FISHEREDIT: FISHER INFORMATION GUIDED KNOWLEDGE EDITING FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sequential knowledge editing in Large Language Models (LLMs) often leads to a degradation of general capabilities due to cumulative parameter interference. In this paper, we revisit prominent projection-based methods like AlphaEdit and reveal a surprising mismatch: their robustness is primarily driven by update regularization rather than the intended null-space projection. Based on this insight, we propose FisherEdit, a theoretically grounded method built on Fisher geometry, which replaces coarse activation-space projections with importance-aware regularization in parameter space. It penalizes changes to sensitive parameters more strongly while retaining flexibility for new facts, and admits an efficient row-wise closed-form solution. Experiments on Llama-3, Qwen2.5, and Olmo-3 across multiple benchmarks demonstrate that FisherEdit achieves state-of-the-art performance while better preserving model stability and downstream task accuracy.

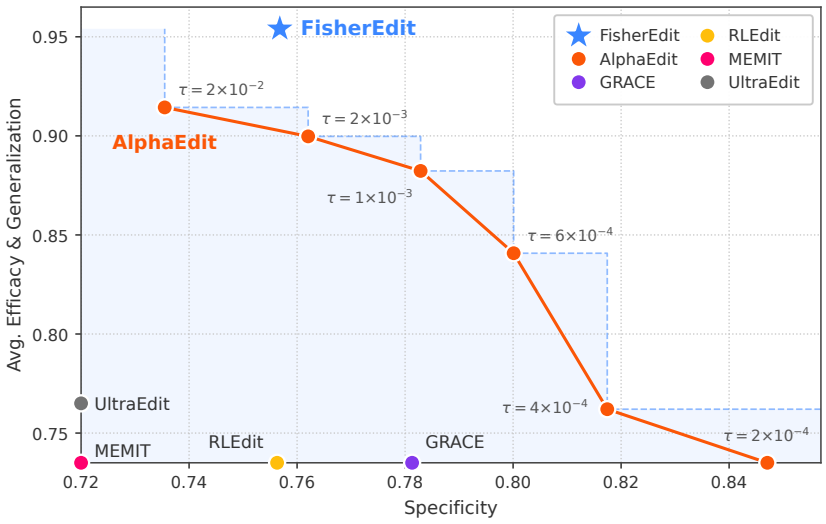

Figure 1: Efficacy–locality trade-off on Llama-3-8B-Instruct. Sweeping AlphaEdit's truncation threshold τ traces a monotone Pareto curve. FisherEdit lands strictly above the curve.

1 INTRODUCTION

Large language models (LLMs) encode extensive factual knowledge during pretraining, and this knowledge must often be revised as facts become outdated, incorrect, or inconsistent with new evidence (Petroni et al., 2019; Roberts et al., 2020; Geva et al., 2021). Knowledge editing addresses this need by updating specific facts without full retraining, while aiming to preserve unrelated behavior (Yao et al., 2023; Wang et al., 2024b). This goal is especially difficult in sequential editing, where repeated weight changes can accumulate, interfere with one another, and degrade the model's general abilities (Yang et al., 2024). Maintaining reliable behavior under repeated edits, therefore, remains a central challenge for knowledge editing.

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

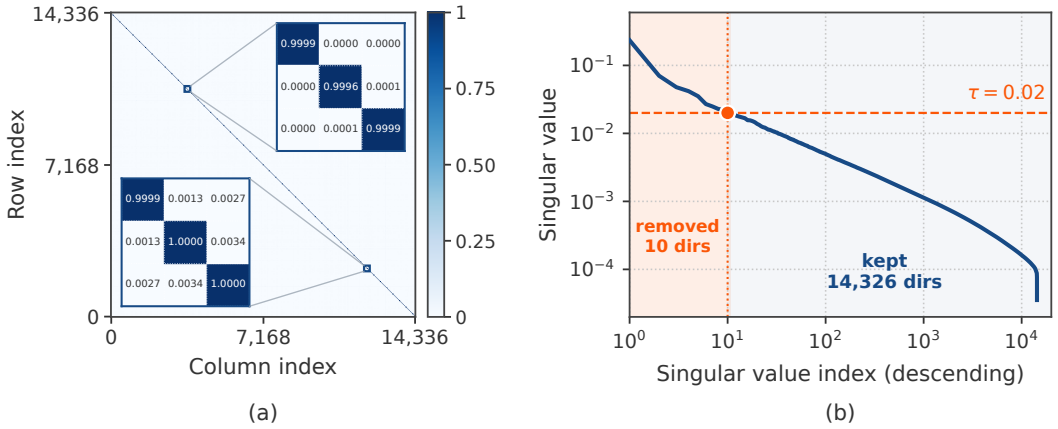

Figure 2: (a) AlphaEdit's null-space projector $\mathbf{P}_\tau$ on Llama-3-8B-Instruct is close to the identity matrix, indicating that most edit directions are only weakly affected by the projection. (b) Singular spectrum of $\mathbf{K}_0\mathbf{K}_0^\top$ on the same layer. Only a few dozen directions exceed the truncation threshold τ.

A prominent approach is direct weight editing. Locate-then-edit methods (Meng et al., 2022a;b) identify where factual associations are stored and update the corresponding weights. Because repeated updates can interfere with previously edited or preserved knowledge, recent methods add preservation constraints to restrict how the model changes. AlphaEdit (Fang et al., 2025) is a representative example: it uses a null-space projection to steer edits away from activation directions associated with prior knowledge. This projection-based view has become influential in sequential editing and has motivated several follow-up methods (Xu et al., 2025; Ikram et al., 2026).

Given AlphaEdit's influence, we revisit it to examine what its null-space projection actually contributes. AlphaEdit offers a clean geometric intuition: edits are projected away from activation directions associated with prior knowledge, so the corresponding knowledge should be preserved. However, this intuition only partially explains its empirical behavior. As illustrated in Fig. 2, the default projector remains close to the identity and removes only a small portion of the activation space, leaving most update directions largely unchanged. Our more systematic analysis further shows that preservation in AlphaEdit is strongly shaped by update regularization, not projection alone. This motivates us to rethink preservation in sequential editing as a problem of designing more precise update regularization.

The key issue is how to constrain edits without removing the flexibility needed to encode new facts. Existing preservation mechanisms often operate at a coarse level: hard subspace constraints remove entire directions (Lyu et al., 2025; Xu et al., 2025; Ikram et al., 2026), while regularization-based methods typically constrain the update through global magnitude, complexity, or spectral properties (Gupta et al., 2025; Liu et al., 2025; Zhang et al., 2026). While effective, these constraints do not directly distinguish parameters by their importance to general behavior. Parameter sensitivity in LLMs is highly uneven: some weight perturbations strongly affect general predictions, while others have limited impact (Frantar & Alistarh, 2023; Lin et al., 2024). Effective preservation should therefore be importance-aware, imposing stronger constraints on sensitive parameters while leaving less critical parameters available for editing. Following this principle, we propose **FisherEdit**, a Fisher-guided regularization method for sequential knowledge editing. It uses a diagonal Fisher estimate to measure parameter sensitivity and weight the update penalties accordingly. This constrains sensitive weights more strongly while allowing less sensitive weights to adapt to new facts. Our contributions are as follows:

- We re-examine AlphaEdit and find that its preservation benefit comes mainly from update regularization, with little contribution from the default null-space projection. Strengthening the projection trades edit efficacy for locality.
- Based on our analysis, we rethink preservation in sequential editing as regularization and introduce FisherEdit, a theoretically motivated method that uses Fisher geometry to derive an importance-aware, element-wise update penalty, together with a specialized row-wise closed-form solver.

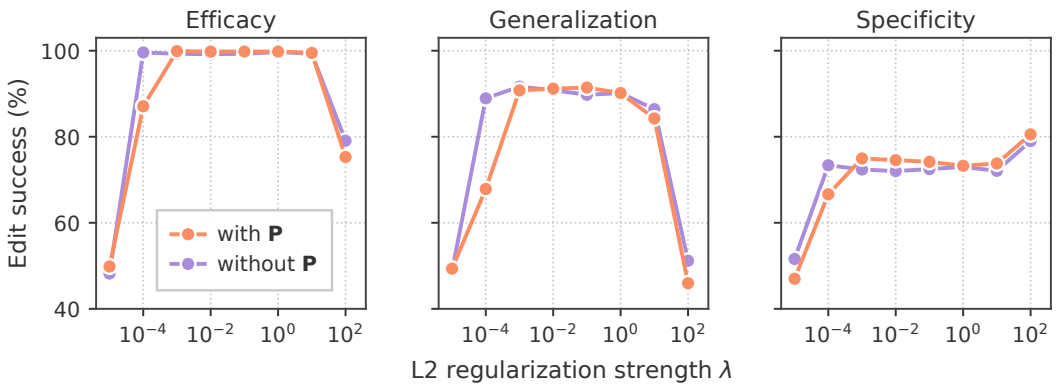

Figure 3: Edit success for AlphaEdit and projector-disabled variant on Llama-3-8B. Across most of the regularization range, the w/ and w/o $\mathbf{P}$ curves almost overlap on all three edit-quality metrics.

- Across backbones and benchmarks, FisherEdit achieves state-of-the-art performance while better preserving general abilities on downstream tasks and reducing representation- and parameter-space drift relative to prior methods.

2 Related Work

Knowledge editing (KE) updates factual associations in LLMs or reroutes queries to corrected knowledge, without full retraining or broad degradation of unrelated behavior (Yao et al., 2023; Wang et al., 2024b). Existing methods can be broadly grouped into learning-based editors and direct weight-editing methods.

Learning-based editors train auxiliary modules to produce, route, or store edits for a frozen base model. Representative methods include KnowledgeEditor (De Cao et al., 2021), MEND (Mitchell et al., 2022a), MALMEN (Tan et al., 2024), SERAC (Mitchell et al., 2022b) and RLEdit (Li et al., 2025). These methods require additional training, and their effectiveness depends on the editor's capacity and training distribution.

Direct weight-editing methods modify target-model weights directly. Knowledge Neurons (Dai et al., 2022) and ROME (Meng et al., 2022a) localize factual associations in transformer MLPs, while MEMIT (Meng et al., 2022b) extends this to batched least-squares editing. Subsequent work adds constraints to preserve prior knowledge (Gu et al., 2024; Ma et al., 2025; Zhang et al., 2025). Among projection-based methods, AlphaEdit (Fang et al., 2025) uses the null space of preserved activation keys; EvoEdit (Lyu et al., 2025) evolves it during editing; and KDE (Xu et al., 2025) decouples past and current knowledge through orthogonal projection. CrispEdit (Ikram et al., 2026) instead hard-thresholds a K-FAC curvature spectrum, restricting edits to the approximate null space of capability curvature. Such hard projections can remove both prior-knowledge and edit-relevant signals, trading efficacy for locality. FisherEdit instead uses a soft, element-wise Fisher penalty, retaining all update directions while constraining important parameters more strongly.

