RHOEDIT: BALANCING MODEL EDITING AND CAPABILITY PRESERVATION VIA RELATIVE CURVATURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Model editing aims to change a specific fact or behavior while leaving the rest of the model working as before. This is difficult because parameter updates can damage unrelated knowledge and general capabilities. We introduce RHOEDIT, which formulates capability-preserving model editing as constrained optimization and derives a relative-curvature gradient adjustment from a local second-order model. The adjustment jointly uses edit and capability curvature, attenuating each gradient component more strongly along directions of greater relative capability sensitivity. It thereby retains useful editing directions while protecting the behavior represented by a capability reference set. We approximate both curvatures with Kronecker-factored approximate curvature to make the adjustment practical for LLMs. Across ZsRE, CounterFact, and WikiBigEdit, RHOEDIT achieves an average editing score of 68.8 while retaining nearly all of the unedited model's general capabilities (65.1 versus 65.4). Compared with AlphaEdit, RHOEDIT improves editing by 1.8 points and general capabilities by 18.5 points; compared with CrispEdit, it improves editing by 6.8 points while slightly improving general capabilities. These results show that relative curvature improves the balance between model editing and capability preservation.

1 INTRODUCTION

Large language models (LLMs) acquire broad knowledge and skills during pretraining. After training, however, facts change, errors are discovered, and applications may require new behavior. Retraining the entire model for every small correction is expensive and slow. Model editing studies how to change a specific fact or behavior without rebuilding the model from scratch (Sinitsin et al., 2020; De Cao et al., 2021; Yao et al., 2023; Wang et al., 2024c). Its goal is simple to state: make the intended change while leaving the rest of the model working as before.

This goal is difficult because a language model does not expose its knowledge as independent records that can simply be replaced (Geva et al., 2021; Dai et al., 2022; Meng et al., 2022). The same parameters contribute to many predictions, so an update for one prompt can also change answers to other prompts. If the update is too limited, the model may reproduce the new answer only for the exact wording it has seen. If the update is too broad, it may damage unrelated knowledge or general skills. Model editing must therefore make the requested change while preserving the model's existing abilities.

Research on model editing has developed along several complementary paths. Early work trained neural networks to accept targeted updates or learned an editor that predicts parameter changes from edit examples (Sinitsin et al., 2020; De Cao et al., 2021; Mitchell et al., 2022a). Locate-then-edit methods later identified layers or representations associated with a fact and modified only those parts of the model (Dai et al., 2022; Meng et al., 2022; 2023). In parallel, memory- and adapter-based methods kept the base model largely unchanged and stored new behavior in additional components (Mitchell et al., 2022b; Hartvigsen et al., 2023; Yu et al., 2024; Wang et al., 2024a). Drawing on constrained optimization and continual learning, another line restricted parameter updates to protect representations or directions considered important to existing behavior (Zhu et al., 2020; Wang et al., 2021; Fang et al., 2025; Ikram et al., 2026). Together, these approaches have expanded model editing from isolated factual corrections to settings with many edits.

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

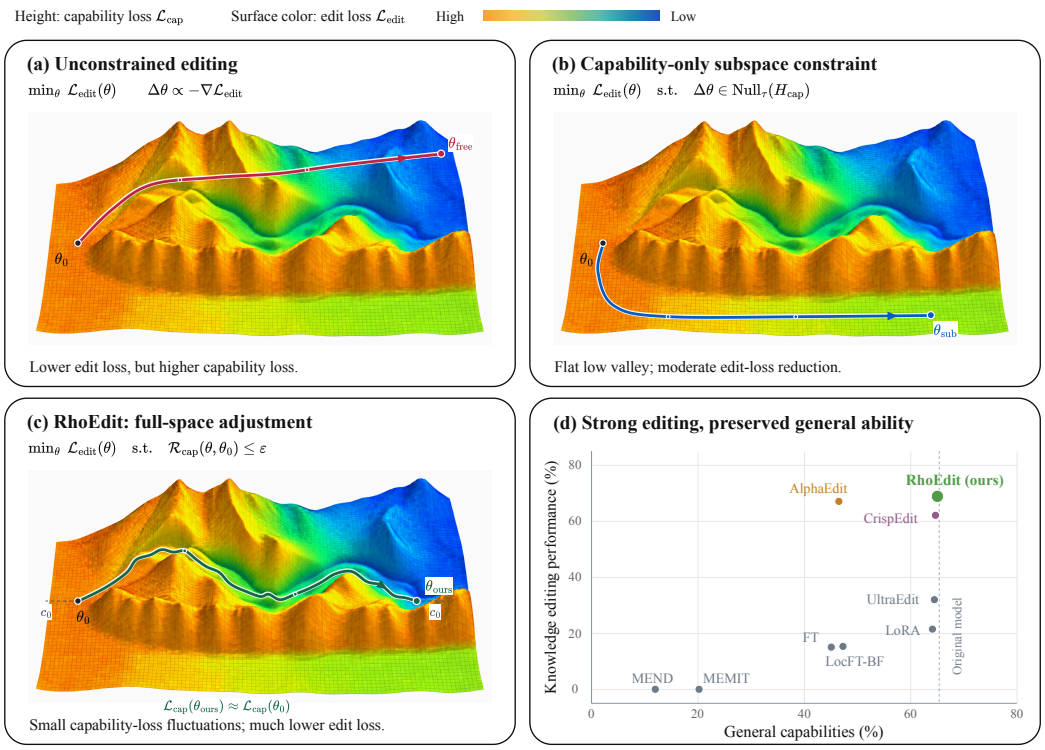

Figure 1: **Geometric intuition and results for RHOEDIT.** (a) Unconstrained editing lowers edit loss at the cost of capability degradation. (b) A capability-only subspace preserves capability but excludes useful editing directions. (c) RHOEDIT adjusts updates in the full space, attaining lower edit loss at the same capability-loss level. (d) RHOEDIT achieves the highest average editing performance while maintaining general capabilities near the unedited model's level.

Despite these improvements, existing methods typically either derive updates primarily from edit information or structural assumptions (Yang et al., 2025b; Gu et al., 2026; Meng et al., 2022; 2023; Pan et al., 2025) or use a single preservation set as a proxy for the model's broader capabilities when selecting admissible update subspaces (Fang et al., 2025; Ikram et al., 2026), leading to a suboptimal trade-off between editing efficacy and preservation.

In this paper, we introduce RHOEDIT, an editor that jointly uses edit and capability curvature to balance learning new behavior with retaining existing behavior. Figure 1 illustrates how our full-space adjustment retains useful editing directions excluded by a capability-only subspace constraint while maintaining strong preservation, thereby achieving a better edit–preservation trade-off. Our approach builds on three core ideas.

1. Gradient adjustment via relative curvature. We formulate editing as a constrained optimization problem, solve a local second-order model of it in closed form, and identify within the solution a gradient adjustment that measures capability curvature relative to edit curvature. The adjustment then shrinks the gradient along each direction by the capability that direction costs per unit of edit progress.

2. A common curvature geometry for editing and preservation. We formulate both edit and capability curvature using generalized Gauss–Newton matrices and show that they coincide with the model Fisher information matrices in our setting. These matrices measure how sensitive the model's predictions are to parameter changes on edit and capability data. This connects the relative-curvature adjustment to predictive sensitivity and enables both curvatures to be estimated using the same computational tools.

3. Scalable K-FAC realization for LLMs. Applying the joint-curvature adjustment directly would require matrices too large to construct for LLMs. We approximate both edit and capability curvature

with Kronecker-factored approximate curvature (K-FAC) (Martens & Grosse, 2015; George et al., 2018). This lets us apply the relative-curvature adjustment mechanism practical at LLM scale.

Experimental results. We evaluate RHOEDIT on Llama-3-8B-Instruct across ZsRE, CounterFact, and WikiBigEdit (Table 1). Across the three datasets, RHOEDIT achieves the strongest overall balance between editing and capability preservation, scoring 68.8 on average for editing while retaining nearly all of the unedited model's general capabilities (65.1 versus 65.4; Figure 1(d)). Compared with AlphaEdit (Fang et al., 2025), which comes closest in editing performance, RHOEDIT improves editing by 1.8 points while achieving an 18.5-point higher average across the five general-capability benchmarks. Compared with CrispEdit (Ikram et al., 2026), which comes closest in capability preservation, RHOEDIT improves editing by 6.8 points while slightly improving general-capability performance.

