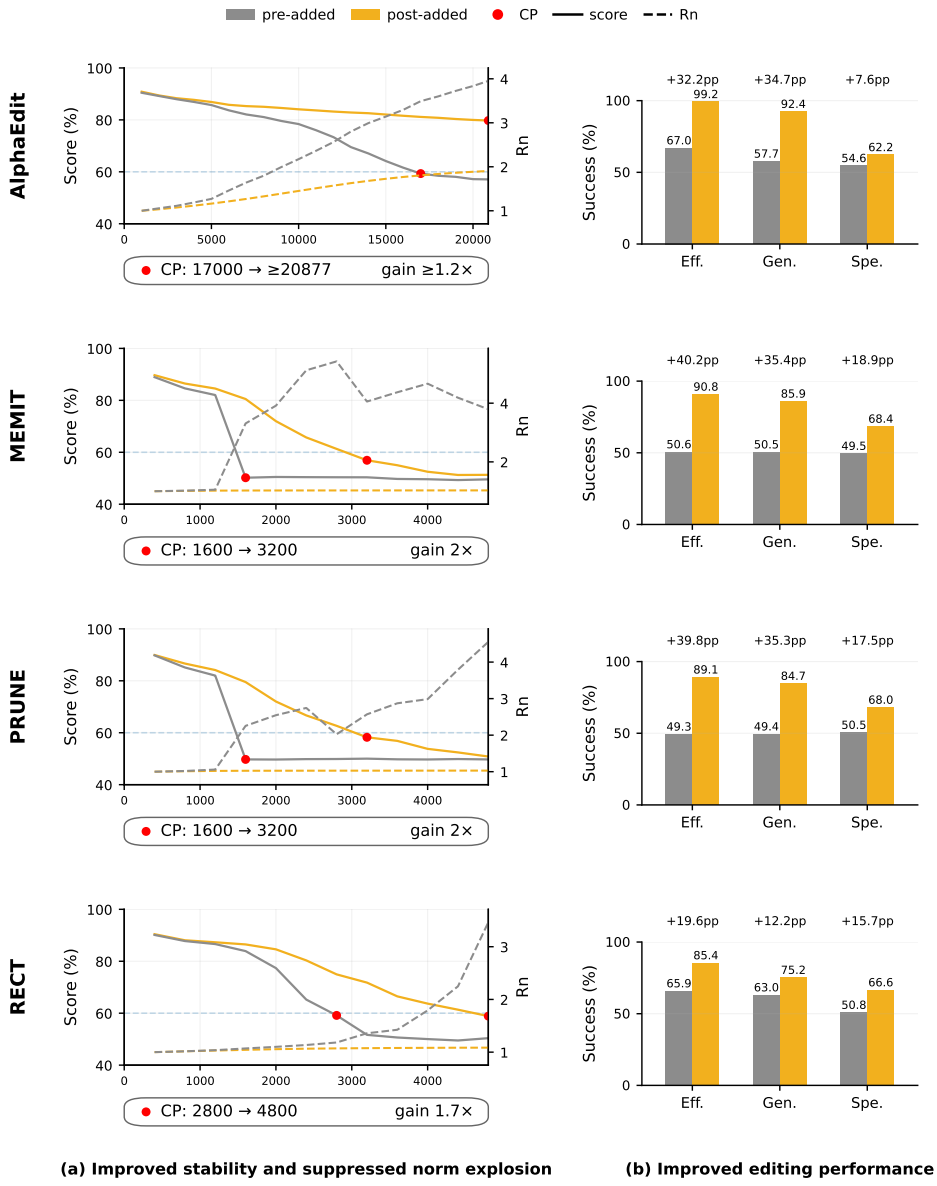


Figure 16: **NAS improves stability and editing performance on GPT-J. (a)** Improved stability and suppressed norm explosion: dual-axis trajectories show the editing score (solid) and relative weight norm R_n (dashed); the red dot marks the collapse point (CP, first step with score ≤ 60). **(b)** Improved editing performance: success at the original (w/o NAS) CP, reported on rewrite/para./neighbor (Eff./Gen./Spe.), comparing pre-added vs. post-added.

We further validate the plug-in behavior of NAS on an additional backbone, GPT-J, using the same experimental setup as in the main text. We integrate NAS into four representative Locate-and-Edit editors (MEMIT, PRUNE, RECT, and AlphaEdit) and run long-horizon sequential editing on CounterFact, tracking both the editing score and the relative weight-norm statistic R_n.

Figure 16 summarizes the results. Across all four editors, attaching NAS reliably suppresses the growth of R_n and delays the collapse point (CP), indicating that the stabilization effect is not specific to Llama3-8B. In particular, NAS extends the editable horizon by $2\times$ for MEMIT and PRUNE (CP:

1600 → 3200), by 1.7× for RECT (2800 → 4800), and by at least 1.2× for AlphaEdit (17,000 → ≥20,877), where the NAS-augmented run does not collapse within the full CounterFact stream.

In addition to improved stability, NAS also improves editing quality under a shared reference condition. Evaluated at the original (w/o NAS) CP of each base editor, NAS substantially increases post-edit success on rewrite/paraphrase/neighborhood (Eff./Gen./Spe.) across methods. The gains are large and consistent, with typical improvements of roughly $+20$–$+40$ percentage points on efficacy and generalization, and $+8$–$+19$ percentage points on specificity, mirroring the main-paper findings. Overall, the GPT-J results corroborate that NAS is a portable, plug-and-play stabilization component that both mitigates instability (via suppressing norm growth) and improves editing performance across diverse L&E baselines.