Connection to continual-learning regularization. Importance-weighted regularization is classical in continual learning: EWC (Kirkpatrick et al., 2017) penalizes movement of parameters with large diagonal Fisher, and SI (Zenke et al., 2017) and MAS (Aljundi et al., 2018) derive related online importance estimates. FisherEdit adapts this principle to least-squares editing, where updates are computed in closed form rather than through SGD. Its Fisher penalty is motivated by a local second-order approximation to predictive KL divergence (Sec. 4.4).

3 Rethinking Null-Space Projection

We now examine what drives preservation in AlphaEdit (Fang et al., 2025). After introducing its editing objective, we use controlled ablations to separate the effects of null-space projection and update regularization.

3.1 PRELIMINARIES

Following (Meng et al., 2022a;b), we edit the MLP down-projection matrix $\mathbf{W} \in \mathbb{R}^{d\times m}$. For a batch of n edits, let $\mathbf{K} \in \mathbb{R}^{m\times n}$ denote the edit keys, $\mathbf{V} \in \mathbb{R}^{d\times n}$ the target values. The goal is to find an update $\Delta \in \mathbb{R}^{d\times m}$ such that the edited weights map edit keys to the target values:

$$(\mathbf{W} + \Delta)\mathbf{K} \approx \mathbf{V}. \tag{1}$$

Equivalently, Δ should fit the residual left by the current weights:

$$\Delta\mathbf{K} \approx \mathbf{E}, \qquad \mathbf{E} \triangleq \mathbf{V} - \mathbf{W}\mathbf{K}. \tag{2}$$

In sequential editing, the update should also avoid changing previous edits and unrelated knowledge. Let $\mathbf{K}_p \in \mathbb{R}^{m\times n_p}$ denote historical edit keys from previous batches, $\mathbf{K}_1 \in \mathbb{R}^{m\times n_1}$ denote current edit keys, and $\mathbf{K}_0 \in \mathbb{R}^{m\times n_0}$ denote preserved keys sampled from a background corpus. A standard locate-then-edit objective is

$$\Delta = \arg\min_{\tilde{\Delta}} \underbrace{\|\tilde{\Delta}\mathbf{K}_1 - \mathbf{E}\|^2}_{(a)} + \underbrace{\|\tilde{\Delta}\mathbf{K}_p\|^2}_{(b)} + \underbrace{\|\tilde{\Delta}\mathbf{K}_0\|^2}_{(c)}. \tag{3}$$

Term (a) fits the new edits, while terms (b) and (c) penalize changes along historical-edit and preserved-knowledge directions, respectively.

AlphaEdit (Fang et al., 2025) replaces the penalty on $\mathbf{K}_0$ with a null-space projector $\mathbf{P}_\tau$. Specifically, $\mathbf{P}_\tau$ is computed from the eigenspectrum of $\mathbf{K}_0\mathbf{K}_0^\top$: directions whose eigenvalues exceed a threshold τ are treated as occupied, and the remaining subspace is used for editing. AlphaEdit solves

$$\Delta = \arg\min_{\tilde{\Delta}} \|\tilde{\Delta}\mathbf{P}_\tau\mathbf{K}_1 - \mathbf{E}\|^2 + \|\tilde{\Delta}\mathbf{P}_\tau\mathbf{K}_p\|^2 + \lambda\|\tilde{\Delta}\mathbf{P}_\tau\|^2. \tag{4}$$

The following analysis asks whether the projection $\mathbf{P}_\tau$, the regularization strength λ, or the threshold τ is responsible for AlphaEdit's empirical preservation behavior.

3.2 WHAT ACTUALLY DRIVES PRESERVATION?

Finding 1: The Projector $\mathbf{P}_\tau$ is near-identity. If the null-space mechanism is the main source of preservation, the projector should remove a substantial set of directions associated with prior knowledge. We first inspect the projector used by AlphaEdit under its default setting. As shown in Fig. 2(a), the matrix is almost identical to the identity: its diagonal entries are close to one, and its off-diagonal entries are negligible. This behavior comes from the spectrum of $\mathbf{K}_0\mathbf{K}_0^\top$. Fig. 2(b) shows that only a small number of singular directions exceed the default threshold τ, while the vast majority fall below it and are kept for editing. Thus, the projected-out subspace covers only a tiny fraction of the activation space. In practice, $\mathbf{P}_\tau$ leaves most update directions unchanged.

We then ask whether this near-identity projector affects editing outcomes. We compare AlphaEdit with a variant that disables the projector while keeping the same regularization strength. As shown in Fig. 3, the two variants largely overlap across Efficacy, Generalization, and Specificity metrics over the working range of λ. This suggests that, under the default setting, AlphaEdit's projector contributes little to its empirical preservation behavior.

Finding 2: Regularization is the main control knob. If the projector changes little, the remaining candidate explanation is the update regularizer. To isolate its effect, we keep the projector fixed and vary only the regularization strength λ. Fig. 4 shows that this single change produces large shifts across all edit-success metrics. With weak regularization, the update becomes unstable and performance collapses; with stronger regularization, performance is substantially restored. These results suggest that much of AlphaEdit's robustness comes from controlling update magnitude through regularization, with limited contribution from the default projection.

Finding 3: Stronger projection only moves along a trade-off. One possible explanation is that the default threshold τ makes the projector too weak. We therefore lower τ, forcing $\mathbf{P}_\tau$ to remove more activation directions. If the null-space interpretation were sufficient, a stronger projection should improve preservation without harming edit success. The results show a different pattern. As τ decreases, specificity improves, but efficacy and generalization drop sharply (Fig. 1). Thus, stronger projection does not recover a better protected subspace; it simply trades editing success for locality. This happens because the removed directions are not purely preserved-knowledge directions. They can also contain information needed to realize the edit. Removing them therefore protects some unrelated behavior, but weakens the edit itself.

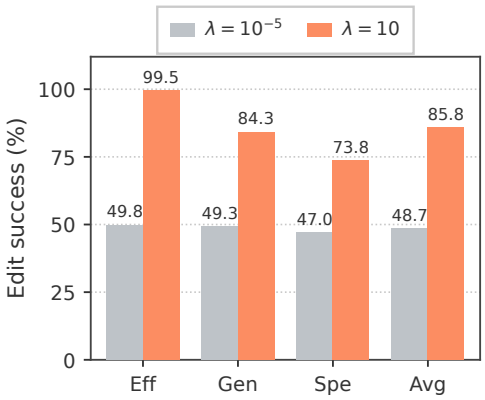

Figure 4: Within AlphaEdit on Llama-3-8B-Instruct, changing λ from 10^{-5} to 10 produces a large jump in all three metrics.

Takeaway. These findings suggest a different design principle for sequential editing: preservation should act on parameters, not entire activation directions. Hard projection can remove directions needed for the new edit, while uniform shrinkage ignores which weights matter most for general behavior. A better constraint should distinguish sensitive weights from safer ones. This motivates FisherEdit, which applies a Fisher-weighted, element-wise penalty directly to the update.

4 Method

4.1 Fisher-Guided Editing Objective

Motivated by the analysis above, our FisherEdit shifts preservation from activation-space projection to parameter-space regularization. We keep the standard edit-fitting objective, but penalize each parameter update according to its Fisher importance:

$$\Delta = \arg\min_{\tilde{\Delta}} \|\tilde{\Delta}\mathbf{K}_1 - \mathbf{E}\|^2 + \|\tilde{\Delta}\mathbf{K}_p\|^2 + \lambda\|\sqrt{\mathbf{F}} \odot \tilde{\Delta}\|^2, \tag{5}$$

Here, $\mathbf{F} \in \mathbb{R}^{d\times m}$ is the diagonal Fisher Information matrix and $\odot$ denotes element-wise multiplication. This term makes changes to high-Fisher parameters more costly while leaving low-Fisher parameters more flexible for encoding new facts. Compared with the objectives in Eqns. (3)(4), FisherEdit keeps the edit-fitting and historical-edit terms, but replaces the preserved-key penalty with a Fisher-weighted penalty on the update itself.

4.2 Solving the Fisher-Weighted Objective

Eqn. (5) is a quadratic objective that admits a closed-form solution. Because the Fisher penalty is element-wise, we can solve for each row of the update independently. Let $\mathbf{G} = \mathbf{K}_1\mathbf{K}_1^\top + \mathbf{K}_p\mathbf{K}_p^\top$, and let δ_i, e_i, and f_i denote the i-th rows of Δ, $\mathbf{E}$, and $\mathbf{F}$, respectively. With $\mathbf{D}_i = \mathrm{diag}(f_i)$, setting the gradient of Eqn. (5) to zero gives $\delta_i(\mathbf{G} + \lambda\mathbf{D}_i) = e_i\mathbf{K}_1^\top$ for each row, yielding

$$\delta_i = e_i\mathbf{K}_1^\top(\mathbf{G} + \lambda\mathbf{D}_i)^{-1}. \tag{6}$$

The shared matrix $\mathbf{G}$ captures the constraints from current and historical edit keys, while the row-specific $\mathbf{D}_i$ weights the update according to parameter importance.

Directly evaluating Eqn. (6) requires an $m \times m$ matrix inverse for each row. Since m is the MLP intermediate dimension, this becomes expensive at LLM scale. We therefore exploit the low-rank structure $\mathbf{G} = \mathbf{U}\mathbf{U}^\top$, where $\mathbf{U} = [\mathbf{K}_1 \mid \mathbf{K}_p] \in \mathbb{R}^{m\times r}$ and $r = n_1 + n_p$. Applying the Woodbury

identity (Golub & Van Loan, 2013) gives the equivalent update

$$\boxed{\delta_i = e_i \mathbf{K}_1^\top \Big[(\lambda \mathbf{D}_i)^{-1} - (\lambda \mathbf{D}_i)^{-1}\mathbf{U} \big(\mathbf{I}_r + \mathbf{U}^\top (\lambda \mathbf{D}_i)^{-1}\mathbf{U}\big)^{-1} \mathbf{U}^\top (\lambda \mathbf{D}_i)^{-1} \Big].} \tag{7}$$

Since $\mathbf{D}_i$ is diagonal, $(\lambda \mathbf{D}_i)^{-1}$ is computed by element-wise division. The remaining linear system has size $r \times r$, reducing the cost when the number of edit keys is much smaller than the MLP dimension ($r \ll m$). All d row-wise systems can be solved using batched linear algebra, with an overall complexity of $O(dmr^2 + dr^3)$. The full derivation is provided in Appendix G.

4.3 Estimating the Fisher Diagonal

The solver above assumes to the Fisher diagonal $\mathbf{F}$ for the edited weight block. We estimate this quantity once before editing and reuse it throughout the edit sequence. This keeps the editing procedure efficient while giving each parameter a fixed importance score measured from the model's original predictive behavior.