2 PROBLEM FORMULATION: MODEL EDITING

Let $\boldsymbol{\theta}_0$ denote the parameters of a base model and $\boldsymbol{\theta} \in \Theta$ the parameters after editing. For a next-token prediction context x, the model produces logits $\boldsymbol{z}_{\boldsymbol{\theta}}(x)$ and a predictive distribution $p_{\boldsymbol{\theta}}(\cdot \mid x) = \mathrm{softmax}(\boldsymbol{z}_{\boldsymbol{\theta}}(x))$. Given an edit dataset $\mathcal{D}_{\mathrm{edit}} = \{(x_i^{\mathrm{e}}, y_i^{\mathrm{e}})\}_{i=1}^{T}$, we measure edit performance by the negative log-likelihood $\mathcal{L}_{\mathrm{edit}}(\boldsymbol{\theta}) := -\frac{1}{T}\sum_{i=1}^{T} \log p_{\boldsymbol{\theta}}(y_i^{\mathrm{e}} \mid x_i^{\mathrm{e}})$. To quantify changes in the behavior to preserve, we use a reference dataset $\mathcal{D}_{\mathrm{cap}} = \{x_j^{\mathrm{c}}\}_{j=1}^{N}$ and define $\mathcal{R}_{\mathrm{cap}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0) := \frac{1}{N}\sum_{j=1}^{N} d\big(\boldsymbol{z}_{\boldsymbol{\theta}}(x_j^{\mathrm{c}}), \boldsymbol{z}_{\boldsymbol{\theta}_0}(x_j^{\mathrm{c}})\big)$, where d is a nonnegative discrepancy between model outputs. Model editing then minimizes the edit loss subject to a bounded change on the capability reference set:

$$\min_{\boldsymbol{\theta}\in\Theta} \ \mathcal{L}_{\mathrm{edit}}(\boldsymbol{\theta}) \quad \text{subject to} \quad \mathcal{R}_{\mathrm{cap}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0) \le \varepsilon, \tag{1}$$

where $\varepsilon \ge 0$ bounds the empirical change on the capability reference set. A standard Lagrangian relaxation of equation 1 minimizes $\mathcal{L}_{\mathrm{edit}}(\boldsymbol{\theta}) + \lambda \mathcal{R}_{\mathrm{cap}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$ for a fixed multiplier $\lambda \ge 0$.

3 RELATIVE-CURVATURE GRADIENT ADJUSTMENT FOR MODEL EDITING

Model editing seeks progress on the requested behavior while limiting changes to the behavior to preserve. The key idea of RHOEDIT is to adjust the edit gradient according to capability curvature relative to edit curvature. Components with larger relative capability curvature receive stronger attenuation, while those with smaller relative capability curvature retain more of the edit gradient.

We begin in Section 3.1 by deriving a local second-order model of the model editing problem formulated in equation 1 and solving it in closed form, revealing a relative-curvature gradient adjustment. Then, we derive a Gauss–Newton formulation of the adjustment in Section 3.2, and make the adjustment computable at LLM scale in Section 3.3.

3.1 FROM CONSTRAINED EDITING TO RELATIVE-CURVATURE ADJUSTMENT

To relate edit progress to capability cost, we first examine the local geometry of equation 1 at $\boldsymbol{\theta}_0$. We analyze this under the following assumption on d.

Assumption 1. *The map $\boldsymbol{\theta} \mapsto \boldsymbol{z}_{\boldsymbol{\theta}}(x)$ is twice continuously differentiable, and the discrepancy d is twice continuously differentiable in its first argument and satisfies $d(\boldsymbol{v}, \boldsymbol{v}) = 0$, $\nabla_{\boldsymbol{u}} d(\boldsymbol{u}, \boldsymbol{v})|_{\boldsymbol{u}=\boldsymbol{v}} = 0$, and $\nabla^2_{\boldsymbol{u}} d(\boldsymbol{u}, \boldsymbol{v})|_{\boldsymbol{u}=\boldsymbol{v}} \succeq 0$ for every model output $\boldsymbol{v}$.*

These conditions are mild: a twice-differentiable nonnegative d with $d(\boldsymbol{v}, \boldsymbol{v}) = 0$ is minimized at $\boldsymbol{u} = \boldsymbol{v}$, so the remaining two conditions follow. Bregman divergences provide a broad family of such discrepancies (Bregman, 1967). In particular, the log-sum-exp generator on logits yields $d(\boldsymbol{z}_{\boldsymbol{\theta}}, \boldsymbol{z}_{\boldsymbol{\theta}_0}) = D_{\mathrm{KL}}(p_{\boldsymbol{\theta}_0} \| p_{\boldsymbol{\theta}})$, which we use throughout this paper. Appendix A gives the derivation.

Under Assumption 1, $\mathcal{R}_{\mathrm{cap}}$ vanishes together with its gradient at $\boldsymbol{\theta}_0$, whereas the edit loss has a nonzero gradient there, so the second order local models of both terms of equation 1 at $\boldsymbol{\theta}_0$ are

$$\begin{aligned} \mathcal{R}_{\mathrm{cap}}(\boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}, \boldsymbol{\theta}_0) &= \tfrac{1}{2}\Delta\boldsymbol{\theta}^\top \boldsymbol{H}_{\mathrm{cap}} \Delta\boldsymbol{\theta} + o\big(\|\Delta\boldsymbol{\theta}\|_2^2\big)\,, \\ \mathcal{L}_{\mathrm{edit}}(\boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}) &= \mathcal{L}_{\mathrm{edit}}(\boldsymbol{\theta}_0) + \boldsymbol{g}_{\mathrm{edit}}^\top \Delta\boldsymbol{\theta} + \tfrac{1}{2}\Delta\boldsymbol{\theta}^\top \boldsymbol{H}_{\mathrm{edit}} \Delta\boldsymbol{\theta} + o\big(\|\Delta\boldsymbol{\theta}\|_2^2\big)\,, \end{aligned} \tag{2}$$

with $\boldsymbol{g}_{\text{edit}} := \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{edit}}(\boldsymbol{\theta}_0)$, $\boldsymbol{H}_{\text{edit}} := \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_{\text{edit}}(\boldsymbol{\theta}_0)$, and $\boldsymbol{H}_{\text{cap}} := \nabla^2_{\boldsymbol{\theta}} \mathcal{R}_{\text{cap}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0)|_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}$. The two Hessians describe complementary effects of the same parameter change: $\boldsymbol{H}_{\text{edit}}$ its effect on the edit objective, and $\boldsymbol{H}_{\text{cap}}$ its effect on the behavior to preserve.

We apply the local models in equation 2 to the Lagrangian relaxation of equation 1. Writing $\boldsymbol{\theta} = \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}$, substituting the two expansions into $\mathcal{L}_{\text{edit}}(\boldsymbol{\theta}) + \lambda \mathcal{R}_{\text{cap}}(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$, dropping the constant $\mathcal{L}_{\text{edit}}(\boldsymbol{\theta}_0)$, and retaining terms up to second order gives the following local quadratic problem:

Local formulation

$$\min_{\Delta\boldsymbol{\theta}} \; \boldsymbol{g}_{\text{edit}}^{\top} \Delta\boldsymbol{\theta} + \frac{1}{2} \Delta\boldsymbol{\theta}^{\top} \boldsymbol{H}_{\text{edit}} \Delta\boldsymbol{\theta} + \frac{\lambda}{2} \Delta\boldsymbol{\theta}^{\top} \boldsymbol{H}_{\text{cap}} \Delta\boldsymbol{\theta}. \tag{3}$$

The multiplier λ controls the trade-off between edit progress and capability preservation in this local model. In practice, we select λ directly as the capability-penalty strength; Appendix B shows how the capability cost under the local quadratic model decreases as λ increases.