C.9 Results on Additional Base Models

We further evaluate Locate-and-Edit (L&E) editors on **GPT2-XL** under the **same sequential editing setup as RQ1**. Table 12 reports the long-horizon sequential editing results.

Table 12: Additional Long-Horizon Sequential model editing results on GPT2-XL. Reported values are mean±std for each metric. Pre-edited is separated by a rule; Best in **bold**, second-best in underlined.

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
GPT2-XL	Pre-edited	$21.34_{\pm 0.25}$	$24.07_{\pm 0.31}$	$78.72_{\pm 0.19}$	$527.26_{\pm 0.88}$	$2.86_{\pm 0.04}$	$22.59_{\pm 0.22}$	$21.80_{\pm 0.18}$	$24.33_{\pm 0.21}$
	MEMIT	$49.56_{\pm 0.14}$	$49.56_{\pm 0.14}$	$50.42_{\pm 0.22}$	$496.38_{\pm 0.45}$	$4.34_{\pm 0.11}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	PRUNE	$47.86_{\pm 0.21}$	$48.14_{\pm 0.19}$	$52.72_{\pm 0.25}$	$469.10_{\pm 0.72}$	$4.32_{\pm 0.13}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	RECT	$47.66_{\pm 0.18}$	$47.47_{\pm 0.23}$	$52.32_{\pm 0.29}$	$235.52_{\pm 0.33}$	$0.70_{\pm 0.02}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	AlphaEdit	$\underline{73.92}_{\pm 0.28}$	$\underline{58.16}_{\pm 0.24}$	$\underline{54.20}_{\pm 0.31}$	$\underline{566.62}_{\pm 1.02}$	$\underline{14.53}_{\pm 0.27}$	$\underline{43.62}_{\pm 0.33}$	$\underline{35.90}_{\pm 0.29}$	$\underline{10.52}_{\pm 0.11}$
	NAS	**91.72±0.18**	**73.15±0.25**	**56.76±0.22**	**568.23±0.95**	**27.97±0.31**	**58.54±0.26**	**49.32±0.28**	**12.76±0.15**

Consistent with our main findings, several L&E baselines exhibit severe degradation on ZsRE, collapsing to **0** post-edit success for MEMIT/PRUNE/RECT. ALPHAEDIT remains substantially more stable, and NAS achieves the best performance across *all* reported metrics on both CounterFact and ZsRE. In particular, NAS improves over ALPHAEDIT by **+17.8/+15.0/+2.6** points on CounterFact (Eff./Gen./Spe.) and **+14.9/+13.4/+2.2** on ZsRE, while also markedly increasing consistency (**+13.4**). These results further support the robustness and portability of NAS to additional backbones beyond those in the main text.

C.10 Additional Baselines under Official Sequential-Editing Configurations

Motivation and protocol. To complement the main results, we report additional baselines that are not included in the main tables due to *protocol mismatch*: several non-L&E editors do not support our *atomic sequential editing* setting (i.e., updating exactly one fact per step), and are instead designed and tuned for their own official update granularity and step sizes. To avoid redefining these methods, we evaluate all additional baselines using their *official* sequential-editing configurations and learning-rate/step schedules. MEMOIR does not support our generation-based evaluation, hence Fluency/Consistency entries are marked as NULL.

NAS under two sequential-editing granularities. For a controlled comparison against these official settings, we report two variants of our method: NAS follows the atomic protocol used throughout the paper, whereas NAS† follows the batched sequential-editing configuration (updating 500 facts per update step) to match the granularity used by several additional baselines. Tab. 13 summarizes results after 10,000 sequential edits.

Table 13: Sequential model editing results on additional baselines (10,000 sequential edits). Rows are visually grouped by whether the method belongs to the L&E family; *best/second are computed across **all** methods within each backbone* (ignoring NULL entries). Best in **bold**, second-best in underlined.

Model	Method	CounterFact					ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
LLaMA3	GRACE	**99.08**	10.20	**88.39**	**630.42**	24.40	90.16	1.84	17.52
	WISE	17.90	20.23	80.79	355.55	1.63	31.77	31.26	24.59
	MEMOIR	90.44	64.84	54.71	NULL	NULL	90.15	87.67	31.78
	ENCORE	92.03	84.44	56.98	577.27	26.81	90.86	87.31	31.51
	LyapLock	37.40	26.96	70.38	601.12	27.32	77.09	72.22	28.90
	NAS	98.85	85.50	64.62	619.99	29.93	92.76	88.71	**32.28**
	NAS†	98.64	**88.21**	66.13	625.07	**31.35**	**93.74**	**89.39**	32.18
Qwen2.5	GRACE	96.05	16.49	**84.48**	334.05	6.44	**98.64**	0.85	20.04
	WISE	11.76	36.45	83.94	397.35	3.80	30.91	32.43	25.87
	NAS	98.22	65.48	76.42	621.36	29.83	72.15	60.09	18.78
	NAS†	**99.21**	**77.21**	74.86	**621.37**	**31.76**	98.48	**91.00**	**44.68**
GPT-J	GRACE	99.27	16.40	80.81	**621.62**	29.78	93.77	1.62	17.99
	MEMOIR	87.56	49.93	52.45	NULL	NULL	84.19	78.56	27.23
	WISE	45.85	43.02	58.33	510.89	8.80	32.73	29.08	26.00
	LyapLock	47.56	34.60	**83.05**	591.39	30.82	64.23	58.62	**29.25**
	NAS	99.60	93.36	67.00	611.39	39.57	92.74	81.28	24.32
	NAS†	**99.61**	**93.92**	67.19	609.90	**42.25**	**97.65**	**92.78**	24.60
GPT2-XL	GRACE	**99.14**	16.92	**77.18**	526.09	3.03	92.73	1.39	25.15
	MELO	61.48	47.15	53.92	542.73	11.63	68.15	69.65	19.56
	WISE	33.12	33.48	67.27	541.08	6.20	37.52	39.23	40.27
	ENCORE	92.92	77.09	57.79	514.90	23.40	86.40	77.33	24.66
	LyapLock	31.80	28.68	76.81	**621.09**	31.29	34.76	32.73	25.82
	NAS	96.98	82.03	59.24	557.40	33.18	72.38	64.64	16.93
	NAS†	97.24	**83.77**	59.54	566.36	**35.32**	**97.07**	**87.74**	**43.18**