We use the empirical Fisher diagonal. Given a held-out calibration set $\mathcal{D} = \{(x_t, y_t)\}_{t=1}^N$, the Fisher value for parameter $W_{ij}^{(\ell)}$ in layer ℓ is estimated as

$$\mathbf{F}_{ij}^{(\ell)} = \frac{1}{N} \sum_{t=1}^{N} \left(\frac{\partial \mathcal{L}(x_t, y_t)}{\partial W_{ij}^{(\ell)}} \right)^2, \tag{8}$$

where $\mathcal{L}$ is the cross-entropy loss. We compute this estimate only for the MLP down-projection matrices that are later edited, rather than for all model parameters.

For calibration, we follow Šliogeris et al. (2025) and use N=1,000 examples sampled from MMLU (Hendrycks et al., 2020). The calibration data is separate from the edit requests and is used only to estimate parameter sensitivity. Each example is processed individually, so the squared gradients correspond to per-example empirical Fisher estimates rather than gradients averaged over a batch.

4.4 Theoretical Interpretation

In this section, we explain why the element-wise Fisher regularizer in Eqn. (5) provides an effective regularization principle for model editing. Specifically, we relate it to the locality cost of an update Δ, measured by the KL divergence between the pre-edit and post-edit predictive distributions on a reference set $\mathcal{D}_{\text{ref}}$:

$$\mathcal{L}_{\text{loc}}(\Delta) = \mathbb{E}_{x \sim \mathcal{D}_{\text{ref}}} \Big[\mathrm{KL}\big(p_{\mathbf{W}}(\cdot \,|\, x) \,\|\, p_{\mathbf{W}+\Delta}(\cdot \,|\, x)\big) \Big]. \tag{9}$$

To establish this connection, we consider a second-order expansion of $\mathcal{L}_{\text{loc}}$ around $\Delta = 0$. Under the regularity conditions in Appendix H.2, its value and gradient vanish at this point, and its Hessian equals the Fisher matrix $\mathbf{F}^{(S)}$ restricted to the edited parameters. Writing $u = \mathrm{vec}(\Delta)$, we obtain

$$\mathcal{L}_{\text{loc}}(\Delta) = \tfrac{1}{2}\, u^\top \mathbf{F}^{(S)} u + o(\|u\|^2). \tag{10}$$

The full $md \times md$ Fisher is intractable at LLM scale. Following Kirkpatrick et al. (2017), we use a diagonal approximation, estimated in practice by the empirical Fisher in Eqn. (8):

$$\tfrac{1}{2}\, u^\top \mathbf{F}^{(S)} u \;\approx\; \tfrac{1}{2} \sum_{i,j} F_{ij}\, \Delta_{ij}^2, \tag{11}$$

which recovers the form of the regularizer in Eqn. (5), with the constant absorbed into λ. FisherEdit therefore uses empirical parameter sensitivities to approximate the local cost of changing the pre-edit distribution, providing a tractable regularization principle for preserving model behavior. The full Taylor expansion and derivation are provided in Appendix H.

5 Experiments

We conduct comprehensive experiments to answer two core questions: (1) How does FisherEdit improve the overall performance of editing? (2) Does FisherEdit protect the general performance?

Table 1: CounterFact results across three backbones under sequential editing. Best per column in **bold**; second best underlined.

	Llama-3-8B-Instruct				Qwen2.5-7B-Instruct				Olmo-3-7B-Instruct			
Method	**Eff**	**Gen**	**Spe**	**Avg**	**Eff**	**Gen**	**Spe**	**Avg**	**Eff**	**Gen**	**Spe**	**Avg**
FT	65.02	58.05	41.63	54.90	73.23	62.76	40.52	58.84	95.17	70.87	56.12	74.05
ROME	33.84	36.51	68.16	46.17	37.33	40.92	65.52	47.92	36.10	39.12	69.32	48.18
MEMIT	48.00	47.64	52.52	49.39	64.20	62.41	55.88	60.83	71.38	69.53	55.45	65.45
MEND	48.41	48.41	51.05	49.29	1.13	0.67	1.15	0.98	38.97	43.13	62.18	48.09
GRACE	21.84	26.15	**78.13**	42.04	37.23	41.94	64.55	47.91	39.17	42.00	64.09	48.42
RECT	33.94	36.87	70.05	46.95	36.20	40.61	65.87	47.56	36.71	39.89	68.39	48.33
PRUNE	35.48	37.48	67.88	46.95	37.33	40.92	65.52	47.92	36.00	38.71	69.47	48.06
WISE	43.59	42.31	61.55	49.15	44.21	45.23	61.76	50.40	39.28	42.10	63.80	48.39
AnyEdit	52.00	47.33	51.72	50.35	69.53	48.05	56.34	57.97	63.48	51.28	48.29	54.35
RLEdit	24.20	27.43	75.63	42.42	35.79	38.20	66.42	46.80	22.25	29.94	**75.63**	42.61
UltraEdit	52.41	53.12	46.77	50.77	60.61	42.05	**69.57**	57.41	43.38	45.33	57.57	48.76
AlphaEdit	99.79	90.15	73.25	87.73	**100.00**	64.15	66.36	76.84	**99.48**	75.84	72.76	82.69
FisherEdit (ours)	**99.90**	**90.92**	75.68	**88.83**	99.90	**71.08**	66.66	**79.21**	99.38	**79.53**	73.33	**84.08**

Table 2: Main results across CounterFact, Ripple, and EVOKE on Olmo-3-7B-Instruct under sequential editing. Best per column in **bold**; second best underlined.

	CounterFact				Ripple				EVOKE			
Method	**Eff**	**Gen**	**Spe**	**Avg**	**Eff**	**Gen**	**Spe**	**Avg**	**Eff**	**Gen**	**Spe**	**Avg**
FT	95.17	70.87	56.12	74.05	76.10	59.87	63.79	66.59	94.10	79.48	47.70	73.76
ROME	36.10	39.12	69.32	48.18	42.60	44.01	67.30	51.30	33.50	34.22	65.58	44.43
MEMIT	71.38	69.53	55.45	65.45	48.40	45.99	69.85	54.75	64.80	63.81	51.16	59.92
MEND	38.97	43.13	62.18	48.09	32.90	34.94	74.62	47.48	37.20	37.32	63.73	46.08
GRACE	39.17	42.00	64.09	48.42	50.20	48.39	60.89	53.16	35.40	35.80	63.52	44.91
RECT	36.71	39.89	68.39	48.33	41.00	41.91	68.93	50.61	34.00	34.42	65.66	44.69
PRUNE	36.00	38.71	69.47	48.06	43.80	43.93	67.17	51.63	33.30	34.21	65.59	44.37
WISE	39.28	42.10	63.80	48.39	56.20	56.71	55.06	55.99	36.00	35.96	62.77	44.91
AnyEdit	63.48	51.28	48.29	54.35	50.80	49.85	**82.60**	61.08	68.10	58.90	40.54	55.85
RLEdit	22.25	29.94	**75.63**	42.61	6.90	10.03	82.55	33.16	17.10	18.83	**78.51**	38.15
UltraEdit	43.38	45.33	57.57	48.76	48.00	49.04	57.03	51.36	48.90	48.88	51.19	49.66
AlphaEdit	**99.48**	75.84	72.76	82.69	96.70	**75.10**	33.39	68.40	**99.90**	92.47	70.14	87.50
FisherEdit (ours)	99.38	**79.53**	73.33	**84.08**	**97.30**	74.66	42.28	**71.41**	**99.90**	**92.94**	71.08	**87.97**

5.1 Experiment Setup

Datasets and Models. We evaluate FisherEdit on three widely used model editing benchmarks: CounterFact (Meng et al., 2022a), Ripple (Cohen et al., 2024), and EVOKE (Zhang et al., 2025). Experiments are conducted under a sequential editing setting with 1,000 edits, using three representative backbones: Llama-3-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Qwen et al., 2025), and Olmo-3-7B-Instruct (Olmo et al., 2025). To assess the impact on general capabilities, we further evaluate the edited models on HumanEval (Chen et al., 2021), ARC-Challenge/Easy (Clark et al., 2018), IFEval (Zhou et al., 2023), WinoGrande (Sakaguchi et al., 2021) and GPQA (Rein et al., 2023), and report WikiText (Merity et al., 2016) perplexity after sequential editing on CounterFact.

Baselines and Metrics. We compare FisherEdit with twelve baselines, including the learning-based and direct weight-editing methods discussed in Section 2, as well as FT (Zhu et al., 2020), GRACE (Hartvigsen et al., 2023), WISE (Wang et al., 2024a), AnyEdit (Jiang et al., 2025), and UltraEdit (Gu et al., 2025). For editing performance, we report efficacy (Eff), generalization (Gen), specificity (Spe), and their unweighted average (Avg). For general performance, we report task accuracy and WikiText perplexity.

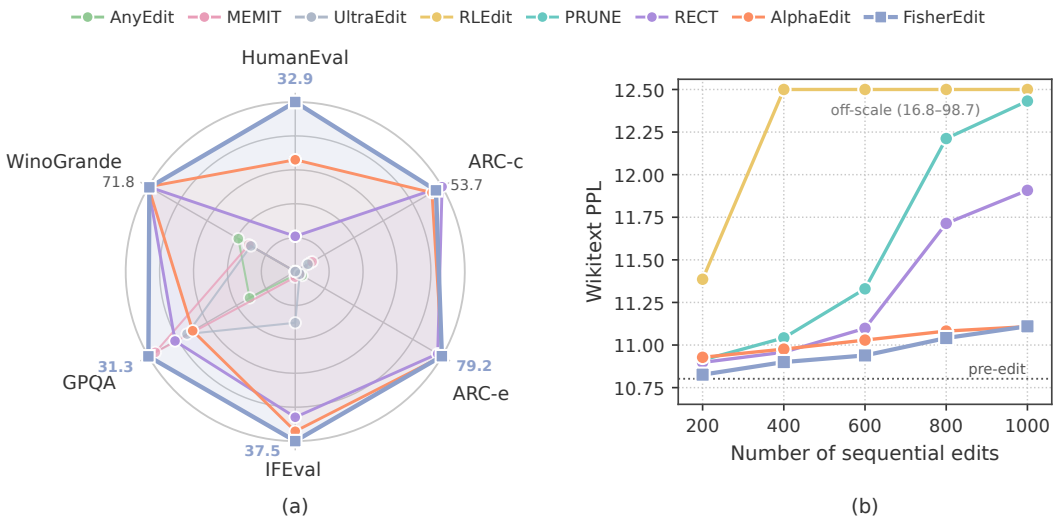

Figure 5: (a) Downstream task scores of Llama-3-8B-Instruct on HumanEval, ARC-c, ARC-e, IFEval, WinoGrande and GPQA. (b) Wikitext perplexity of Llama-3-8B-Instruct after applying $N \in \{200, 400, 600, 800, 1000\}$ CounterFact edits.