Setting the gradient of equation 3 to zero yields the closed-form solution:

Solution

$$\Delta\boldsymbol{\theta}^{\star} = -\left(\boldsymbol{H}_{\text{edit}} + \lambda \boldsymbol{H}_{\text{cap}}\right)^{-1} \boldsymbol{g}_{\text{edit}} = -\boldsymbol{H}_{\text{edit}}^{-1} \left(\boldsymbol{I} + \lambda \boldsymbol{H}_{\text{cap}} \boldsymbol{H}_{\text{edit}}^{-1}\right)^{-1} \boldsymbol{g}_{\text{edit}}. \tag{4}$$

The factorization in equation 4 can be understood as two steps. To describe them, we define

$$\boldsymbol{g}_{\text{edit}}^{\text{adj}} := \mathcal{A} \boldsymbol{g}_{\text{edit}}, \qquad \mathcal{A} := \left(\boldsymbol{I} + \lambda \boldsymbol{H}_{\text{cap}} \boldsymbol{H}_{\text{edit}}^{-1}\right)^{-1}. \tag{5}$$

The transformation $\mathcal{A}$, which we call the *relative-curvature gradient adjustment*, maps the edit gradient $\boldsymbol{g}_{\text{edit}}$ to the adjusted gradient $\boldsymbol{g}_{\text{edit}}^{\text{adj}}$. The subsequent multiplication by $-\boldsymbol{H}_{\text{edit}}^{-1}$ gives the Newton step for the local quadratic model of $\mathcal{L}_{\text{edit}}$, using the adjusted gradient.

To characterize how $\mathcal{A}$ transforms the edit gradient, we seek a directional decomposition in which $\mathcal{A}$ scales each gradient component independently and the associated scaling factors can be expressed in terms of the edit and capability curvatures. Identifying such a decomposition amounts to solving the generalized eigenvalue problem $\boldsymbol{C}_{\text{cap}} \boldsymbol{v} = \rho \boldsymbol{C}_{\text{edit}} \boldsymbol{v}$ for generalized eigenpairs $(\rho, \boldsymbol{v})$ with $\boldsymbol{v} \neq \boldsymbol{0}$. The following proposition states the resulting directional form of $\mathcal{A}$.

Proposition 1. *Let $\boldsymbol{C}_{\text{edit}} \succ 0$ and $\boldsymbol{C}_{\text{cap}} \succeq 0$ be symmetric edit and capability curvature matrices. Solving the generalized eigenvalue problem $\boldsymbol{C}_{\text{cap}} \boldsymbol{v} = \rho \boldsymbol{C}_{\text{edit}} \boldsymbol{v}$ can be reduced to the ordinary eigendecomposition $\boldsymbol{C}_{\text{edit}}^{-1/2} \boldsymbol{C}_{\text{cap}} \boldsymbol{C}_{\text{edit}}^{-1/2} = \boldsymbol{Q} \operatorname{diag}(\rho_i) \boldsymbol{Q}^{\top}$, where $\boldsymbol{Q}$ is orthogonal and $\rho_i \geq 0$. For $\mathcal{A} := (\boldsymbol{I} + \lambda \boldsymbol{C}_{\text{cap}} \boldsymbol{C}_{\text{edit}}^{-1})^{-1}$ with $\lambda \geq 0$, every vector $\boldsymbol{g}$ satisfies*

$$\mathcal{A} \boldsymbol{g} = \boldsymbol{C}_{\text{edit}}^{1/2} \boldsymbol{Q} \operatorname{diag}\left(\frac{1}{1 + \lambda \rho_i}\right) \boldsymbol{Q}^{\top} \boldsymbol{C}_{\text{edit}}^{-1/2} \boldsymbol{g}. \tag{6}$$

Equation 6 shows that $\mathcal{A}$ represents the edit gradient in coordinates induced by the generalized eigensystem, multiplies coordinate i by $(1 + \lambda \rho_i)^{-1}$, and then maps the adjusted coordinates back to the original space. By the defining generalized eigenvalue relation, each ρ_i measures capability curvature relative to edit curvature along its associated generalized eigendirection. For a fixed $\lambda > 0$, the scaling factor equals one when $\rho_i = 0$ and decreases as ρ_i increases, so coordinates associated with larger relative curvature are attenuated more strongly. Appendix B gives the full derivation.

The attenuation therefore depends on the ratio between the two curvatures: a direction with large capability curvature can be attenuated weakly when its edit curvature is larger still, whereas a direction with small capability curvature can be attenuated strongly when its edit curvature is smaller still. By accounting jointly for the curvature of the behavior to retain and that of the edit objective, $\mathcal{A}$ provides an editing-aware form of capability protection. This direction-wise dependence on relative curvature gives RHOEDIT its name.

The factorization in equation 4 separates the capability-preserving gradient adjustment $\mathcal{A}$ from the edit-curvature (Newton) preconditioning by $-\boldsymbol{H}_{\text{edit}}^{-1}$. RHOEDIT applies $\mathcal{A}$ to each edit gradient and uses the chosen optimizer to convert the adjusted gradient into a parameter update, replacing the explicit scaling by $-\boldsymbol{H}_{\text{edit}}^{-1}$. [1]

[1] In practice, we observed numerical instability in estimating inverse edit curvature from small edit datasets.

3.2 From Hessians to Gauss–Newton Curvature

The relative-curvature adjustment is intended to attenuate each direction according to a ratio of capability curvature to edit curvature. This interpretation requires the edit curvature to be positive-definite and the capability curvature to be positive semidefinite, as formalized in Proposition 1. However, the Hessian of a neural network need not provide this geometry. For a loss $\ell(\boldsymbol{z}_{\boldsymbol{\theta}}(x))$ composed with the logits, write $\boldsymbol{J}_{\boldsymbol{\theta}} := \partial \boldsymbol{z}_{\boldsymbol{\theta}}(x)/\partial \boldsymbol{\theta}$. The second-order chain rule decomposes the parameter Hessian into a pullback of the output-space loss Hessian through $\boldsymbol{J}_{\boldsymbol{\theta}}$ and a term involving the Hessian of the logit map:

$$\nabla^2_{\boldsymbol{\theta}} \mathbb{E}[\ell(\boldsymbol{z}_{\boldsymbol{\theta}}(x))] = \underbrace{\mathbb{E}\big[\boldsymbol{J}_{\boldsymbol{\theta}}^\top (\nabla^2_{\boldsymbol{z}} \ell) \boldsymbol{J}_{\boldsymbol{\theta}}\big]}_{\boldsymbol{G}} + \mathbb{E}\left[\sum_k (\partial_{z_k} \ell) \nabla^2_{\boldsymbol{\theta}} z_{\boldsymbol{\theta},k}(x)\right]. \tag{7}$$

The first term is the generalized Gauss–Newton (GGN) matrix (Schraudolph, 2002; Martens, 2020). For the output losses used here, $\nabla^2_{\boldsymbol{z}} \ell \succeq 0$, so $\boldsymbol{G} \succeq 0$; the second term can still make the full Hessian indefinite.

The capability discrepancy and edit negative log-likelihood differ in whether the residual term vanishes at the base model. For the capability discrepancy, Assumption 1 gives $\nabla_{\boldsymbol{z}} d(\boldsymbol{z}, \boldsymbol{z}_{\boldsymbol{\theta}_0})|_{\boldsymbol{z}=\boldsymbol{z}_{\boldsymbol{\theta}_0}} = \boldsymbol{0}$ at every reference context. Consequently, each coefficient $\partial_{z_k} d$ in the residual of equation 7 is zero at $\boldsymbol{\theta}_0$, and $\boldsymbol{H}_{\mathrm{cap}} = \boldsymbol{G}_{\mathrm{cap}}$ exactly. For an edit target y, the negative log-likelihood has output gradient $\boldsymbol{s} - \boldsymbol{e}_y$, where $\boldsymbol{s} := \mathrm{softmax}(\boldsymbol{z})$ and $\boldsymbol{e}_y$ is one-hot. This gradient need not vanish at $\boldsymbol{\theta}_0$; replacing $\boldsymbol{H}_{\mathrm{edit}}$ by $\boldsymbol{G}_{\mathrm{edit}}$ therefore discards a potentially nonzero residual and is an approximation.