Summary. Across additional baselines and backbones, NAS consistently achieves a favorable efficacy–generalization balance under sequential editing, while several methods exhibit strong efficacy/locality but weak generalization. We highlight two representative cases below.

Observations. Tab. 13 suggests that additional baselines can score highly on efficacy/locality while failing to generalize. On LLaMA3/CounterFact, GRACE reaches Eff./Spe.=99.08/88.39 (with Flu.=630.42), but its Gen. is 10.20; on LLaMA3/ZsRE, its Gen. further drops to 1.84. In contrast, NAS yields a more balanced profile on the same backbone: on CounterFact, NAS achieves Eff./Gen./Spe.=98.85/85.50/64.62 and NAS† achieves 98.64/88.21/66.13; on ZsRE, NAS attains Eff./Gen.=92.76/88.71 and NAS† attains 93.74/89.39.

Example: GPT-J. On GPT-J, NAS improves efficacy and generalization simultaneously. On CounterFact, NAS and NAS† achieve Eff./Gen.=99.60/93.36 and 99.61/93.92, respectively, compared to GRACE (Gen.=16.40) and WISE (Gen.=43.02). LyapLock attains high specificity (Spe.=83.05) but with substantially lower Eff./Gen. (47.56/34.60). On ZsRE, NAS† achieves Eff./Gen.=97.65/92.78, compared to LyapLock 64.23/58.62.

Transition. To further assess the additional editors of the L&E-paradigm and the effect of enabling NAS as a plug-in, we turn to a longer-horizon evaluation of WikiBigEdit in the next section.

C.11 Plug-in Experiments of Additional L&E methods on WikiBigEdit

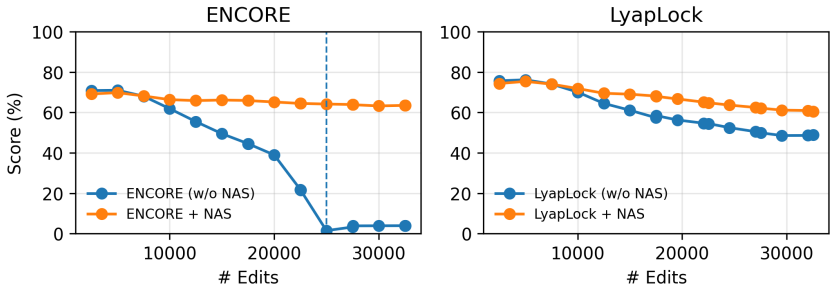


Figure 17: Single-step sequential editing on WikiBigEdit: ENCORE and LyapLock with vs. without NAS (Score $= (\mathrm{ES} + \mathrm{GS} + \mathrm{LS})/3$).

Table 14: Single-step sequential editing on WikiBigEdit at 32,541 edits (%; higher is better).

Method	ES	GS	LS	Score
ENCORE	4.77	3.80	2.92	3.83
ENCORE + NAS	76.83	67.75	**45.91**	63.50
LyapLock	62.60	55.10	29.24	48.98
LyapLock + NAS	75.72	64.13	41.75	60.53
NAS	**80.45**	**68.74**	43.17	**64.12**

Setup. We evaluate long-horizon *atomic sequential editing* on WikiBigEdit, where each edit updates exactly one fact before proceeding to the next. We periodically evaluate at increasing checkpoints up to 32,541 edits. We report Efficacy (ES), Generalization (GS), and Specificity (LS), and summarize them using $\mathrm{Score} = (\mathrm{ES} + \mathrm{GS} + \mathrm{LS})/3$.

Additional plug-in results. We further enable NAS as a *drop-in* component for existing LE editors by toggling a single additional line of code while keeping other hyperparameters and settings unchanged. Fig. 17 compares ENCORE and LyapLock with vs. without NAS. ENCORE without NAS collapses around $\sim$25k edits and degrades to near-zero accuracy thereafter, whereas ENCORE+NAS remains stable through the full horizon. Concretely, at 32,541 edits ENCORE improves from ES/GS/LS = 4.77/3.80/2.92 (Score 3.83) to 76.83/67.75/45.91 (Score 63.50) with NAS (Tab. 14). Similarly, NAS consistently improves LyapLock across checkpoints, yielding ES/GS/LS = 75.72/64.13/41.75 (Score 60.53) at 32,541 edits, compared to 62.60/55.10/29.24 (Score 48.98) without NAS.

Reference performance of NAS. For reference, we also report results of NAS under the same setting. At 32,541 edits, NAS attains ES/GS/LS = 80.45/68.74/43.17 (Score 64.12) (Tab. 14).