5.2 Editing Performance

Table 1 reports CounterFact results across three backbones. FisherEdit achieves the highest Avg in all three settings, with gains over AlphaEdit of $+1.10$ on Llama, $+2.37$ on Qwen, and $+1.39$ on Olmo. Since Efficacy is already close to saturated for both AlphaEdit and FisherEdit, the main differences appear in Gen and Spe. This indicates that FisherEdit improves the quality of the update primarily by better balancing paraphrase transfer and neighborhood preservation.

Table 2 further evaluates whether the same trend holds across datasets on a fixed backbone. On Olmo-3-7B-Instruct, FisherEdit obtains the best Avg on CounterFact, Ripple, and EVOKE. The improvements over AlphaEdit are consistent: $+1.39$, $+3.01$, and $+0.47$ points, respectively. These results support the claim that Fisher-weighted regularization provides a robust preservation mechanism across both model and benchmark variations.

We also evaluate FisherEdit's computational efficiency by measuring wall-clock time and peak GPU memory across the editing pipeline, with a detailed comparison to AlphaEdit in Appendix C.5. The results show that FisherEdit substantially reduces preprocessing time and overall runtime including preprocessing, while using nearly the same peak GPU memory.

5.3 General Performance

General domains. Preserving general capabilities beyond the edited facts is crucial, yet is often overlooked by prior editing methods. From Figure 5(a), FisherEdit maintains stable performance across all six downstream tasks, covering code generation, science, commonsense reasoning, instruction following, and challenging knowledge-intensive question answering. In contrast, AlphaEdit and RECT exhibit noticeable drops on HumanEval and GPQA. These results suggest that the Fisher-diagonal regularizer helps mitigate collateral damage while still enabling effective edits.

Language modeling. We further evaluate the impact of editing on general language modeling by reporting WikiText perplexity after sequentially editing $N \in \{200, 400, 600, 800, 1000\}$ CounterFact samples. As shown in Figure 5(b), FisherEdit better preserves general text modeling ability than competing methods during sequential editing. Its WikiText perplexity remains consistently low across all edit counts, increasing only mildly from 10.826 after 200 edits to 11.110 after 1,000 edits. By contrast, AlphaEdit, RECT, and PRUNE show larger increases. These results indicate that FisherEdit maintains the most stable general-language perplexity throughout the editing process.

5.4 Ablation Study

Fisher Matrix. To examine whether the Fisher diagonal captures model-specific information, we replace $\mathbf{F}$ in Eq. (5) with an all-ones matrix (*Uniform*), which reduces the objective to ℓ_2 regularization, or i.i.d. samples from $\mathrm{Uniform}[0, 1]$ (*Random*), while keeping all other hyperparameters fixed. From Table 3, replacing $\mathbf{F}$ with *Uniform* largely preserves rewrite and paraphrase performance, which are mainly controlled by λ as discussed in §3, but substantially weakens neighborhood locality. Meanwhile, *Random* degrades all three metrics. These results indicate that FisherEdit benefits from the structured information encoded in the Fisher diagonal, rather than from generic non-uniform reweighting.

Table 3: Fisher-matrix ablation on CounterFact with Llama-3-8B-Instruct under sequential editing. Fisher diagonal $\mathbf{F}$ is replaced with an all-ones matrix (*Uniform*) or with i.i.d. uniform random values (*Random*).

	Eff	Gen	Spe	Avg
Fisher	**99.58**	91.28	**75.13**	**88.66**
Uniform	99.28	**91.48**	72.22	87.66
Random	95.89	85.74	69.94	83.86

Top-Fisher masking. The Fisher diagonal is heavy-tailed and model-dependent: in layer 4, its largest 1% of cached entries carry 16.8%, 52.6%, and 30.4% of the total mass for Llama-3-8B-Instruct, Qwen2.5-7B-Instruct, and Olmo-3-7B-Instruct, respectively (a uniform matrix would carry 1%). To test whether the high-Fisher coordinates are important for preservation, we rerun the Llama-3-8B-Instruct/CounterFact setting with 1,000 sequential edits while selectively removing or retaining the highest-Fisher coordinates. For *mask-top*, entries at the top q percent are replaced by that layer's median; for *keep-top*, only those entries are retained and all others are replaced by the median. As shown in Table 4, masking the top 1% decreases Specificity by 1.65 points, eliminating approximately 48% of the Full-Fisher–Uniform Specificity gap and 63% of the Average gap. Conversely, keeping only the top 10% recovers approximately 60% of the Specificity advantage and 70% of the Average advantage, while the full diagonal remains strongest. These results provide direct evidence that the proposed Fisher diagonal plays a fundamental role in FisherEdit's gains, beyond uniform ℓ_2 regularization alone.

Table 4: Effect of masking or retaining high-Fisher coordinates.

	Eff	Gen	Spe	Avg
Full Fisher	**99.90**	90.92	**75.68**	**88.83**
Mask top 1%	99.49	90.87	74.03	88.13
Mask top 10%	99.38	91.03	73.50	87.97
Mask top 30%	99.49	89.54	74.33	87.79
Keep top 10%	99.69	**91.49**	74.29	88.49
Uniform	99.90	91.02	72.22	87.71

Table 5: Ablation on the number of MMLU examples n used to estimate the Fisher diagonal.

n	Eff	Gen	Spe	Avg
50	99.79	89.84	74.18	87.94
100	99.69	**92.15**	73.48	88.44
500	99.79	91.12	75.10	**88.67**
1000	99.58	91.28	**75.13**	88.66
2000	**99.79**	90.76	74.66	88.40

Number of Calibration Examples. We vary the number of examples n used to estimate the Fisher diagonal while keeping the editing configuration fixed. As shown in Table 5, FisherEdit is largely insensitive to the estimation sample size: the average score remains within a narrow range, from 87.94 to 88.67, and the neighborhood score changes only mildly. Even with only $n{=}50$ examples, the Fisher constraint retains most of its benefit, suggesting that the regularizer does not require a large reference set to provide effective protection.

6 Conclusion

In this work, we addressed parameter interference and capability degradation in sequential knowledge editing with FisherEdit, a Fisher-guided framework that replaces coarse activation-space projection with importance-aware parameter regularization and computes updates through an efficient row-wise closed-form solver. Theoretical analysis and empirical results show that FisherEdit reduces distributional drift and achieves state-of-the-art editing performance while better preserving general capabilities across backbones and benchmarks.

AI USE STATEMENT

AI assistants were used only for language polishing, LaTeX checking, and coding/debugging assistance. All scientific claims, experiments, analyses, and final writing were verified and controlled by the authors.

REPRODUCIBILITY STATEMENT

The editing algorithm, datasets, evaluation metrics, and hyperparameters are specified in Appendices A and B. Appendices G and H provide the derivations, and Appendix C.1 reports results across random seeds.

REFERENCES

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European conference on computer vision*, pp. 144–161. Springer, 2018.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

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

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, 2022.

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*, pp. 6491–6506, 2021.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained knowledge editing for language models. In *International Conference on Learning Representations*, volume 2025, pp. 16366–16396, 2025.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International conference on machine learning*, pp. 10323–10337. PMLR, 2023.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819, 2024.

Xiaojie Gu, Guangxu Chen, Jungang Li, Jia-Chen Gu, Xuming Hu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in large language models. *arXiv e-prints*, pp. arXiv–2505, 2025.

Akshat Gupta, Phudish Prateepamornkul, Maochuan Lu, Ahmed Alaa, Thomas Hartvigsen, and Gopala Anumanchipalli. Lifelong knowledge editing requires better regularization. *arXiv preprint arXiv:2502.01636*, 2025.

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959, 2023.

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

Zarif Ikram, Arad Firouzkouhi, Stephen Tu, Mahdi Soltanolkotabi, and Paria Rashidinejad. Crispedit: Low-curvature projections for scalable non-destructive llm editing. *arXiv preprint arXiv:2602.15823*, 2026.

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 *International Conference on Machine Learning*, pp. 27510–27533. PMLR, 2025.

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

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *International Conference on Machine Learning*, pp. 34920–34942. PMLR, 2025.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6: 87–100, 2024.

Qingyuan Liu, Jia-Chen Gu, Yunzhi Yao, Hong Wang, and Nanyun Peng. Energy-regularized sequential model editing on hyperspheres. *arXiv preprint arXiv:2510.01172*, 2025.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7052–7063, 2021.

Sicheng Lyu, Yu Gu, Xinyu Wang, Jerry Huang, Sitao Luan, Yufei Cui, Xiao-Wen Chang, and Peng Lu. Evoedit: Evolving null-space alignment for robust and efficient knowledge editing. *arXiv preprint arXiv:2510.13851*, 2025.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *International Conference on Learning Representations*, volume 2025, pp. 17302–17329, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pp. 15817–15831. PMLR, 2022b.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

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)*, pp. 2463–2473, 2019.

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.

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

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)*, pp. 5418–5426, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Vytenis Šliogeris, Povilas Daniušis, and Artūras Nakvosas. Full-parameter continual pretraining of gemma2: Insights into fluency and domain knowledge. *arXiv preprint arXiv:2505.05946*, 2025.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *International Conference on Learning Representations*, volume 2024, pp. 54676–54690, 2024.

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. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024a.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37, 2024b.

Xiaobao Wu, Liangming Pan, William Yang Wang, and Luu Anh Tuan. Akew: Assessing knowledge editing in the wild. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15118–15133, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, volume 2024, pp. 35623–35646, 2024.

Haoyu Xu, Pengxiang Lan, Enneng Yang, Guibing Guo, Jianzhe Zhao, Linying Jiang, and Xingwei Wang. Knowledge decoupling via orthogonal projection for lifelong editing of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13194–13213, 2025.

Wanli Yang, Fei Sun, Xinyu Ma, Xun Liu, Dawei Yin, and Xueqi Cheng. The butterfly effect of model editing: Few edits can trigger large language models collapse. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5419–5437, 2024.

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*, pp. 10222–10240, 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. Pmlr, 2017.

Chi Zhang, Mengqi Zhang, Xiaotian Ye, Runxi Cheng, Zisheng Zhou, Ying Zhou, Pengjie Ren, and Zhumin Chen. Spectral characterization and mitigation of sequential knowledge editing collapse. *arXiv preprint arXiv:2601.11042*, 2026.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Zhumin Chen, et al. Uncovering overfitting in large language model editing. In *International Conference on Learning Representations*, volume 2025, pp. 79550–79568, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*, 2020.