For categorical next-token distributions, this replacement also gives the curvatures a probabilistic interpretation. Both the edit loss $\ell_y(\boldsymbol{z}) = -\log \mathrm{softmax}(\boldsymbol{z})_y$ and the forward-KL capability discrepancy $d(\boldsymbol{z}, \boldsymbol{z}_0) = D_{\mathrm{KL}}(\mathrm{softmax}(\boldsymbol{z}_0) \| \, \mathrm{softmax}(\boldsymbol{z}))$ have output Hessian

$$\nabla^2_{\boldsymbol{z}} \ell_y(\boldsymbol{z}) = \nabla^2_{\boldsymbol{z}} d(\boldsymbol{z}, \boldsymbol{z}_0) = \mathrm{Diag}(\boldsymbol{s}) - \boldsymbol{s}\boldsymbol{s}^\top = \mathbb{E}_{\tilde{y} \sim \boldsymbol{s}}\big[(\boldsymbol{e}_{\tilde{y}} - \boldsymbol{s})(\boldsymbol{e}_{\tilde{y}} - \boldsymbol{s})^\top\big]. \tag{8}$$

Because $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tilde{y} \mid x) = \boldsymbol{J}_{\boldsymbol{\theta}}^\top (\boldsymbol{e}_{\tilde{y}} - \boldsymbol{s})$, pulling equation 8 back through the logits gives the model Fisher (Martens, 2020; Kunstner et al., 2019). Applying the output-curvature formula above to both the edit and capability datasets and using $\boldsymbol{H}_{\mathrm{cap}} = \boldsymbol{G}_{\mathrm{cap}}$ at $\boldsymbol{\theta}_0$ yields

$$\boldsymbol{G}_{\mathrm{edit}} = \boldsymbol{F}_{\mathrm{edit}}, \qquad \boldsymbol{H}_{\mathrm{cap}} = \boldsymbol{G}_{\mathrm{cap}} = \boldsymbol{F}_{\mathrm{cap}}. \tag{9}$$

To satisfy the positive-definiteness condition for the edit curvature, we use positive damping: $\boldsymbol{G}_{\mathrm{edit}} + \delta \boldsymbol{I} \succ 0$ for $\delta > 0$. We suppress this damping in the notation; the directional characterization of Proposition 1 applies to the resulting positive-definite edit curvature and positive-semidefinite capability curvature. Appendix C gives the full Hessian–GGN and Fisher derivations.

3.3 Scalable K-FAC Realization

After replacing the Hessians by the Gauss–Newton curvatures in Section 3.2, the relative-curvature adjustment in equation 5 becomes

$$\boldsymbol{g}^{\mathrm{adj}}_{\mathrm{edit}} = \big(\boldsymbol{I} + \lambda \boldsymbol{G}_{\mathrm{cap}} \boldsymbol{G}^{-1}_{\mathrm{edit}}\big)^{-1} \boldsymbol{g}_{\mathrm{edit}}. \tag{10}$$

The full curvature matrices remain intractable at LLM scale: for a weight block $\boldsymbol{W} \in \mathbb{R}^{m \times n}$, each contains $m^2 n^2$ entries. By equation 9, both curvatures equal model Fisher matrices. For each dataset, the corresponding Fisher is the expected outer product of $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tilde{y} \mid x)$, where $\tilde{y}$ is sampled from the model. For a linear weight block, let $\boldsymbol{a} \in \mathbb{R}^n$ be its input activation and $\boldsymbol{\delta} \in \mathbb{R}^m$ the gradient of $\log p_{\boldsymbol{\theta}}(\tilde{y} \mid x)$ with respect to its output. Then $\nabla_{\boldsymbol{W}} \log p_{\boldsymbol{\theta}}(\tilde{y} \mid x) = \boldsymbol{\delta}\boldsymbol{a}^\top$. Under column-major vectorization, the outer product of this gradient is $(\boldsymbol{a}\boldsymbol{a}^\top) \otimes (\boldsymbol{\delta}\boldsymbol{\delta}^\top)$. We use K-FAC to approximate the expectation of this Kronecker product by the Kronecker product of the two expectations (Martens & Grosse, 2015; George et al., 2018):

$$\boldsymbol{G}_{\boldsymbol{W}} = \mathbb{E}\big[(\boldsymbol{a}\boldsymbol{a}^\top) \otimes (\boldsymbol{\delta}\boldsymbol{\delta}^\top)\big] \approx \underbrace{\mathbb{E}[\boldsymbol{a}\boldsymbol{a}^\top]}_{\boldsymbol{A}} \otimes \underbrace{\mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top]}_{\boldsymbol{B}}. \tag{11}$$

We compute these expectations separately on the edit and capability datasets, obtaining one pair of factors for each:

$$\boldsymbol{G}_{\mathrm{edit}} \approx \boldsymbol{A}_{\mathrm{edit}} \otimes \boldsymbol{B}_{\mathrm{edit}}, \qquad \boldsymbol{G}_{\mathrm{cap}} \approx \boldsymbol{A}_{\mathrm{cap}} \otimes \boldsymbol{B}_{\mathrm{cap}}, \tag{12}$$

We apply this approximation to each weight block and ignore curvature between different weight blocks. Appendix D describes how we estimate the factors.

We use positive-definite damped edit factors $\boldsymbol{A}_{\text{edit}}$ and $\boldsymbol{B}_{\text{edit}}$ when forming their inverses; the capability factors remain positive semidefinite. Using the inverse and product rules for Kronecker products, we obtain the following approximation to the matrix product in equation 10:

$$\boldsymbol{G}_{\text{cap}}\boldsymbol{G}_{\text{edit}}^{-1} \approx \left(\boldsymbol{A}_{\text{cap}}\boldsymbol{A}_{\text{edit}}^{-1}\right) \otimes \left(\boldsymbol{B}_{\text{cap}}\boldsymbol{B}_{\text{edit}}^{-1}\right). \tag{13}$$

This reduces the directional analysis to two smaller problems, one for the $\boldsymbol{A}$ factors and one for the $\boldsymbol{B}$ factors. Following Proposition 1, we choose $\boldsymbol{Q}_A$ and $\boldsymbol{Q}_B$ so that the edit factors become identity matrices and the capability factors become diagonal:

$$\begin{aligned} \boldsymbol{Q}_A^\top \boldsymbol{A}_{\text{edit}} \boldsymbol{Q}_A = \boldsymbol{I}, \quad \boldsymbol{Q}_A^\top \boldsymbol{A}_{\text{cap}} \boldsymbol{Q}_A = \boldsymbol{D}_A := \operatorname{diag}(a_1, \ldots, a_n), \\ \boldsymbol{Q}_B^\top \boldsymbol{B}_{\text{edit}} \boldsymbol{Q}_B = \boldsymbol{I}, \quad \boldsymbol{Q}_B^\top \boldsymbol{B}_{\text{cap}} \boldsymbol{Q}_B = \boldsymbol{D}_B := \operatorname{diag}(b_1, \ldots, b_m). \end{aligned} \tag{14}$$

The generalized eigenvalues a_j and b_i are nonnegative because the capability factors are positive semidefinite. Appendix D describes the solution of these problems.