Takeaway. Overall, NAS serves as a simple and effective plug-in that substantially improves long-horizon stability under single-step sequential editing when combined with prior LE methods.

C.12 Increment-Level Clamps Do Not Break the Loop

Standard L&E code exposes a clamp factor c (`clamp_norm_factor`) that limits the optimized increment to $\|\Delta\| \leq c\,\|v^{\mathrm{old}}\|$. The reference $\|v^{\mathrm{old}}\|$ is read from the current, edited model. Under the norm-feedback loop it grows, so the admissible increment grows with it. NAS instead fixes $\|v^{\mathrm{new}}\|$ to a statistic of the original model. To check whether tuning c can replace the anchor, we sweep c under atomic sequential editing on Llama3-8B and CounterFact.

Table 15: Clamp sweep for AlphaEdit (Llama3-8B, CounterFact, atomic). Raw prompt-recall Rewrite/Paraphrase/Neighborhood accuracy (%); a collapsed model scores about 0. ≈1: all three metrics about 1%. NAS uses the default clamp 0.75 unchanged.

Clamp c	@1k	@2k	@4k	@6k	@8k	@10k
0.25	89.4/27.1/23.5	88.4/28.0/22.3	87.4/29.0/20.5	86.4/30.5/18.3	85.5/31.0/17.1	84.0/31.4/16.0
0.75 (default)	96.2/75.3/19.0	90.0/74.1/17.1	4.8/2.3/1.4	≈1	≈1	1.4/0.2/0.1
1.00	36.7/34.8/8.4	0.0/0.0/0.0	3.2/0.7/0.7	2.9/0.5/0.4	–	–
NAS ($c = 0.75$)	98.2/65.2/21.2	97.4/63.3/18.1	96.6/59.0/15.7	96.0/57.6/14.6	–	93.9/55.0/12.3

For AlphaEdit, a tight clamp ($c = 0.25$) avoids collapse but loses paraphrase generalization from the first thousand edits (27–31% throughout). The shipped default ($c = 0.75$) is healthy early and collapses between 3k and 4k edits, and $c = 1.0$ collapses almost immediately. NAS keeps the early quality of the default and does not collapse within 10k edits; a second run with a different edit order gives 93.9/56.1/12.3 at 10k.

Table 16: Clamp sweep for MEMIT (Llama3-8B, CounterFact, atomic). Probability-criterion Rewrite/Paraphrase/Neighborhood success (%), where a collapsed model scores about 50 by chance; last column: raw prompt-recall rewrite accuracy at 3k.

Clamp c	@2k	@3k	Raw rewrite @3k
0.25	–	47.9/47.9/52.1	0.0
0.50	–	48.8/48.8/51.2	0.0
0.75 (default)	51.3/51.3/48.8	51.2/51.2/48.8	0.0
1.00	48.0/48.0/52.1	48.3/48.3/51.7	0.0

For MEMIT, every clamp value is already at chance level at its first available checkpoint (2–3k edits), with raw rewrite accuracy 0.0. MEMIT+NAS edits successfully up to about 1.5k edits, a $3.75\times$ longer horizon than vanilla MEMIT (Fig. 5). No value of c extends the MEMIT collapse point, and on AlphaEdit the values that avoid collapse do so by weakening every edit.

C.13 Magnitude versus Direction: Interventional Evidence

The analysis in Section 3.1 is built from correlations between norms. This section tests the loop by intervening on it.

Switching the anchor off and on. We run ROME+NAS on Llama3-8B with three edit orders and turn NAS off at three points in the stream (9 runs). In all 9 runs collapse returns within 15–50 further edits. After the switch, $\log \|W\|^2$ grows at the rate that Prop. 3.2 predicts from the per-step statistics of a separate vanilla run started from scratch: the ratio of measured to predicted rate lies in $[0.98, 1.33]$, with fit $R^2 \geq 0.94$. In the other direction, we start vanilla ROME and turn NAS on near the point where the loop starts ($n \in \{5, 10\}$). The norm growth stops and the run stays healthy for more than 190 further edits, about $4\times$ the vanilla collapse horizon. At $n = 10$, the held-out counterfactual-preference rate drops back from 22.2 to 9.6 (about 6 before editing, about 50 at collapse), and neighborhood specificity rises from 79 to 86. This is the approach to β from above discussed in Appendix B.4.

Weight surgery. We take sequential AlphaEdit checkpoints on Llama3-8B and build two counterfactual weights for each edited layer: *mag-only* keeps the direction of W_0 and uses the norm of the edited W_n; *dir-only* keeps the direction of W_n and uses the norm of W_0. We measure teacher-forced accuracy on 300 held-out facts (base model: 0.292).

Table 17: Counterfactual weight surgery on sequential AlphaEdit checkpoints (Llama3-8B). Norm ratio: $\|W_n\|/\|W_0\|$, min–max over the 5 edited layers. Teacher-forced accuracy on 300 held-out facts; base model 0.292.