A FisherEdit Algorithm Flow

Algorithm 1 FisherEdit sequential editing

Require: pretrained model $\mathbf{W}$, edited layers $\mathcal{L}$, edit stream $\{\mathcal{B}_t\}_{t=1}^{T}$, diagonal Fisher $\mathbf{F}$, regularization strength λ

Initialize historical key memory $\mathbf{K}_p^{(\ell)} \leftarrow \varnothing$ for each layer $\ell \in \mathcal{L}$
for $t = 1$ to T **do**
  Format the current batch prompts and compute the target representation $z^\star$ for each edit request
  for each layer $\ell \in \mathcal{L}$ **do**
    Trace the current lookup keys $\mathbf{K}_1^{(\ell)}$ and compute the remaining residual $\mathbf{E}^{(\ell)}$
    Retrieve historical keys $\mathbf{K}_p^{(\ell)}$ from memory
    Form $\mathbf{U}^{(\ell)} = [\mathbf{K}_1^{(\ell)}, \mathbf{K}_p^{(\ell)}]$ and $\mathbf{E}'^{(\ell)} = [\mathbf{E}^{(\ell)}, \mathbf{0}]$
    Solve the Fisher-weighted row-wise update $\Delta^{(\ell)}$ by Eqn. (7)
    Update the layer weights: $\mathbf{W}^{(\ell)} \leftarrow \mathbf{W}^{(\ell)} + \Delta^{(\ell)}$
    Append the current batch keys to memory: $\mathbf{K}_p^{(\ell)} \leftarrow [\mathbf{K}_p^{(\ell)}, \mathbf{K}_1^{(\ell)}]$
  end for
end for
return the edited model

B Experimental Details

B.1 Evaluation Metrics

For an edit request $(s_i, o_i) \to o_i^*$, where s_i is the subject, o_i is the original object, and o_i^* is the edited target object, let q_i^{edit} denote the rewrite prompt used to state the edited fact, $\mathcal{Q}_i^{\text{para}}$ the set of paraphrase prompts that express the same edited fact, and $\mathcal{Q}_i^{\text{loc}}$ the set of neighborhood prompts used to test preserved knowledge. After applying the edit sequence, we evaluate all metrics with the post-edit model $p_{W+\Delta}$.

For a prompt q and a candidate answer $a = (a_1, \ldots, a_T)$, define the average target negative log-likelihood

$$\ell_\Delta(a \mid q) = -\frac{1}{T}\sum_{t=1}^{T} \log p_{\mathbf{W}+\Delta}(a_t \mid q, a_{<t}). \tag{12}$$

Lower $\ell_\Delta(a \mid q)$ means that the edited model assigns higher probability to answer a under prompt q.

Efficacy. Efficacy measures whether the edited model recalls the new target on the original rewrite prompt. A request is counted as successful when the new answer has lower average NLL than the original answer:

$$\begin{aligned} e_i &= \mathbf{1}\left[\ell_\Delta(o_i^* \mid q_i^{\text{edit}}) < \ell_\Delta(o_i \mid q_i^{\text{edit}})\right], \\ \text{Eff} &= \frac{1}{n}\sum_{i=1}^{n} e_i. \end{aligned} \tag{13}$$

Generalization. Generalization measures whether the same update transfers to paraphrased prompts. For each request, we average the same new-versus-original comparison over its paraphrase set:

$$\begin{aligned} g_i(q) &= \mathbf{1}[\ell_\Delta(o_i^* \mid q) < \ell_\Delta(o_i \mid q)], \\ \text{Gen} &= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|\mathcal{Q}_i^{\text{para}}|} \sum_{q \in \mathcal{Q}_i^{\text{para}}} g_i(q). \end{aligned} \tag{14}$$

Specificity. Specificity measures whether unrelated or neighboring facts are preserved. Let $o_i^{\text{loc}}(q)$ be the correct preserved answer for a neighborhood prompt q. A neighborhood prompt is counted as

successful when the preserved answer remains more likely than the edit target:

$$s_i(q) = \mathbf{1}\left[\ell_\Delta(o_i^{\mathrm{loc}}(q) \mid q) < \ell_\Delta(o_i^* \mid q)\right],$$
$$\mathrm{Spe} = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|\mathcal{Q}_i^{\mathrm{loc}}|} \sum_{q \in \mathcal{Q}_i^{\mathrm{loc}}} s_i(q). \tag{15}$$

B.2 Dataset Details

Main editing datasets. CounterFact (Meng et al., 2022a) is a factual knowledge editing benchmark in which each request rewrites a subject–relation association from an original object to a counterfactual target object. Each record also provides paraphrase prompts for testing whether the edited fact generalizes beyond the original wording, as well as neighborhood prompts for testing whether unrelated facts are preserved. We use the AKEW (Wu et al., 2024) version of CounterFact and remove records whose aligned AKEW annotations are unavailable, leaving 975 valid edit requests. RippleEdits (Cohen et al., 2024) evaluates whether an edit has appropriate ripple effects on related queries while preserving other relation-specific knowledge about the edited subject. In our experiments, we use the `random` subset of RippleEdits and evaluate the first 1,000 records. EVOKE (Zhang et al., 2025) is designed to evaluate overfitting in model editing by combining direct rewrite prompts with augmented paraphrase prompts and locality prompts. We use the `main` subset of EVOKE and evaluate the first 1,000 records. All main editing results are reported under sequential editing with 100 edit requests per batch. The edited model is not reset between batches; after processing the full edit stream, we evaluate the final edited model on all requests.

Downstream datasets. To measure whether editing damages general capabilities beyond the edited facts, we evaluate the edited models on five downstream benchmarks. HumanEval (Chen et al., 2021) evaluates code generation by asking models to complete Python programming problems, and we report pass@1. ARC-Challenge and ARC-Easy (Clark et al., 2018) evaluate multiple-choice science question answering, with ARC-Challenge containing the harder subset and ARC-Easy containing the easier subset; we report normalized accuracy. IFEval (Zhou et al., 2023) evaluates instruction-following behavior by checking whether model outputs satisfy verifiable prompt-level constraints, and we report strict prompt accuracy. WinoGrande (Sakaguchi et al., 2021) evaluates commonsense reasoning through adversarial Winograd-style pronoun resolution examples, and we report accuracy. GPQA (Rein et al., 2023) evaluates difficult graduate-level, Google-proof multiple-choice questions; we use the Diamond zero-shot setting and report normalized accuracy. We additionally report WikiText (Merity et al., 2016) perplexity to measure general language-modeling quality after sequentially applying CounterFact edits.

B.3 Hyperparameter Configurations

All experiments use instruction-tuned backbones: Llama-3-8B-Instruct, Qwen2.5-7B-Instruct, and Olmo-3-7B-Instruct. Models are loaded with HuggingFace `AutoModelForCausalLM` and evaluated with their corresponding tokenizers. Because the backbones are instruction-tuned, both editing prompts and evaluation prompts are serialized with each tokenizer's chat template. We run the entire editing pipeline in fp32, including offline precomputation, and use bf16 for downstream inference. We edit the MLP down-projection matrices `model.layers.{}.mlp.down_proj` at layers 4–8, use the last subject token as the fact lookup token, and optimize target values for 25 gradient steps with learning rate 0.1 and weight decay 0.5. The target loss layer to calculate $\mathbf{V}$ is 31 for Llama-3-8B-Instruct and Olmo-3-7B-Instruct, and 27 for Qwen2.5-7B-Instruct. We set $\lambda = 10^{-3}$ across all benchmarks and backbones. For Fisher diagonal estimation, we estimates the empirical Fisher diagonal once from 1,000 MMLU examples, computing per-example squared cross-entropy gradients only for the edited MLP down-projection weights. The resulting Fisher tensors are cached and reused throughout the sequential edit stream.

C Additional Experimental Results

C.1 Seed Variance and Significance Testing

Table 1 reports single runs under the benchmark's native edit order. To quantify run-to-run variability, we repeat the experiment with three random seeds for both FisherEdit and AlphaEdit. Each seed reshuffles the edit stream, i.e. the composition and ordering of the sequential batches, and reseeds all stochastic components including the target-value optimization. Both methods see exactly the same cases under matched seeds, so the comparison is paired.

Table 6: Three-seed comparison on CounterFact with Llama-3-8B-Instruct. Mean ± standard deviation over seeds; each seed reshuffles the sequential edit stream.

Method	Eff	Gen	Spe	Avg
AlphaEdit	99.69 ± 0.10	90.17 ± 0.75	73.75 ± 0.30	87.87 ± 0.28
FisherEdit	**99.79** ± 0.21	**92.68** ± 0.70	**73.91** ± 0.16	**88.80** ± 0.28

The largest gap is in Generalization, where FisherEdit leads by 2.51 points on average. A two-sided Welch's t-test on the three seed-level Generalization scores gives $p = 0.013$. The Average advantage is also consistent rather than seed-specific: FisherEdit is higher in all three paired runs, by $+0.76$ to $+1.03$ points (mean $+0.93$). Efficacy is near saturation for both methods and Specificity is comparable, so the seed-level picture matches the single-run trend in §5.2.

C.2 Results on Mistral-7B-Instruct

To check that the conclusion is not tied to the three backbones of §5, we extend the evaluation on Mistral-7B-Instruct-v0.3 on both CounterFact and Ripple, against the six baselines from Tables 1 and 2.

Table 7: Sequential editing on Mistral-7B-Instruct-v0.3.

	CounterFact				Ripple			
Method	**Eff**	**Gen**	**Spe**	**Avg**	**Eff**	**Gen**	**Spe**	**Avg**
MEMIT	50.26	50.26	49.74	50.09	53.80	53.58	49.49	52.29
RECT	29.85	31.54	75.46	45.61	31.50	32.38	**76.15**	46.68
PRUNE	32.41	32.36	71.19	45.32	32.50	33.81	73.24	46.52
RLEdit	48.00	48.26	52.91	49.72	49.30	47.19	55.62	50.70
UltraEdit	54.26	54.15	45.83	51.41	67.60	69.38	34.58	57.19
AlphaEdit	**99.90**	89.18	76.91	88.66	**96.40**	**76.36**	45.67	72.81
FisherEdit (ours)	**99.90**	**90.10**	**77.12**	**89.04**	95.70	75.99	51.12	**74.27**

FisherEdit attains the best Average on both benchmarks. On CounterFact it matches AlphaEdit's saturated Efficacy while improving Generalization and Specificity (88.66 → 89.04 Avg). On Ripple it trades a small amount of rewrite and paraphrase accuracy for a 5.45-point Specificity gain, giving $+1.46$ Avg. Overall, it extends the backbone-level consistency of our conclusion to four instruction-tuned model families.

C.3 AlphaEdit under Matched Regularization Strength

To verify that FisherEdit's gain does not come merely from using a weaker regularization strength than AlphaEdit's published $\ell_2 = 1$, we sweep AlphaEdit over $\ell_2 \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$ on all three backbones in Table 8.

Three orders of magnitude in ℓ_2 move AlphaEdit's Average by at most 0.81 points, and the best setting is model-dependent. FisherEdit stays ahead of every tested coefficient, by $+0.29$, $+1.89$, and $+0.98$ Average points on Llama, Qwen, and Olmo. Tuning a scalar strength therefore does not reproduce what the Fisher diagonal provides.

Table 8: AlphaEdit with swept ℓ_2 strength versus FisherEdit on CounterFact.