Let $\boldsymbol{Q}_\otimes := \boldsymbol{Q}_A \otimes \boldsymbol{Q}_B$. Combining equation 12 and equation 14 gives

$$\boldsymbol{Q}_\otimes^\top \boldsymbol{G}_{\text{edit}} \boldsymbol{Q}_\otimes \approx \boldsymbol{I}, \qquad \boldsymbol{Q}_\otimes^\top \boldsymbol{G}_{\text{cap}} \boldsymbol{Q}_\otimes \approx \boldsymbol{D}_A \otimes \boldsymbol{D}_B. \tag{15}$$

For the K-FAC curvature approximations, equation 15 gives relative curvature $\rho_{ij} = b_i a_j$ for each direction indexed by (i, j). The identities in equation 14 also give the following expression for the matrix product in equation 13:

$$(\boldsymbol{A}_{\text{cap}} \otimes \boldsymbol{B}_{\text{cap}})(\boldsymbol{A}_{\text{edit}} \otimes \boldsymbol{B}_{\text{edit}})^{-1} = \boldsymbol{Q}_\otimes^{-\top} (\boldsymbol{D}_A \otimes \boldsymbol{D}_B) \boldsymbol{Q}_\otimes^\top. \tag{16}$$

Hence its adjustment maps a gradient to $\boldsymbol{Q}_\otimes^\top \boldsymbol{g}$, divides coordinate (i, j) by $1 + \lambda b_i a_j$, and maps back through $\boldsymbol{Q}_\otimes^{-\top}$. The transpose and inverse transpose differ because $\boldsymbol{Q}_\otimes$ is orthonormal in the edit-curvature metric, rather than in Euclidean coordinates. For a block gradient $\boldsymbol{g}_{\text{edit}} \in \mathbb{R}^{m \times n}$, the transformation of the gradient and its inverse take the matrix form

$$\widetilde{\boldsymbol{g}}_{\text{edit}} := \boldsymbol{Q}_B^\top \boldsymbol{g}_{\text{edit}} \boldsymbol{Q}_A, \qquad \boldsymbol{g}_{\text{edit}} = \boldsymbol{Q}_B^{-\top} \widetilde{\boldsymbol{g}}_{\text{edit}} \boldsymbol{Q}_A^{-1}.$$

This matrix form follows from $\operatorname{vec}(\boldsymbol{Q}_B^\top \boldsymbol{g}_{\text{edit}} \boldsymbol{Q}_A) = \boldsymbol{Q}_\otimes^\top \operatorname{vec}(\boldsymbol{g}_{\text{edit}})$. Applying this scaling and transforming back gives

$$\boldsymbol{g}_{\text{edit}}^{\text{adj}} = \boldsymbol{Q}_B^{-\top} \left[\frac{(\widetilde{\boldsymbol{g}}_{\text{edit}})_{ij}}{1 + \lambda b_i a_j} \right]_{ij} \boldsymbol{Q}_A^{-1}. \tag{17}$$

Appendix D gives the derivation of equation 17. For each edit, we estimate the capability factors once at $\boldsymbol{\theta}_0$ before editing and compute the edit factors and generalized bases once at $\boldsymbol{\theta}_0$. We then apply equation 17 at every optimizer step. As established in Section 3.1, this realizes the relative-curvature adjustment of the gradient; the chosen optimizer, rather than the explicit edit-curvature Newton preconditioner, converts the adjusted gradient into a parameter update.

Algorithm 1 summarizes our algorithm.

Algorithm 1 RHOEDIT

Require: Base parameters $\boldsymbol{\theta}_0$; capability set $\mathcal{D}_{\text{cap}}$; edit dataset $\mathcal{D}_{\text{edit}}$; penalty strength λ
Ensure: Edited parameters $\boldsymbol{\theta}$

1: Estimate the capability factors $(\boldsymbol{A}_{\text{cap}}^\ell, \boldsymbol{B}_{\text{cap}}^\ell)$ at $\boldsymbol{\theta}_0$ on $\mathcal{D}_{\text{cap}}$ for all fine-tuned layers ℓ
2: Estimate the edit factors $(\boldsymbol{A}_{\text{edit}}^\ell, \boldsymbol{B}_{\text{edit}}^\ell)$ at $\boldsymbol{\theta}_0$ on $\mathcal{D}_{\text{edit}}$ for all fine-tuned layers ℓ
3: Solve equation 14 for $(\boldsymbol{Q}_A^\ell, \boldsymbol{Q}_B^\ell)$ and the factor eigenvalues $\{a_j^\ell\}$, $\{b_i^\ell\}$
4: **for** each optimizer step on $\mathcal{D}_{\text{edit}}$ **do**
5: Adjust the edit gradient of each fine-tuned layer ℓ by equation 17 and update the parameters
6: **end for**
7: **return** the resulting parameters $\boldsymbol{\theta}$

Our K-FAC curvature estimation replaces a dense generalized eigenvalue problem of dimension mn with two problems of dimensions m and n, reducing the one-time decomposition cost from $O(m^3 n^3)$ to $O(m^3 + n^3)$. Applying the adjustment through the two smaller bases also reduces the cost per gradient from $O(m^2 n^2)$ to $O(m^2 n + m n^2)$ and the memory for curvature, bases, and block gradients from $O(m^2 n^2)$ to $O(m^2 + n^2 + mn)$.

4 EXPERIMENTS

4.1 SETUPS

We primarily focus on editing Llama-3-8B-Instruct. For cross-model evaluation, we use Qwen2.5-7B-Instruct and report the results in Appendix E.

Datasets and Evaluation. We conduct experiments on 3,000 samples of three standard model editing datasets: ZsRE (Levy et al., 2017), CounterFact (Meng et al., 2022), and WikiBigEdit (Thede et al., 2025). We report two standard model-editing metrics (De Cao et al., 2021; Yang et al., 2025a): *reliability* (also known as efficacy), which assesses whether the edited model produces an acceptable response to the original edit prompt, and *generalization*, which assesses whether the edit transfers to semantically related paraphrases of that prompt. We follow WILD (Yang et al., 2025a), which combines context-guided autoregressive decoding with LLM-as-a-judge scoring, and adopt it through EasyEdit (Wang et al., 2024b), evaluating prompts both with and without QA context. To measure capability degradation, we benchmark on MMLU (Hendrycks et al., 2020), IFEval (Zhou et al., 2023), TruthfulQA (Lin et al., 2022), ARC-Challenge (Clark et al., 2018), and GSM8K (Cobbe et al., 2021). We report our main comparison in Table 1.

Baselines. We compare RHOEDIT with eight editing methods. Standard fine-tuning (FT) directly optimizes model weights, LoRA learns low-rank updates (Hu et al., 2022; Zhang et al., 2023), and LocFT-BF fine-tunes a localized layer (Yang et al., 2025b). MEND maps gradients to parameter updates (Mitchell et al., 2022a), MEMIT writes associations into feed-forward weights (Meng et al., 2023), and UltraEdit uses hidden states and gradients for one-step updates (Gu et al., 2026). AlphaEdit projects edits into a null space estimated from preserved representations (Fang et al., 2025); CrispEdit projects gradients onto low-curvature directions of the capability loss (Ikram et al., 2026).

Table 1: **Llama-3-8B-Instruct, 3,000 edits.** Best and second-best are bold and underlined.

Data	Method	Edited Capabilities					Base Capabilities					
		QA Context		No Context		Avg.	MMLU	IFEval	TruthfulQA	ARC-C	GSM8K	Avg.
		Rel.	Gen.	Rel.	Gen.							
	LLaMA-3-8B-Instruct	1.9	2.0	3.4	3.3	2.7	69.5	69.3	52.8	59.2	76.4	65.4
ZsRE	FT	38.4	35.5	4.9	4.6	20.9	69.5	29.4	51.8	45.1	38.9	46.9
	LoRA	9.9	8.4	31.0	10.8	15.0	69.2	67.7	52.6	57.1	74.5	64.2
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	0.0	22.7	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	22.9	5.9	48.7	22.2	0.0	19.9
	AlphaEdit	71.3	61.9	48.3	38.7	55.1	52.2	43.6	44.2	47.1	46.6	46.7
	UltraEdit	20.7	16.9	24.5	18.5	20.2	69.2	67.1	52.7	56.9	75.8	64.3
	LocFT-BF	36.2	32.9	3.2	2.8	18.8	69.3	32.0	52.3	45.7	49.7	49.8
	CrispEdit	81.9	**70.4**	60.4	51.0	65.9	**69.7**	67.8	**53.0**	**57.3**	77.1	65.0
	RHOEDIT	**86.4**	69.6	**65.9**	**54.3**	**69.0**	69.4	**73.0**	52.5	56.5	**77.9**	**65.9**
	LLaMA-3-8B-Instruct	1.2	1.0	0.9	1.1	1.1	69.5	69.3	52.8	59.2	76.4	65.4
CounterFact	FT	10.9	4.5	1.0	1.0	4.4	68.1	20.3	51.3	37.4	21.6	39.7
	LoRA	15.2	8.9	25.9	6.4	14.1	69.3	66.4	**52.4**	54.8	76.5	63.9
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	22.7	0.0	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	24.6	11.3	48.1	22.1	0.0	21.2
	AlphaEdit	81.2	**67.0**	**77.8**	**67.9**	**73.5**	47.6	25.3	38.0	42.8	42.8	39.3
	UltraEdit	19.1	13.1	19.1	14.4	16.4	69.2	65.8	50.2	56.5	76.9	63.7
	LocFT-BF	24.7	13.8	0.9	2.8	10.6	68.5	20.0	50.9	38.0	20.6	39.6
	CrispEdit	86.0	59.3	57.5	49.0	63.0	**69.4**	65.4	52.0	**58.2**	76.9	**64.4**
	RHOEDIT	**89.2**	63.6	57.2	53.7	65.9	69.3	**68.4**	51.3	55.2	**77.8**	**64.4**
	LLaMA-3-8B-Instruct	8.6	9.2	18.1	17.7	13.4	69.5	69.3	52.8	59.2	76.4	65.4
WikiBigEdit	FT	34.0	32.2	6.0	6.9	19.8	**69.4**	39.0	53.0	48.6	33.4	48.7
	LoRA	32.9	28.0	48.3	32.0	35.3	**69.4**	69.0	52.3	56.0	74.9	64.3
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	22.7	0.0	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	24.5	48.2	3.9	21.6	0.0	19.6
	AlphaEdit	75.9	69.4	**77.1**	**67.6**	**72.5**	57.7	54.7	43.3	52.0	60.2	53.6
	UltraEdit	63.5	59.1	56.9	57.8	59.3	69.2	68.6	**54.0**	58.9	**76.7**	**65.5**
	LocFT-BF	27.2	27.0	5.7	6.0	16.5	69.0	38.6	51.9	46.6	56.3	52.5
	CrispEdit	**81.4**	73.5	35.4	39.0	57.3	69.2	67.1	51.9	**59.1**	76.0	64.7
	RHOEDIT	77.6	**73.7**	70.0	64.9	71.6	69.0	**70.2**	52.1	56.7	76.5	64.9