Checkpoint	Norm ratio	Edited	Mag-only	Dir-only
vanilla @500	1.0013–1.0079	0.265	0.292	0.265
vanilla @1000	1.0037–1.0249	0.163	0.292	0.163
vanilla @1500	1.0069–1.0502	0.167	0.292	0.170
vanilla @2000	1.0107–1.0816	0.157	0.295	0.153
vanilla @2500	1.0151–1.1195	0.127	0.295	0.133
vanilla @3000	1.0199–1.1673	0.097	0.292	0.122
vanilla @3500	1.0272–1.2696	0.043	0.295	0.057
vanilla @4000	1.0627–1.7092	0.007	0.295	0.017
vanilla @4500	1.1933–2.7628	0.030	0.272	0.040
vanilla @5000	1.4446–4.2380	0.000	0.215	0.027
NAS @5000	1.0060–1.0210	0.202	0.292	0.198

Dir-only follows the edited model at every checkpoint, so the damage is carried by the direction of the edited weights. Mag-only stays at the base level up to a norm inflation of about $1.7\times$ and drops only beyond about $2.8\times$ (@4500 onward). Norm growth alone is therefore tolerated up to a point and harmful beyond it. Under NAS the norm ratio stays at most 1.021, mag-only equals the base model, and the residual drift is small.

Dose–response. We fix the direction of the perturbation to the accumulated edit direction and scale its magnitude by α. Held-out specificity falls monotonically with the dose: 86, 66, 54, and 51 for $\alpha = 0.1, 0.25, 0.5, 1.0$. A perturbation with the same Frobenius norm as $\alpha = 1.0$ but a random direction is largely harmless (88 versus 51). The harm comes from magnitude added along the direction that the loop builds up, not from the norm by itself. NAS acts only on the magnitude, and under NAS this directional damage does not develop.

Vector-level signature of the loop. Under vanilla editing, successive read-out vectors v_n^{old} line up with a single direction: their sliding-window cosine similarity rises from 0.38 to 0.96, and the output of the edited MLP grows to $23.5\times$ the norm of the residual stream. Under NAS both quantities stay flat (cosine at most 0.17; ratio about $0.6\times$). This is a direct measurement of the loop closing, and of the layer incompatibility described in Section 3.2: downstream layers receive inputs far outside their usual range.

C.14 Short Streams and Retention of Earlier Edits

Many applications apply far fewer edits than our full streams. Table 18 reports the first 4k edits of the atomic CounterFact runs on Llama3-8B. At every checkpoint all previously edited facts are re-evaluated with the current model, so the numbers include the earliest edits.

Table 18: Short-horizon atomic sequential editing (Llama3-8B, CounterFact). Probability-criterion Rewrite/Paraphrase/Neighborhood success (%) over all facts edited so far; a fully collapsed model scores about 50 by chance.

Editor	@500	@1000	@2000	@4000
AlphaEdit	99.6/92.3/72.1	98.6/93.3/65.4	97.5/93.5/62.4	54.7/52.8/51.2
AlphaEdit+NAS	100.0/89.5/80.0	99.8/88.7/76.5	99.6/88.2/72.8	99.5/87.2/69.3
ROME	70.6/63.8/50.3	71.4/66.1/48.9	68.3/63.3/49.3	61.9/58.3/48.9
ROME+NAS	89.6/82.0/83.0	87.4/81.3/81.1	84.3/78.0/77.4	79.5/72.4/75.0
MEMIT	48.8/49.0/51.0	51.9/51.9/48.1	51.2/51.2/48.8	51.2/51.2/48.8
MEMIT+NAS	99.0/89.2/80.5	91.1/82.6/70.1	42.2/39.2/62.9	44.9/39.3/60.3

At short horizons NAS matches or slightly exceeds rewrite efficacy and gains 8–11 points of neighborhood specificity. On AlphaEdit it costs 2.8–5.3 points of paraphrase accuracy up to 2k edits; by 4k vanilla AlphaEdit has collapsed (paraphrase 52.8 versus 87.2). On ROME, NAS is better on all three metrics at every checkpoint. Vanilla ROME is near the chance level from early on; in raw prompt-recall it collapses almost immediately (rewrite 26.0% at 100 edits, 6.5% at 200). On MEMIT, the weakest host (vanilla collapse point about 400), NAS makes 500- and 1,000-edit streams usable and degrades after its extended collapse point of about 1.5k, as in Fig. 5.

Retention. Without replay, sequential forgetting has two sources: collapse through the norm-feedback loop, which loses all edits at once, and gradual rank-one interference between edits (Eq. (80)). NAS removes the first. ROME+NAS still answers 79.5% of all 4,000 edited facts at the end of the stream, while vanilla ROME is at the chance level by then. The slow decline of ROME+NAS from 89.6 to 79.5 between 500 and 4,000 edits is the remaining interference channel. The two kinds of protection add up: replay without the anchor (vanilla AlphaEdit) is at 54.7, the anchor without replay (ROME+NAS) at 79.5, and both together (AlphaEdit+NAS) at 99.5.

C.15 Long-Form Editing on UnKEBench

We evaluate long-form targets on UnKEBench [Deng et al., 2025], which contains 1,000 counterfactual long-form edits and is also used by AnyEdit [Jiang et al., 2025]. We apply the full stream with atomic sequential editing on Llama3-8B and report teacher-forced token accuracy and NLL on the edited question and its paraphrase after all 1,000 edits.

Table 19: Sequential long-form editing on UnKEBench (Llama3-8B, 1,000 atomic edits). Teacher-forced token accuracy (%) and NLL, Rewrite/Paraphrase.