Backbone	Method	ℓ_2	Eff	Gen	Spe	Avg
Llama	AlphaEdit	1	99.79	90.15	73.25	87.73
		10^{-1}	99.79	91.44	74.17	88.47
		10^{-2}	99.79	91.18	74.56	88.51
		10^{-3}	99.90	90.77	74.95	88.54
	FisherEdit	10^{-3}	99.90	90.92	75.68	**88.83**
Qwen	AlphaEdit	1	100.00	64.15	66.36	76.84
		10^{-1}	99.79	65.79	66.38	77.32
		10^{-2}	100.00	65.08	66.19	77.09
		10^{-3}	99.28	65.44	62.55	75.76
	FisherEdit	10^{-3}	99.90	71.08	66.66	**79.21**
Olmo	AlphaEdit	1	99.48	75.84	72.76	82.69
		10^{-1}	99.59	76.46	72.87	82.97
		10^{-2}	99.59	76.41	72.95	82.98
		10^{-3}	99.59	76.62	73.08	83.10
	FisherEdit	10^{-3}	99.38	79.53	73.33	**84.08**

C.4 Fisher Drift and Online Fisher Updating

FisherEdit estimates the Fisher diagonal once before editing and reuses it throughout the edit stream. Because the edited weights change during sequential editing, parameter importance could in principle drift, and re-estimating the Fisher online might then help. We examine both questions under the Llama-3-8B-Instruct / CounterFact setting of Table 1 with 1,000 sequential edits.

Parameter-importance drift. We save the model after 1,000 sequential edits and re-estimate its Fisher diagonal on the same held-out calibration set used before editing. We then compare the pre- and post-editing diagonals at each edited layer using log-Pearson and Spearman correlation, cosine similarity, and the overlap of the top-1% and top-10% coordinate sets.

Table 9: Agreement between the pre- and post-editing Fisher diagonals after 1,000 sequential edits, per edited layer.

Layer	Pearson (log)	Spearman	Cosine	Top-1% overlap	Top-10% overlap
4	0.878	0.874	0.824	0.61	0.73
5	0.871	0.859	0.947	0.61	0.72
6	0.883	0.864	0.979	0.65	0.72
7	0.911	0.893	0.981	0.71	0.75
8	0.913	0.897	0.919	0.71	0.75

As Table 9 shows, the importance structure remains strongly aligned after 1,000 edits: Spearman correlation is 0.86–0.90, log-Pearson correlation is 0.87–0.91, cosine similarity is 0.82–0.98, and 72–75% of the highest-Fisher 10% of parameters remain in that set. Both the ranking and the membership of the high-Fisher parameter sets are therefore largely stable under sequential editing with FisherEdit.

Online Fisher variant. To test whether online re-estimation nevertheless helps, we implement a variant that updates the Fisher after every edit batch. Let $\mathbf{F}_t$ be the Fisher used for batch t. After applying the edits of batch t, we run the updated model on the just-edited prompt/target pairs, compute per-example squared cross-entropy gradients for the edited down-projection weights, and average them into a local diagonal $\mathbf{F}_{\text{local},t}$. The Fisher for the next batch is

$$\mathbf{F}_{t+1} = \gamma\, \mathbf{F}_t + (1-\gamma)\, \mathbf{F}_{\text{local},t}, \tag{16}$$

where $\gamma \in [0, 1]$ controls how quickly the estimate adapts. Setting $\gamma = 1$ exactly recovers the fixed Fisher used in the paper, while smaller γ gives more weight to the newly computed local Fisher. All other editing settings are unchanged.

Table 10: Fixed versus online Fisher on CounterFact with Llama-3-8B-Instruct. Extra time is the additional wall-clock cost per 100-edit batch relative to the fixed Fisher.

Fisher	Eff	Gen	Spe	Avg	Extra time / batch
Fixed ($\gamma = 1$)	**99.90**	**90.92**	**75.68**	**88.83**	0
Online ($\gamma = 0.99$)	99.59	90.21	75.14	88.31	$\sim$95 s
Online ($\gamma = 0.9$)	99.79	90.56	74.87	88.41	$\sim$97 s

Table 10 shows that online updating provides no improvement: both online variants fall slightly below the fixed Fisher on all four metrics, with Average decreasing by 0.42–0.52 points. They also require an additional forward and backward pass over every edit batch, adding roughly 95–97 seconds per 100 edits, about 45% of the per-batch editing time reported in Table 11. Together with the limited drift in Table 9, these results support the fixed Fisher as the more effective and more efficient choice.

C.5 Computational Cost

We instrument every phase of the pipeline and report wall-clock time and peak allocated GPU memory on a single RTX 6000 Ada, under the same setting of Table 1.

Table 11: Phase-wise computational cost relative to AlphaEdit.

Phase	AlphaEdit	FisherEdit	Diff.
Wall-clock time			
Preprocessing (min)	648.0	**17.7**	-97.3%
Editing / 100 edits (s)	**195.7**	213.0	$+8.9\%$
Evaluation (s)	328.4	**318.7**	-3.0%
End-to-end (min)	680.4	**53.4**	-92.1%
Peak memory			
Peak GPU memory (GiB)	35.19	**34.94**	-0.7%

FisherEdit substantially lowers the cost of constructing the preservation constraint. AlphaEdit requires second-moment computation over 100k Wikipedia samples and layerwise SVD, taking 10.80 hours in our experiment. FisherEdit estimates a diagonal Fisher matrix once from 1,000 MMLU examples and completes preprocessing in 17.70 minutes. Thus, FisherEdit reduces preprocessing time by $36.6\times$ and end-to-end runtime by $12.8\times$ for 1,000 sequential edits, without introducing inference-time overhead.

The trade-off is an 8.9% increase in per-batch editing time (195.7 versus 213.0 seconds). This overhead is caused by the Fisher-weighted row-wise solve, while the Woodbury formulation keeps the computation efficient and peak GPU memory nearly unchanged.

D Hidden-Space Drift Analysis

To better understand how FisherEdit changes internal representations, we visualize the hidden-state movement induced by FisherEdit and AlphaEdit. We use the EVOKE sequential editing run on Olmo-3-7B-Instruct and select the first and last edited layers, layers 4 and 8. For each edit request, we record the last-token hidden state of the rewrite prompt and ten neighborhood prompts before and after editing. Each point in Figure 6 corresponds to one edit request: the x-axis measures the ℓ_2 movement of the rewrite prompt, while the y-axis measures the average ℓ_2 movement of its neighborhood prompts. Lower neighborhood movement indicates less collateral change in nearby but unrelated contexts.

At layer 4, FisherEdit produces a clearly smaller movement than AlphaEdit, indicating that the Fisher-weighted constraint already limits unintended representation drift at the first edited layer. At

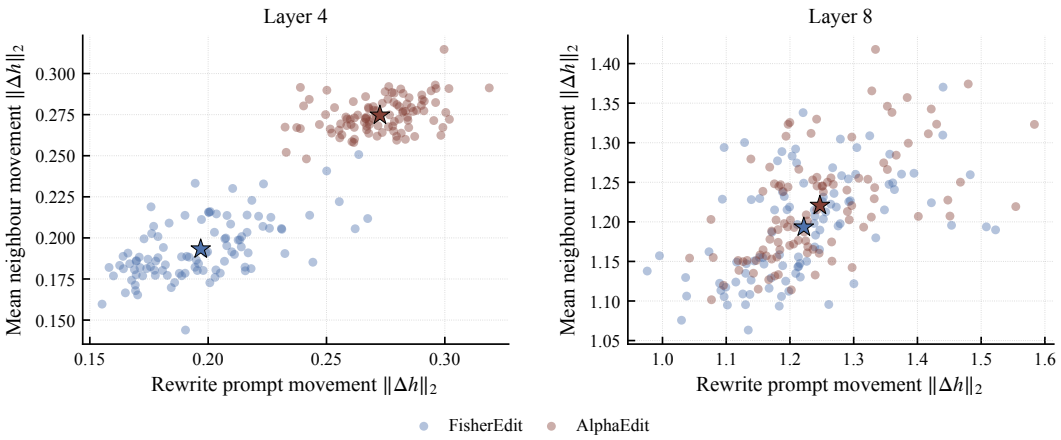

Figure 6: Hidden-state movement of rewrite and neighborhood prompts at layers 4 and 8.

layer 8, the advantage becomes less pronounced but remains visible. This attenuation is expected, since later edited layers receive representations that have already been affected by preceding edited layers, making the hidden-space comparison noisier. Overall, the case study suggests that FisherEdit better controls hidden representation drift, especially at the early stage of the edited layer stack.

E Weight-Space View of Regularization

To complement the projection analysis in §3, we examine how AlphaEdit's weight updates evolve during sequential editing. Figure 7 reports the total update magnitude per batch. For each edit batch, we aggregate the five edited layers by $\|\Delta_t\|_F = (\sum_\ell \|\Delta_t^{(\ell)}\|_F^2)^{1/2}$. It shows that AlphaEdit's update scale is sharply controlled by λ during sequential editing. With extremely weak regularization ($\lambda = 10^{-5}$), the update magnitude grows by many orders of magnitude as editing proceeds. At $\lambda = 10^{-4}$, the trajectory no longer shows sustained exponential growth, but it exhibits large transient spikes, suggesting a near-unstable boundary. For $\lambda \geq 10^{-3}$, the update magnitude remains bounded throughout the edit stream. This weight-space view refines the main finding in §3 that AlphaEdit's behavior is governed primarily by regularization strength, while the near-identity projector does not by itself prevent unstable updates.

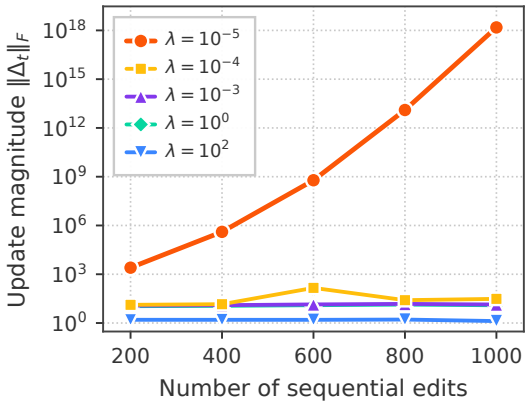

Figure 7: AlphaEdit update magnitude during sequential editing under different λ.