4.2 Main Results

Across the three datasets, we observe two patterns. First, sacrificing base capabilities does not guarantee effective editing: FT and LocFT-BF substantially degrade these capabilities yet achieve limited editing success, while MEND and MEMIT exhibit severe degradation and fail on both edit reliability and generalization. AlphaEdit achieves strong overall editing performance but also substantially degrades base capabilities. Second, methods that largely preserve base capabilities, including LoRA, UltraEdit, and CrispEdit, leave room for better editing; CrispEdit is strongest among these on ZsRE and CounterFact, whereas UltraEdit leads on WikiBigEdit. In comparison, **RhoEdit achieves the strongest overall editing performance among methods that largely preserve base capabilities across all three datasets**. It leads all methods on ZsRE and trails only AlphaEdit on CounterFact and WikiBigEdit, while keeping overall capability close to that of the unedited model.

4.3 Ablations and Analysis

We probe how much capability data is needed to estimate capability curvature and how the strength of its adjustment affects editing. Both sweeps use Llama-3-8B-Instruct on 3,000 ZsRE edits, with the remaining settings fixed as in Appendix F.

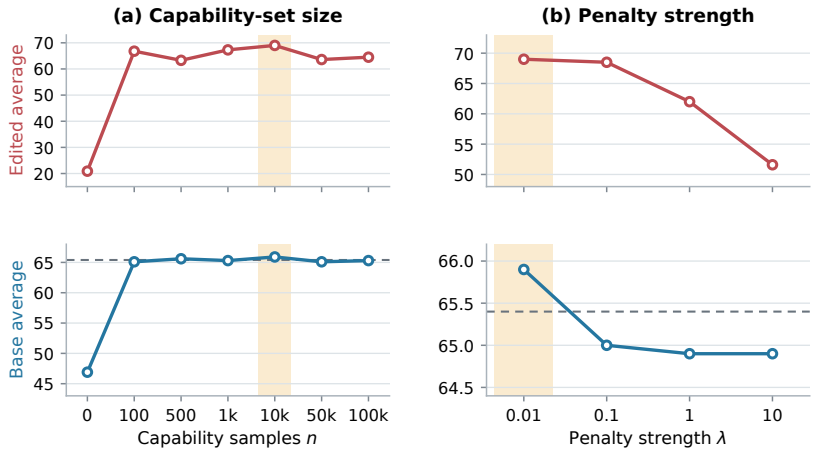

Figure 2: **Capability-set size and penalty-strength sweeps on 3,000 ZsRE edits.** The upper panels show the mean of the four WILD edit metrics; the lower panels show the mean of five general-capability benchmarks. Higher is better. The dashed line marks the unedited model's base average, and shaded positions mark the settings used in Table 1. In (a), $n = 0$ removes capability curvature and reproduces the FT result.

Capability-set size. To determine whether preservation depends on a large capability set, we compare nested reference sets in Figure 2(a). Both editing and base-capability averages recover from the unconstrained FT result even with the smallest nonzero set. Across larger sets, base capability remains close to the unedited model, while the editing performance fluctuates without a consistent gain from adding more samples. Each larger set contains the smaller ones, so the sweep tracks the amount of capability data under a fixed corpus ordering. The contrast with $n = 0$ supports the role of the capability curvature in this setting; the relatively flat results suggest that its estimate need not use the large capability set.

Penalty strength. Figure 2(b) tests whether stronger attenuation improves preservation enough to justify its effect on editing. Increasing λ lowers the editing performance, with the clearest decline at stronger penalties, while the base-capability average remains near the unedited reference throughout. The stronger adjustments therefore give up editing performance without a corresponding gain in average capability preservation over the sweep range. The reported setting lies at the high-editing end of this sweep while retaining base capability.

Table 2: **Scaling Llama-3-8B-Instruct to 10,000 ZsRE edits.** The unedited model is shown only as a reference. Best and second-best scores among editing methods are bold and underlined.

Method	Edited Capabilities					Base Capabilities					
	QA Context		No Context		Avg.	MMLU	IFEval	TruthfulQA	ARC-C	GSM8K	Avg.
	Rel.	Gen.	Rel.	Gen.							
LLaMA-3-8B-Instruct	1.9	2.0	3.4	3.3	2.7	69.5	69.3	52.8	59.2	76.4	65.4
AlphaEdit	0.3	0.4	0.8	0.7	0.6	24.2	13.9	47.8	23.0	1.7	22.1
LocFT-BF	4.1	3.4	4.1	3.3	3.7	65.4	10.9	47.0	31.7	0.5	31.1
CrispEdit	<u>41.9</u>	<u>37.2</u>	<u>22.4</u>	<u>19.2</u>	<u>30.2</u>	<u>68.4</u>	<u>21.3</u>	**53.4**	<u>36.5</u>	<u>8.4</u>	<u>37.6</u>
RHOEDIT	**85.2**	**74.7**	**48.1**	**42.0**	**62.5**	**69.1**	**68.4**	<u>51.8</u>	**54.7**	**77.0**	**64.2**

Scaling to 10,000 edits. We extend the Llama-3-8B-Instruct ZsRE experiment from 3,000 to 10,000 edits, holding the other training and evaluation settings fixed, to test whether the editing–preservation balance persists at a larger scale. Table 2 shows that AlphaEdit and LocFT-BF lose most of their editing success, while CrispEdit retains some editing performance but substantially degrades base capabilities. RHOEDIT leads all tested baselines on all four edit metrics while keeping its base-capability average close to that of the unedited model.

5 RELATED WORK

Model editing paradigms. Model editing methods differ in where they store an edit and how they compute the update. Memory-based editors attach auxiliary models, memories, or adapters and route relevant inputs through them (Dong et al., 2022; Mitchell et al., 2022b; Hartvigsen et al., 2023; Yu et al., 2024; Wang et al., 2024a). Meta-learned editors predict parameter changes from edit examples or gradients (Mitchell et al., 2022a; Tan et al., 2024; Li et al., 2025). Locate-then-edit methods identify layers or representations associated with a fact and modify selected weights (Dai et al., 2022; Meng et al., 2022; 2023; Pan et al., 2025). Fine-tuning methods directly optimize model parameters for the requested behavior (Zhu et al., 2020; Wang et al., 2021; Yang et al., 2025b).