Editor	Accuracy (Rew./Para.)	NLL (Rew./Para.)
MEMIT	0.47 / 0.47	11.98 / 11.98
MEMIT+NAS	70.93 / 69.47	1.05 / 1.11
AlphaEdit	2.70 / 2.46	8.75 / 8.82
AlphaEdit+NAS	84.68 / 77.66	0.52 / 0.78

Both vanilla editors are destroyed by the stream (at most 3% token accuracy after 1,000 edits). With NAS the same editors keep 71–85%, and NLL is 17–23$\times$ lower, so the result does not depend on the accuracy criterion. Making rank-one L&E editors good at long-form targets is a separate problem, addressed by UnKE and AnyEdit with different update rules; NAS does not address it. The result shows that the same stabilization still works when the targets are long. The gap is larger than on CounterFact. One possible reason is that long-form targets produce larger unconstrained $\|\hat{v}^{\text{new}}\|$; we have not measured this directly.

C.16 Composition with SPHERE

SPHERE [Liu et al., 2026] restricts the *direction* of updates by projecting out components along the principal directions of the pretrained weights. NAS fixes the *magnitude* of the written value. The two act on different quantities, so they can be combined. We port the official SPHERE projection (recommended $\beta = 0.5$, $\alpha = 0.5$) into our AlphaEdit pipeline; AlphaEdit is also SPHERE's official host. Long runs use an equivalent thin-SVD implementation of the same projection, which matches the reference implementation to a maximum elementwise difference of 5×10^{-15}. All variants share one code path and protocol.

Table 20: Composition with SPHERE, atomic sequential editing on CounterFact. Raw prompt-recall Rewrite/Paraphrase/Neighborhood accuracy (%) after 10k edits; a collapsed model scores about 0. –: not run.

Variant	Llama3-8B	GPT-J	Qwen2.5-7B
AlphaEdit	1.4/0.2/0.1	79.5/42.2/3.6	82.0/24.0/10.1
+SPHERE	61.0/35.2/7.5	79.7/45.9/4.2	80.6/23.4/10.4
+NAS	93.9/55.0/12.3	98.5/67.1/7.3	93.9/32.0/9.9
+NAS+SPHERE	93.7/51.2/13.0	–	–

Table 21: Composition with SPHERE, 500 facts per step (Llama3-8B, CounterFact). Score = harmonic mean of probability-based Eff./Gen./Spe.

Variant	@10k	@20k
AlphaEdit	79.59	70.77
+SPHERE	80.35	75.09
+NAS	81.68	77.83
+NAS+SPHERE	**82.03**	**78.53**

On the atomic stream, vanilla AlphaEdit collapses on Llama3-8B between 3k and 4k edits (a second edit order gives 0.4/0.1/0.0), and SPHERE slows this decay substantially. GPT-J and Qwen2.5 degrade more slowly but stay well below NAS. Adding SPHERE to NAS gives the same rewrite accuracy on Llama3-8B (93.7 versus 93.9, within seed variation; a second edit order of AlphaEdit+NAS gives 93.9/56.1/12.3), the best neighborhood accuracy (13.0), and a lower paraphrase accuracy (51.2 versus 55.0). At batched granularity the combination is best at both checkpoints. Fixing the magnitude is the main lever, and the direction constraint adds a smaller, consistent gain on top.

C.17 Batched Granularity: Full Results

Table 22 gives the per-metric numbers behind Table 2. All runs use 500 facts per step and the same streams as the atomic experiments.

Table 22: Sequential editing with 500 facts per step. CounterFact: Score (Eff./Gen./Spe.), probability-based. ZsRE: raw-accuracy Eff./Gen./Spe. End of stream: 20k edits for CounterFact, 19,086 for ZsRE. NAS† = AlphaEdit+NAS.

Setting	Method	@10k	End of stream
CF, Llama3-8B	MEMIT	58.59 (68.1/61.7/49.2)	53.83 (61.1/52.3/49.4)
	PRUNE	61.08 (70.3/65.4/51.0)	55.80 (65.4/55.2/49.1)
	RECT	64.98 (72.6/70.8/54.8)	57.11 (66.3/57.5/49.9)
	AlphaEdit	79.59 (97.5/91.3/60.7)	70.77 (86.8/77.1/55.8)
	NAS†	**81.68** (98.6/87.3/66.1)	**77.83** (97.6/84.0/61.0)
CF, GPT-J	AlphaEdit	79.77	62.45
	NAS†	**84.65**	**80.30**
ZsRE, Llama3-8B	MEMIT	6.0/5.3/1.4	1.9/1.9/1.2
	AlphaEdit	92.0/87.3/31.3	86.5/82.1/38.2
	NAS†	**92.9/88.0/32.4**	**90.5/86.4/40.6**
ZsRE, GPT-J	AlphaEdit	86.0/76.8/23.0	56.0/48.7/21.1
	NAS†	**96.5/88.9/23.9**	**82.6/73.4/30.1**

NAS† leads or ties in every comparison, and the gap to AlphaEdit grows with stream length (Llama3-8B CounterFact $+2.1 \rightarrow +7.1$; GPT-J CounterFact $+4.9 \rightarrow +17.9$; GPT-J ZsRE rewrite $+10.5 \rightarrow +26.6$). On Llama3-8B CounterFact at 10k, AlphaEdit has higher paraphrase accuracy (91.3 versus 87.3), and NAS† is better on efficacy and specificity.

C.18 Runtime Overhead of Norm-Anchor Scaling (NAS)

We compare the wall-clock runtime with and without NAS under identical editing hyperparameters. Following our sequential-editing setup on **Llama3-8B**, we run n=100 edits for each method and report the per-edit runtime (mean±std). We summarize the relative overhead as $\Delta\% = \frac{t_{\text{NAS}} - t_{\text{base}}}{t_{\text{base}}} \times 100\%$.