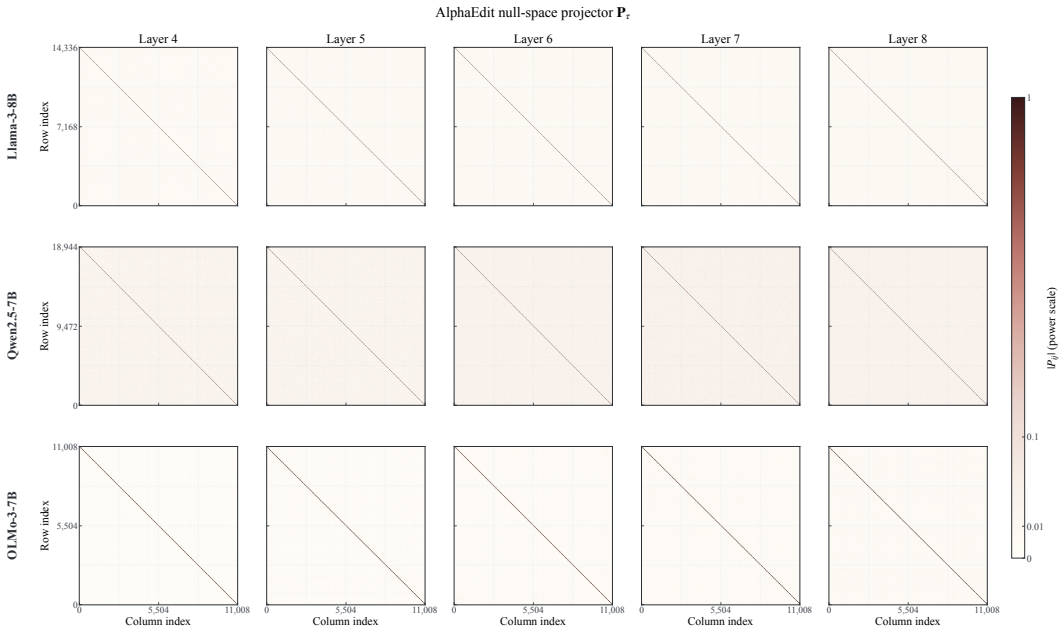

Figure 8: Full-layer visualization of AlphaEdit's null-space projector $\mathbf{P}_\tau$ across edited backbones.

F Full-Layer Visualization of the Null-Space Projector

Figure 8 summarizes the full-layer projectors for the edited backbones. Across layers, $\mathbf{P}_\tau$ remains nearly diagonal and close to identity, indicating that AlphaEdit removes only a limited part of the activation space.

G Derivation of Eqn. (6)

For the Fisher-weighted objective in Eqn. (5), define

$$L(\tilde{\Delta}) = \|\tilde{\Delta}\mathbf{K}_1 - \mathbf{E}\|^2 + \|\tilde{\Delta}\mathbf{K}_p\|^2 + \lambda\|\sqrt{\mathbf{F}} \odot \tilde{\Delta}\|^2. \tag{17}$$

Taking the derivative with respect to $\tilde{\Delta}$ gives

$$\nabla_{\tilde{\Delta}} L = 2(\tilde{\Delta}\mathbf{K}_1 - \mathbf{E})\mathbf{K}_1^\top + 2\tilde{\Delta}\mathbf{K}_p\mathbf{K}_p^\top + 2\lambda(\mathbf{F} \odot \tilde{\Delta}). \tag{18}$$

Setting the derivative to zero and rearranging yields

$$\tilde{\Delta}(\mathbf{K}_1\mathbf{K}_1^\top + \mathbf{K}_p\mathbf{K}_p^\top) + \lambda(\mathbf{F} \odot \tilde{\Delta}) = \mathbf{E}\mathbf{K}_1^\top. \tag{19}$$

Let $\mathbf{G} = \mathbf{K}_1\mathbf{K}_1^\top + \mathbf{K}_p\mathbf{K}_p^\top$. Because the Fisher penalty is element-wise, this normal equation decomposes across rows. Let $\tilde{\delta}_i$, e_i, and f_i denote the i-th rows of $\tilde{\Delta}$, $\mathbf{E}$, and $\mathbf{F}$, respectively, and define $\mathbf{D}_i = \mathrm{diag}(f_i)$. The i-th row therefore satisfies

$$\tilde{\delta}_i(\mathbf{G} + \lambda\mathbf{D}_i) = e_i\mathbf{K}_1^\top, \tag{20}$$

which gives the closed-form row-wise solution

$$\tilde{\delta}_i = e_i\mathbf{K}_1^\top(\mathbf{G} + \lambda\mathbf{D}_i)^{-1}. \tag{21}$$

At the optimum, $\tilde{\delta}_i = \delta_i$, recovering Eqn. (6).

For the efficient implementation, concatenate the current and historical keys as $\mathbf{U} = [\mathbf{K}_1 \mid \mathbf{K}_p] \in \mathbb{R}^{m\times r}$, with $r = n_1 + n_p$, so that $\mathbf{G} = \mathbf{U}\mathbf{U}^\top$. The Woodbury identity gives

$$\begin{aligned}(\mathbf{G} + \lambda\mathbf{D}_i)^{-1} = (\lambda\mathbf{D}_i)^{-1} - (\lambda\mathbf{D}_i)^{-1}\mathbf{U} \\ \left(\mathbf{I}_r + \mathbf{U}^\top(\lambda\mathbf{D}_i)^{-1}\mathbf{U}\right)^{-1}\mathbf{U}^\top(\lambda\mathbf{D}_i)^{-1}.\end{aligned} \tag{22}$$

Substituting this expression into Eqn. (6) yields Eqn. (7), replacing the $m \times m$ inverse with diagonal scaling and an $r \times r$ linear solve.

H Full Derivation of §4.4

In the following, we give the full derivation of the theoretical interpretation in §4.4. Specifically, we show that the element-wise Fisher regularizer in Eqn. (5) corresponds to a computable diagonal approximation of the second-order Taylor expansion of the locality KL objective around $\Delta = 0$, restricted to the edited parameter subspace.

H.1 Setup and Notation

Let $\mathbf{W}$ denote the pre-edit value of the edited parameter block and Δ the update. Following Eqn. (9), the single-sample locality cost is

$$\ell_x(\Delta) = \mathrm{KL}\big(p_{\mathbf{W}}(\cdot \mid x) \,\|\, p_{\mathbf{W}+\Delta}(\cdot \mid x)\big), \tag{23}$$

and the reference-distribution objective is $\mathcal{L}_{\mathrm{loc}}(\Delta) = \mathbb{E}_{x\sim\mathcal{D}_{\mathrm{ref}}}[\ell_x(\Delta)]$.

According to Eqn. (23), we have

$$\begin{aligned}\ell_x(\Delta) =& \mathrm{KL}\big(p_{\mathbf{W}}(\cdot \mid x) \,\|\, p_{\mathbf{W}+\Delta}(\cdot \mid x)\big) \\ =& \sum_y p_{\mathbf{W}}(y \mid x) \log \frac{p_{\mathbf{W}}(y \mid x)}{p_{\mathbf{W}+\Delta}(y \mid x)} \\ =& \sum_y p_{\mathbf{W}}(y \mid x) \log p_{\mathbf{W}}(y \mid x) \\ & - \sum_y p_{\mathbf{W}}(y \mid x) \log p_{\mathbf{W}+\Delta}(y \mid x).\end{aligned} \tag{24}$$

The first term depends only on the pre-edit distribution $p_{\mathbf{W}}(\cdot)$ and is therefore constant with respect to Δ. Thus, we can write

$$\ell_x(\Delta) = \mathrm{const}(x) - \sum_y p_{\mathbf{W}}(y \mid x) \log p_{\mathbf{W}+\Delta}(y \mid x), \tag{25}$$

where $\mathrm{const}(x) = \sum_y p_{\mathbf{W}}(y \mid x) \log p_{\mathbf{W}}(y \mid x)$. Therefore, only the cross-entropy term contributes derivatives with respect to Δ.

H.2 Regularity Conditions

The derivation assumes the following standard regularity conditions on the model distribution:

- *Smoothness.* $p_{\mathbf{W}}(y \mid x)$ is twice continuously differentiable in the edited parameters $\mathbf{W}$.
- *Parameter-independent support.* The support $\{y : p_{\mathbf{W}}(y \mid x) > 0\}$ does not depend on $\mathbf{W}$. This holds for softmax outputs since every token has positive probability.
- *Interchange of differentiation and summation/expectation.* Derivatives with respect to w can be interchanged with the summation over y and the expectation over $x \sim \mathcal{D}_{\mathrm{ref}}$.
- *Finite Fisher moments.* $\mathbb{E}_{y\sim p_{\mathbf{W}}(\cdot|x)} \|\nabla_{\mathbf{W}} \log p_{\mathbf{W}}(y \mid x)\|^2 < \infty$, and the corresponding Hessian expectation exists.

Under these conditions, the information equality between the KL Hessian and the Fisher information holds (Amari, 1998).

H.3 The Taylor Expansion of Single-Sample Locality Cost

Next, we consider the Taylor expansion of Eqn. (25) around $\Delta = 0$. Let $u = \mathrm{vec}(\Delta)$. We have,

$$\begin{aligned}\ell_x(\Delta) = \ell_x(0) + \nabla_u \ell_x(0)^\top u \\ + \tfrac{1}{2} u^\top \nabla_u^2 \ell_x(0) u + o(\|u\|^2).\end{aligned} \tag{26}$$

Zeroth order terms. Setting $\Delta = 0$ in Eqn. (23) gives $\ell_x(0) = \mathrm{KL}(p_{\mathbf{W}} \,\|\, p_{\mathbf{W}}) = 0$.

First order terms. Differentiating Eqn. (25) with respect to Δ gives

$$\nabla_\Delta \ell_x(\Delta) = -\sum_y p_{\mathbf{W}}(y|x)\, \nabla_\Delta \log p_{\mathbf{W}+\Delta}(y|x), \tag{27}$$

Let $\bar{\mathbf{W}} = \mathbf{W} + \Delta$. Since $\mathrm{vec}(\bar{\mathbf{W}}) = \mathrm{vec}(\mathbf{W}) + \mathrm{vec}(\Delta)$, the Jacobian $\partial \mathrm{vec}(\bar{\mathbf{W}})/\partial \mathrm{vec}(\Delta)$ is the identity. Therefore,

$$\begin{aligned}&\nabla_{\mathrm{vec}(\Delta)} \log p_{\mathbf{W}+\Delta}(y \mid x)\big|_{\Delta=0} \\ =& \frac{\partial \mathrm{vec}(\mathbf{W}+\Delta)}{\partial \mathrm{vec}(\Delta)} \nabla_{\mathrm{vec}(\mathbf{W}+\Delta)} \log p_{\mathbf{W}+\Delta}(y|x)\Big|_{\Delta=0} \\ =& \nabla_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y \mid x).\end{aligned} \tag{28}$$

For $\Delta = 0$, by Eqn. (27) and Eqn. (28), we have

$$\nabla_\Delta \ell_x(0) = -\sum_y p_{\mathbf{W}}(y \mid x)\, \nabla_{\mathbf{W}} \log p_{\mathbf{W}}(y \mid x). \tag{29}$$

By swapping summation and differentiation, and using $\sum_y p_{\mathbf{W}}(y \mid x) = 1$, we can derive

$$\begin{aligned}&\sum_y p_{\mathbf{W}}(y \mid x) \nabla_{\mathbf{W}} \log p_{\mathbf{W}}(y \mid x) \\ =& \sum_y \nabla_{\mathbf{W}} p_{\mathbf{W}}(y \mid x) \\ =& \nabla_{\mathbf{W}} \sum_y p_{\mathbf{W}}(y \mid x) \\ =& 0.\end{aligned} \tag{30}$$

Therefore, we have $\nabla_\Delta \ell_x(0) = 0$.