Capability-preserving fine-tuning. Fine-tuning-based editors optimize an edit loss while limiting changes to existing behavior through weight regularization (Zhu et al., 2020), restricted layers (Yang et al., 2025b), or projected gradients (Wang et al., 2021). Layer restrictions determine where an update can occur; importance penalties and projections use estimates of which changes might harm retained behavior. Related continual-learning methods assign larger penalties to parameters important for earlier tasks, using Fisher information in elastic weight consolidation (Kirkpatrick et al., 2017) or accumulated parameter contributions in synaptic intelligence (Zenke et al., 2017). Among editing methods, CrispEdit projects gradients onto low-curvature directions of a capability objective (Ikram et al., 2026); AlphaEdit applies a null-space projection to a locate-then-edit update (Fang et al., 2025). These projection methods select admissible directions using preservation-side information. As a fine-tuning method, RHOEDIT also accounts for the local curvature of the edit loss. Derived from a constrained editing objective, its adjustment scales gradient components by capability curvature relative to edit curvature.

6 CONCLUSION

Effective editing and capability preservation can coexist when a model update is judged by both the requested change and the behavior it may disturb. We formalized this balance as a constrained editing problem and derived RHOEDIT to adjust the edit gradient using capability curvature relative to edit curvature. The adjustment protects sensitive existing behavior while keeping directions that remain useful for the edit available. A scalable curvature approximation makes this principle practical for LLMs. Across our evaluations, RHOEDIT makes effective edits while measured general capabilities remain close to their original level, including when the number of edits grows. Together, these results suggest that preserving a model's broader usefulness requires considering what each update can accomplish alongside what it might disrupt.

AI USE STATEMENT

We used generative AI tools to draft portions of the manuscript and to edit the writing for clarity, grammar, and style. The authors provided the scientific content, checked all AI-assisted text against the underlying analysis and experimental results, and revised it as needed. We did not use generative AI for research tasks requiring disclosure under the ICLR 2027 AI Policy for Authors. All authors reviewed the final manuscript and take responsibility for its content, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or crowdsourcing. Experiments edit released instruction-tuned models on the public benchmarks, and the paper does not introduce a new dataset. RHOEDIT is intended to apply a requested factual or behavioral edit while limiting damage to unrelated capabilities. The same update can insert false information or change safety-relevant behavior when the requested edits are themselves harmful. Reported scores measure editing efficacy and retention of general capabilities on the stated benchmarks.

REFERENCES

Lev M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

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.

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

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*, 2021.

Qingxiu Dong, Damai Dai, Yifan Song, Jingjing Xu, Zhifang Sui, and Lei Li. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 5937–5947, 2022.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker factored eigenbasis. *Advances in neural information processing systems*, 31, 2018.

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.

Xiaojie Gu, Ziying Huang, Jia-Chen Gu, and Kai Zhang. Ultraedit: Training-, subject-, and memory-free lifelong editing in language models. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=GoJLp3BlRV.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Zarif Ikram, Arad Firouzkouhi, Stephen Tu, Mahdi Soltanolkotabi, and Paria Rashidinejad. Crispedit: Low-curvature projections for scalable non-destructive LLM editing. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gTHlB3WK0l.

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.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=1jUXprrfcb.

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)*, pp. 3214–3252, 2022.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020. URL http://jmlr.org/papers/v21/17-678.html.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

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, 2022.

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

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.

Haowen Pan, Xiaozhi Wang, Yixin Cao, Zenglin Shi, Xun Yang, Juanzi Li, and Meng Wang. Precise localization of memories: A fine-grained neuron-level knowledge editing technique for LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5xP1HDvpXI.

Nicol N. Schraudolph. Fast curvature matrix–vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. doi: 10.1162/08997660260028683.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. Editable neural networks. In *International Conference on Learning Representations*, 2020.

Chenmien Tan, Ge Zhang, and Jie Fu. Massive editing for large language models via meta learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=L6L1CJQ2PE.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. WikiBigEdit: Understanding the limits of lifelong knowledge editing in LLMs. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 59326–59354. PMLR, 13–19 Jul 2025.

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.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, et al. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 82–93, 2024b.

Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 184–193, 2021.

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, 2024c.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. The mirage of model editing: Revisiting evaluation in the wild. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15336–15354, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.745. URL https://aclanthology.org/2025.acl-long.745/.

Wanli Yang, Fei Sun, Rui Tang, Hongyu Zang, Du Su, Qi Cao, Jingang Wang, Huawei Shen, and Xueqi Cheng. Fine-tuning done right in model editing. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025b. URL https://openreview.net/forum?id=lJFPwtobkG.

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 *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=NZZB3UGcd8.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. MELO: Enhancing model editing with neuron-indexed dynamic LoRA. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19449–19457, 2024.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

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 Bregman Discrepancies

For a twice continuously differentiable convex function ϕ on the output space, the associated Bregman divergence is

$$d_\phi(\boldsymbol{u}, \boldsymbol{v}) = \phi(\boldsymbol{u}) - \phi(\boldsymbol{v}) - \nabla\phi(\boldsymbol{v})^\top(\boldsymbol{u} - \boldsymbol{v}). \tag{18}$$

Substituting $\boldsymbol{u} = \boldsymbol{v}$ gives $d_\phi(\boldsymbol{v}, \boldsymbol{v}) = 0$. Differentiating in the first argument gives $\nabla_{\boldsymbol{u}} d_\phi(\boldsymbol{u}, \boldsymbol{v}) = \nabla\phi(\boldsymbol{u}) - \nabla\phi(\boldsymbol{v})$, which vanishes at $\boldsymbol{u} = \boldsymbol{v}$, and $\nabla^2_{\boldsymbol{u}} d_\phi(\boldsymbol{u}, \boldsymbol{v}) = \nabla^2\phi(\boldsymbol{u})$, which is positive semidefinite at $\boldsymbol{u} = \boldsymbol{v}$ by convexity of ϕ. Equation 18 therefore satisfies Assumption 1 for any such ϕ, with no requirement that $\boldsymbol{\theta}_0$ be a stationary point of a pretraining or proxy loss.

Two instantiations are relevant here. The quadratic function $\phi(\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|_2^2$ yields $d_\phi(\boldsymbol{u}, \boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2$. For categorical outputs with $\boldsymbol{z}$ the logit vector and $p_{\boldsymbol{z}} = \mathrm{softmax}(\boldsymbol{z})$, the log-sum-exp function $\phi(\boldsymbol{z}) = \log \sum_k \exp(z_k)$ yields $d_\phi(\boldsymbol{z}_{\boldsymbol{\theta}}, \boldsymbol{z}_{\boldsymbol{\theta}_0}) = D_{\mathrm{KL}}(p_{\boldsymbol{\theta}_0} \| p_{\boldsymbol{\theta}})$, which is the capability discrepancy used in our implementation.

B Analysis of the Relative-Curvature Adjustment

Let $\boldsymbol{C}_{\mathrm{edit}} \succ 0$ and $\boldsymbol{C}_{\mathrm{cap}} \succeq 0$ be symmetric curvature matrices. To identify directions along which $\mathcal{A}$ scales gradient components independently, consider a nonzero direction $\boldsymbol{v}$ satisfying

$$\boldsymbol{v}^\top \mathcal{A}\boldsymbol{g} = \alpha \boldsymbol{v}^\top \boldsymbol{g} \qquad \text{for every } \boldsymbol{g}. \tag{19}$$

This condition is equivalent to $\boldsymbol{v}^\top \mathcal{A} = \alpha \boldsymbol{v}^\top$. For $\lambda = 0$, $\mathcal{A} = \boldsymbol{I}$ and every gradient component is unchanged. For $\lambda > 0$, since $\mathcal{A}^{-1} = \boldsymbol{I} + \lambda \boldsymbol{C}_{\mathrm{cap}} \boldsymbol{C}_{\mathrm{edit}}^{-1}$, the condition holds exactly when

$$\boldsymbol{v}^\top \boldsymbol{C}_{\mathrm{cap}} \boldsymbol{C}_{\mathrm{edit}}^{-1} = \rho \boldsymbol{v}^\top, \qquad \alpha = (1 + \lambda\rho)^{-1}. \tag{20}$$