Table 23: Runtime comparison with/without NAS. We report mean±std seconds per edit over n=100 edits. $\Delta\%$ is computed on the per-edit mean runtime.

Method	w/o NAS (s/edit)	w/ NAS (s/edit)	$\Delta\%$
MEMIT	5.20±0.82	5.20±0.68	+0.00
PRUNE	5.17±0.62	5.18±0.68	+0.19
RECT	8.18±0.84	8.08±0.78	-1.22
AlphaEdit	6.75±0.73	6.78±0.79	+0.44

Overall, enabling NAS introduces negligible runtime overhead. Across all tested editors, the per-edit runtime change remains within $\pm 1.3\%$, and is typically below 0.5%. Given that NAS only adds lightweight vector re-scaling (without extra optimization steps), the observed differences are well within the natural variance of per-edit runtime (see the reported std values), supporting that NAS is effectively runtime-free in practice.

D Examples of CounterFact and ZsRE dataset

D.1 CounterFact Examples

1. **Twin-city relation (P190).**
   - **Subject:** Lyon
   - **Prompt template:** `What is the twin city of {}? It is`
   - **Target (true → new):** Beirut → Manila
   - **Context / paraphrase prompts (examples):**
     - `Lyon is a twin city of`
     - `The twin city of Lyon is`
   - **Locality prompts (same relation; other subjects, examples):**
     - `What is the twin city of Los Angeles? It is`
     - `Athens is a twin city of`
     - `The twin city of Beijing is`
   - **Neighborhood prompts (subject-related, examples):**
     - `Lyon's twin city is known for`
     - `People in Lyon's twin city speak the language of`
2. **Mother-tongue relation (P103).**
   - **Subject:** Thomas Joannes Stieltjes
   - **Prompt template:** `The mother tongue of {} is`
   - **Target (true → new):** Dutch → English
   - **Context / paraphrase prompts (examples):**
     - `Thomas Joannes Stieltjes spoke the language`
     - `Thomas Joannes Stieltjes, speaker of`
   - **Locality prompts (same relation; other subjects, examples):**
     - `The mother tongue of Rob Birza is`
     - `Arend Lijphart is a native speaker of`
     - `The native language of Charlie Chaplin is`
3. **Citizenship relation (P27).**
   - **Subject:** Mahmoud Fawzi

- **Prompt template:** `{} has a citizenship from`
- **Target (true → new):** Egypt → Germany
- **Context / paraphrase prompts (examples):**
  - `Mahmoud Fawzi holds a citizenship from`
  - `Mahmoud Fawzi, who is a citizen of`
- **Locality prompts (same relation; other subjects, examples):**
  - `Imhotep, who is a citizen of`
  - `Marc Forster holds a citizenship from`
  - `Katja Ebstein, who is a citizen of`

D.2 ZsRE Examples

1. **Publisher query.**
   - **Subject:** Alien Front Online
   - **Prompt:** `What company published Alien Front Online?`
   - **Target (true → new):** Sega → 2K Games
   - **Rephrase prompt:** `Which company released Alien Front Online?`
   - **Locality (Relation_Specificity, examples):**
     - `The country of origin of Alien Front Online is` → Japan
     - `Alien Front Online country of origin` → Japan
   - **Portability (Reasoning, example):**
     - `Who is the parent company of the publisher of Alien Front Online?` → Take-Two Interactive
2. **Programming-language query.**
   - **Subject:** GNOME Chess
   - **Prompt:** `What programming language was used to write GNOME Chess?`
   - **Target (true → new):** Vala → Python
   - **Rephrase prompt:** `How is the programming language for GNOME Chess?`
   - **Locality (Relation_Specificity, examples):**
     - `The platform of GNOME Chess is` → Unix-like operating system
     - `GNOME Chess platform` → Unix-like operating system
   - **Portability (Reasoning, example):**
     - `Who created the programming language used to write GNOME Chess?` → Guido van Rossum
3. **Launch-year query.**
   - **Subject:** Old Quebec Street Mall
   - **Prompt:** `When was Old Quebec Street Mall launched?`
   - **Target (true → new):** 2003 → 2002
   - **Rephrase prompt:** `When did Old Quebec Street Mall open?`
   - **Locality (Relation_Specificity, examples):**
     - `The located in the administrative territorial entity of Old Quebec Street Mall is` → Guelph
     - `Old Quebec Street Mall located in the administrative territorial entity` → Guelph
   - **Portability (Reasoning, example):**
     - `What major sporting event took place the same year Old Quebec Street Mall was launched?` → Salt Lake City Winter Olympics

E Relation to Prior Norm-Related Work

Table 24 expands Table 4.

Table 24: Prior norm-related work on sequential editing, compared with NAS.

Work	Reports norm growth	Closed-form mechanism	Intervention	Constraint reference	Write size as n grows
Gupta et al. [2025a]	yes	no	none	–	–
ENCORE [Gupta et al., 2025b]	yes	no	early stopping, norm regularizer	current weights	not analyzed
HSE [Fang et al., 2025] (Cor. 2)	no	proof for HSE's own update rule	replay and proportional constraint	current weights ($\|\delta_i\|_F \propto \|W_{i-1}K_i\|_F$)	shrinks with n
NAS (ours)	yes	yes, for the standard unconstrained L&E update (Lemma 3.1, Prop. 3.2)	fix $\|v^{\text{new}}\|$ to τ	original model	does not shrink (RQ4, Eq. (66))

HSE's convergence proof uses a replay term that damps updates roughly as $(q/n)^{3/2}$ and assumes orthogonal keys; the proof itself notes that new knowledge becomes harder to integrate as $n \to \infty$. NAS keeps the size of each write independent of n, and single-edit efficacy is preserved even for the most strongly rescaled edits (Section 4, RQ4). Gupta et al. document monotone Frobenius-norm growth together with activation-norm decay and a shift of the activation subspace. Our representation-drift probe (Fig. 3) and the vector-level measurements in Appendix C.13 are consistent with these observations.