Second order terms. Let $u := \mathrm{vec}(\Delta) \in \mathbb{R}^{md}$ for $\Delta \in \mathbb{R}^{d \times m}$. Treat ℓ_x as a scalar function of u and expand it around $u = 0$. From the derivation above, both the zeroth-order and first-order terms vanish. Thus,

$$\ell_x(\Delta) = \tfrac{1}{2}\, u^\top \mathbf{H}_x\, u + o(\|u\|^2), \tag{31}$$

where

$$\mathbf{H}_x := \nabla_u^2 \ell_x(\Delta)\big|_{\Delta=0}. \tag{32}$$

To compute $\mathbf{H}_x$, we differentiate Eqn. (25) twice with respect to u. Since $p_{\mathbf{W}}(y \mid x)$ and $\mathrm{const}(x)$ do not depend on Δ, we have

$$\nabla_u^2 \ell_x(\Delta) = -\sum_y p_{\mathbf{W}}(y|x)\, \nabla_u^2 \log p_{\mathbf{W}+\Delta}(y|x). \tag{33}$$

Let $\bar{\mathbf{W}} = \mathbf{W} + \Delta$. As shown in Eqn. (28), $\mathrm{vec}(\bar{\mathbf{W}})$ depends on $u = \mathrm{vec}(\Delta)$ through an identity Jacobian. Therefore,

$$\nabla_u^2 \log p_{\mathbf{W}+\Delta}(y|x)\big|_{\Delta=0} = \nabla^2_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y|x). \tag{34}$$

Substituting this into the Hessian gives

$$\mathbf{H}_x = -\sum_y p_{\mathbf{W}}(y \mid x)\, \nabla^2_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y \mid x). \tag{35}$$

Define the single-sample Fisher information matrix restricted to the edited block as

$$\mathbf{F}_x := \sum_y p_{\mathbf{W}}(y \mid x)\, \mathbf{s}_{\mathbf{W}}(y,x)\mathbf{s}_{\mathbf{W}}(y,x)^\top, \tag{36}$$

where

$$\mathbf{s}_{\mathbf{W}}(y,x) := \nabla_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y \mid x). \tag{37}$$

Under the conditions in Appendix H.2, $\mathbf{H}_x = \mathbf{F}_x$. To see this, we differentiate $\nabla_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}} = p_{\mathbf{W}}^{-1} \nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}$ once more:

$$\begin{aligned} &\nabla^2_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}} \\ =&\frac{\nabla^2_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}}{p_{\mathbf{W}}} - \frac{\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}\right)\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}\right)^\top}{p_{\mathbf{W}}^2}. \end{aligned} \tag{38}$$

Multiplying Eqn. (38) by $p_{\mathbf{W}}(y \mid x)$ and summing over y gives

$$\begin{aligned} &-\sum_y p_{\mathbf{W}}(y \mid x) \nabla^2_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y \mid x) \\ =&-\sum_y \nabla^2_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x) \\ &+\sum_y \frac{\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x)\right)\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x)\right)^\top}{p_{\mathbf{W}}(y \mid x)}. \end{aligned} \tag{39}$$

The first term is zero because

$$\begin{aligned} \sum_y \nabla^2_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x) =& \nabla^2_{\mathrm{vec}(\mathbf{W})} \sum_y p_{\mathbf{W}}(y \mid x) \\ =& \nabla^2_{\mathrm{vec}(\mathbf{W})} 1 \\ =& 0. \end{aligned} \tag{40}$$

For the second term, using $\nabla p_{\mathbf{W}} = p_{\mathbf{W}} \nabla \log p_{\mathbf{W}}$, we obtain

$$\begin{aligned} &\sum_y \frac{\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x)\right)\left(\nabla_{\mathrm{vec}(\mathbf{W})} p_{\mathbf{W}}(y \mid x)\right)^\top}{p_{\mathbf{W}}(y \mid x)} \\ =&\sum_y p_{\mathbf{W}}(y \mid x)\, \mathbf{s}_{\mathbf{W}}(y,x)\mathbf{s}_{\mathbf{W}}(y,x)^\top = \mathbf{F}_x. \end{aligned} \tag{41}$$

Therefore,

$$\mathbf{F}_x = -\sum_y p_{\mathbf{W}}(y \mid x) \nabla^2_{\mathrm{vec}(\mathbf{W})} \log p_{\mathbf{W}}(y \mid x) = \mathbf{H}_x, \tag{42}$$

which is the information equality.

H.4 Averaging over the Reference Distribution

Substituting $\mathbf{H}_x = \mathbf{F}_x$ into Eqn. (31) and taking the expectation over $x \sim \mathcal{D}_{\mathrm{ref}}$,

$$\mathcal{L}_{\mathrm{loc}}(\Delta) \approx \tfrac{1}{2}\, \mathrm{vec}(\Delta)^\top \mathbf{F}^{(S)}\, \mathrm{vec}(\Delta), \tag{43}$$

where

$$\mathbf{F}^{(S)} := \mathbb{E}_{x \sim \mathcal{D}_{\mathrm{ref}}}[\mathbf{F}_x] \tag{44}$$

is the Fisher information matrix *restricted to the edited parameter subspace*. This restriction is not cosmetic: the edit only perturbs the weights of a single MLP down-projection block, so the Taylor expansion lives on that subspace and $\mathbf{F}^{(S)}$ is the corresponding Fisher block rather than the full-model Fisher.

H.5 Diagonal Approximation and Recovery of Eqn. (5)

For $\Delta \in \mathbb{R}^{d\times m}$, the full block $\mathbf{F}^{(S)}$ has size $(md) \times (md)$. At LLM scale (md on the order of 10^7), storing $\mathbf{F}^{(S)}$ or solving the associated quadratic program in closed form is infeasible. We therefore adopt the standard diagonal approximation

$$\mathbf{F}^{(S)} \approx \mathrm{Diag}(f), \qquad f \in \mathbb{R}^{md}, \tag{45}$$

and reshape f into a matrix $\widetilde{\mathbf{F}} \in \mathbb{R}^{d\times m}$ of the same shape as Δ. The quadratic form collapses to an element-wise penalty:

$$\mathrm{vec}(\Delta)^\top \mathrm{Diag}(f)\,\mathrm{vec}(\Delta) = \sum_{i,j} F_{ij}\,\Delta_{ij}^2. \tag{46}$$

Combining Eqns. (43) and (46),

$$\mathcal{L}_{\mathrm{loc}}(\Delta) \approx \tfrac{1}{2}\sum_{i,j} F_{ij}\,\Delta_{ij}^2, \tag{47}$$

which is exactly the regularizer appearing in Eqn. (5).

I Case Study

We present representative paired outputs from Ripple. Each example reports two prompts from the same edited sample: a paraphrase of the edited fact and a neighborhood prompt about a nearby fact that should remain unchanged. Across these paired examples, FisherEdit consistently produces the edited answer while preserving the unrelated local fact.

Case 1

Edit / Neighborhood. John Masly: occupation *researcher* → *unemployed*; gender should remain *male*.
Outputs. FisherEdit: *unemployed* / *male*. AlphaEdit: *unknown* / *unknown*. AnyEdit: "John Masly is a historian." / "John P Masly is male." MEMIT: unrelated edit continuation / "male and female farmers, also genderfarmersqueepea".

Case 2

Edit / Neighborhood. Sharifah Mazlina binti Syed Abdul Kadir: occupation *explorer* → *music professor*; citizenship should remain *Malaysia*.
Outputs. FisherEdit: *music professor* / *Malaysia*. AlphaEdit: "music teacher and calligrapher" / *Iran*. AnyEdit: "Minister of Women's Empowerment and Family Development in" / "citizen of Malaysia." MEMIT: corrupted text on both prompts.

Case 3

Edit / Neighborhood. Anthony Deakin: occupation *researcher* → *art dealer*; alma mater should remain *University of Liverpool*.
Outputs. FisherEdit: *art dealer* / *University of Liverpool*. AlphaEdit: *art dealer* / "known only to a few". AnyEdit: "Royal Navy officer and politician." / *University of Sheffield*. MEMIT: degenerate continuations on both prompts.

J FUTURE WORK

Scaling to longer edit streams. An important direction is to extend FisherEdit to substantially longer editing sequences by improving the scalability of its solver and historical-key memory. In Eqn. (7), $r = n_1 + n_p$ counts both the current and accumulated historical edit keys, and the row-wise solve costs $O(dmr^2 + dr^3)$ per edited weight matrix. Since Algorithm 1 appends new keys after each batch, n_p grows throughout the stream even when the batch size is fixed. The computational advantage of the Woodbury formulation relies on $r \ll m$, motivating further improvements to the row-wise solver as historical constraints accumulate. Future work could explore incremental updates to the row-wise factorizations and warm-started iterative solvers that reuse computations across successive edit batches, with the aim of reducing editing latency while preserving earlier edits.

Knowledge editing in reasoning models. FisherEdit's parameter-importance regularization provides a natural basis for updating factual knowledge while protecting the reasoning processes that use it. The method requires locate-then-edit MLP locations and a Fisher diagonal estimated on a calibration set, without architectural modifications. The calibration set serves as an adaptable control knob: estimating the Fisher diagonal directly on complex reasoning traces, such as multi-step mathematical or logical trajectories, would penalize changes to parameters with high sensitivity on these traces. This offers a way to direct preservation toward reasoning-related behavior while allowing factual updates through less sensitive parameters. Future work can investigate this separation between factual recall and reasoning, and develop sequential-editing evaluations that jointly assess successful knowledge replacement, the use of updated facts in multi-step solutions, and preservation of reasoning ability.

Complementarity with retrieval and deployment scenarios. Parameter editing and retrieval-based approaches are complementary. When retrieved context conflicts with memorized knowledge, models may over-rely on stale parametric memory (Longpre et al., 2021), and their use of external evidence depends on its coherence and persuasiveness (Xie et al., 2024). Editing updates the fact at its source, so the correction can remain available when retrieval misses, ranks poorly, is only partially relevant, or is disabled. This motivates combining FisherEdit with retrieval-augmented generation or agentic search to improve consistency between parametric knowledge and external evidence. On-device, offline, and privacy-sensitive deployments also provide useful settings for exploration, particularly when maintaining a retrieval corpus and pipeline is impractical. Once applied, FisherEdit introduces no additional inference-time computation; latency- and cost-sensitive serving can therefore benefit from factual corrections that do not require a retrieval step on every query.