F Broader Impact Discussion

NAS may improve the reliability of factual updates by reducing unintended degradation during long edit streams. However, more stable model editing could also make misleading, biased, or harmful factual edits easier to apply at scale. Responsible use requires auditing edited behavior, tracking edit provenance, and respecting the licenses and access policies of the underlying models and datasets.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the scope as long-horizon sequential Locate-and-Edit model editing. The norm-feedback diagnosis and NAS intervention are supported by Section 3.2; the main multi-model and multi-dataset claims are supported by Section 4 and Table 1; the reported $> 4\times$ horizon extension and 72.2% long-run performance improvement are supported by RQ3 in Section 4.4; and the one-line integration and negligible-overhead claims are supported by Appendix A.3 and Appendix C.18.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The conclusion discusses the fixed original-model reference norm as a limitation of NAS and identifies adaptive or conditional anchoring as future work. Computational cost and runtime overhead are reported in Appendix A.3 and Appendix C.18.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: The paper states the main theoretical results as Lemma 3.1, Proposition 3.2, and Corollary 3.3, with the empirical scaling assumptions stated in Section 3.1. These assumptions are empirically validated by the supporting fits in Figs. 6, 7, and 9. Full derivations are provided in Appendix B.2, Appendix B.3, and Appendix B.4, with the general $C \neq I$ case discussed in Appendix B.5.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper describes the datasets, metrics, backbone models, baselines, and long-horizon sequential-editing protocol in Section 4.1 and Appendix A. Implementation details, anchor estimation, baseline configurations, and compute resources are provided in Appendix A.3 and Appendix A.4. The public code repository (`https://github.com/SasyaTitech/NAS`) includes the main experiment code, including NAS, implementations or integrations of the evaluated baseline editors, dataset/evaluation utilities, smoke tests, runtime/anchor ablation scripts, and a README with environment setup, data/model access instructions, example commands, and result summarization commands.

   Guidelines:

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

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code used for the main experiments is publicly available at `https://github.com/SasyaTitech/NAS`, including NAS, evaluated editor implementations or integrations, dataset wrappers, evaluation scripts, tests, and ablation/runtime scripts. Its README specifies the Python environment, installation commands, data and model access requirements, configurable paths, example experiment commands, smoke tests, and result summarization commands. The repository link is given in the abstract.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4.1 specifies the backbone models, datasets, metrics, baselines, and long-horizon sequential-editing protocol. Appendix A provides detailed dataset and metric definitions, and Appendix A.3 describes NAS anchor estimation, integration details, and compute resources. Appendix A.4 states that baselines use official implementations and recommended hyperparameters whenever available, and the released code includes the corresponding hyperparameter files under `hparams/`. Shared experimental settings, including the edit streams, evaluation checkpoints, and base-editor configurations when applicable, are kept consistent across compared methods.

Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The main long-horizon quantitative tables report mean$\pm$std for each metric in Table 1 and Appendix Table 12. The anchor-estimation ablation in Appendix A.3 reports variability over 5 restarts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix A.3 reports that experiments were run on a single NVIDIA H100 GPU and that the primary in-weight L&E runs require at least 48 GB GPU memory. Appendix C.18 reports per-edit wall-clock runtime with and without NAS for representative L&E editors, and the repository README notes that full long-horizon runs require a single large-memory GPU and many hours of compute.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work studies model-editing methods using existing public benchmarks and pretrained language models, does not involve human subjects or crowdsourcing, and does not introduce a new dataset containing personal or sensitive information. We have reviewed the NeurIPS Code of Ethics and do not identify any deviation from it.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Appendix F discusses both potential positive impacts, such as more reliable factual updates during long edit streams, and potential negative impacts, such as easier large-scale application of misleading, biased, or harmful factual edits.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The submission does not release new model weights, edited model checkpoints, or new high-risk datasets. The released code uses existing public benchmarks and externally hosted pretrained language models; full language-model weights are not included, and gated

models must be obtained through the original providers and their access policies. Potential misuse of stable model editing is discussed in Appendix F.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: Existing models, datasets, and baseline methods are cited in Section 4.1, Appendix A.1, and Appendix A.4. The repository README describes model/data access paths and includes a licenses-and-terms note requiring users to follow the licenses, model cards, dataset cards, and terms of use of the original providers. Full model weights are not redistributed.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces and documents a new code package for NAS and the associated experiments, released at `https://github.com/SasyaTitech/NAS`. The README describes the package structure, installation, configurable paths, supported models/datasets/methods, quick checks, experiment commands, result formats, smoke tests, resource notes, and license/terms information. The submission does not introduce new datasets or release new model weights.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments. All evaluations use existing benchmarks and automated model-evaluation protocols.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The work does not involve human-subject experiments, crowdsourcing, or newly collected participant data, so IRB approval or equivalent human-subjects review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method does not use an LLM as an additional non-standard research component for data generation, annotation, evaluation, or algorithmic decision-making. Pretrained LLMs are the experimental subjects being edited and evaluated, and the specific backbone models are disclosed in Section 4.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.