Using the symmetry of the curvature matrices and multiplying by $\boldsymbol{C}_{\mathrm{edit}}$, the left-eigenvector condition in equation 20 is equivalent to the generalized eigenvalue problem

$$\boldsymbol{C}_{\mathrm{cap}} \boldsymbol{v} = \rho \boldsymbol{C}_{\mathrm{edit}} \boldsymbol{v}. \tag{21}$$

Because $\boldsymbol{C}_{\mathrm{edit}} \succ 0$, set $\boldsymbol{q} := \boldsymbol{C}_{\mathrm{edit}}^{1/2} \boldsymbol{v}$. Then equation 21 is equivalent to the ordinary eigenvalue problem $\boldsymbol{R}\boldsymbol{q} = \rho \boldsymbol{q}$ for

$$\boldsymbol{R} := \boldsymbol{C}_{\mathrm{edit}}^{-1/2} \boldsymbol{C}_{\mathrm{cap}} \boldsymbol{C}_{\mathrm{edit}}^{-1/2}. \tag{22}$$

Since $\boldsymbol{R}$ is symmetric and positive semidefinite, it admits the eigendecomposition

$$\boldsymbol{R} = \boldsymbol{Q}\,\mathrm{diag}(\rho_i)\boldsymbol{Q}^\top, \qquad \boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}, \qquad \rho_i \geq 0. \tag{23}$$

For the corresponding parameter direction $\boldsymbol{v}_i := \boldsymbol{C}_{\mathrm{edit}}^{-1/2} \boldsymbol{q}_i$, where $\boldsymbol{q}_i$ is the ith column of $\boldsymbol{Q}$, we have

$$\boldsymbol{v}_i^\top \boldsymbol{C}_{\mathrm{edit}} \boldsymbol{v}_i = 1, \qquad \boldsymbol{v}_i^\top \boldsymbol{C}_{\mathrm{cap}} \boldsymbol{v}_i = \rho_i, \qquad \rho_i = \frac{\boldsymbol{v}_i^\top \boldsymbol{C}_{\mathrm{cap}} \boldsymbol{v}_i}{\boldsymbol{v}_i^\top \boldsymbol{C}_{\mathrm{edit}} \boldsymbol{v}_i}. \tag{24}$$

Each ρ_i therefore measures capability curvature relative to edit curvature along $\boldsymbol{v}_i$.

B.1 Effect of the Penalty Strength on Capability Cost

Writing $\Delta\boldsymbol{\theta} = \boldsymbol{C}_{\text{edit}}^{-1/2}\boldsymbol{Q}\boldsymbol{t}$ and $\boldsymbol{u} := \boldsymbol{Q}^\top \boldsymbol{C}_{\text{edit}}^{-1/2}\boldsymbol{g}_{\text{edit}}$, the local quadratic objective with curvature pair $(\boldsymbol{C}_{\text{edit}}, \boldsymbol{C}_{\text{cap}})$ becomes

$$\sum_i \left[u_i t_i + \frac{1}{2}(1+\lambda\rho_i) t_i^2 \right]. \tag{25}$$

Its minimizer satisfies $t_i^\star(\lambda) = -u_i/(1+\lambda\rho_i)$. Without the capability penalty, the edit-only Newton step has $t_i^\star(0) = -u_i$. The penalty therefore retains a fraction $(1+\lambda\rho_i)^{-1}$ of that step in each coordinate. The linear coefficient u_i determines the edit-only step, while ρ_i determines its attenuation.

The capability cost attained by this local minimizer is

$$\varepsilon(\lambda) := \tfrac{1}{2}\Delta\boldsymbol{\theta}^\star(\lambda)^\top \boldsymbol{C}_{\text{cap}} \Delta\boldsymbol{\theta}^\star(\lambda) = \frac{1}{2}\sum_i \frac{\rho_i u_i^2}{(1+\lambda\rho_i)^2}. \tag{26}$$

Every summand is nonincreasing in λ, and strictly decreasing whenever $\rho_i u_i^2 > 0$. Thus, $\varepsilon(\lambda)$ is continuous and nonincreasing, starts at the unpenalized cost $\varepsilon(0) = \frac{1}{2}\sum_i \rho_i u_i^2$, and tends to zero as $\lambda \to \infty$. Every budget in $(0, \varepsilon(0)]$ is therefore attained by some $\lambda \geq 0$, which can be found by a scalar root search. In practice, we select λ directly because the target budget is not known in advance. This correspondence concerns the minimizer of the local quadratic model; it does not assert the same attained cost for updates produced by a different optimizer.

B.2 Proof of Proposition 1

Using the definition of $\boldsymbol{R}$ gives

$$\begin{aligned} \boldsymbol{I} + \lambda \boldsymbol{C}_{\text{cap}}\boldsymbol{C}_{\text{edit}}^{-1} &= \boldsymbol{C}_{\text{edit}}^{1/2}(\boldsymbol{I} + \lambda\boldsymbol{R})\boldsymbol{C}_{\text{edit}}^{-1/2}, \\ (\boldsymbol{I} + \lambda\boldsymbol{R})^{-1} &= \boldsymbol{Q}\,\mathrm{diag}\big((1+\lambda\rho_i)^{-1}\big)\,\boldsymbol{Q}^\top. \end{aligned} \tag{27}$$

Inverting the first identity and substituting the second yields

$$\mathcal{A} = \boldsymbol{C}_{\text{edit}}^{1/2}\boldsymbol{Q}\,\mathrm{diag}\big((1+\lambda\rho_i)^{-1}\big)\,\boldsymbol{Q}^\top \boldsymbol{C}_{\text{edit}}^{-1/2}. \tag{28}$$

Multiplying by $\boldsymbol{g}$ proves equation 6.

C Curvature Model: Gauss–Newton and Fisher Curvature

C.1 The Capability Hessian Coincides with the GGN Matrix

Write $d_x(\boldsymbol{z}) := d(\boldsymbol{z}, \boldsymbol{z}_{\boldsymbol{\theta}_0}(x))$ and $\boldsymbol{J}_{\boldsymbol{\theta}}(x) := \partial \boldsymbol{z}_{\boldsymbol{\theta}}(x)/\partial\boldsymbol{\theta}$. Applying the second-order chain rule to the empirical capability loss gives

$$\nabla_{\boldsymbol{\theta}}^2 \mathcal{R}_{\text{cap}} = \mathbb{E}_{x\sim\mathcal{D}_{\text{cap}}}\left[\boldsymbol{J}_{\boldsymbol{\theta}}^\top (\nabla_{\boldsymbol{z}}^2 d_x)\boldsymbol{J}_{\boldsymbol{\theta}} + \sum_k (\partial_{z_k} d_x)\nabla_{\boldsymbol{\theta}}^2 z_{\boldsymbol{\theta},k}(x) \right]. \tag{29}$$

At $\boldsymbol{\theta}_0$, Assumption 1 makes every coefficient $\partial_{z_k} d_x$ zero, regardless of the second derivatives of the model outputs. Writing $\boldsymbol{J}_0 = \boldsymbol{J}_{\boldsymbol{\theta}_0}$ and $\boldsymbol{H}_z(x) = \nabla_{\boldsymbol{z}}^2 d_x|_{\boldsymbol{z}=\boldsymbol{z}_{\boldsymbol{\theta}_0}(x)}$, this yields

$$\boldsymbol{H}_{\text{cap}} = \mathbb{E}_{x\sim\mathcal{D}_{\text{cap}}}[\boldsymbol{J}_0^\top \boldsymbol{H}_z(x)\boldsymbol{J}_0] = \boldsymbol{G}_{\text{cap}} \succeq \boldsymbol{0}. \tag{30}$$

This identity holds for every discrepancy satisfying the assumption; equality with the model Fisher additionally uses the categorical logit instance below. For the edit loss, the logit gradient is $\mathrm{softmax}(\boldsymbol{z}) - \boldsymbol{e}_y$, where $\boldsymbol{e}_y$ is the one-hot target vector. Its residual generally remains, so replacing the edit Hessian by its GGN is a curvature approximation.

C.2 Gauss–Newton Curvature Equals the Model Fisher

For position t in reference text x_j^{c}, write $p_{j,t}(\boldsymbol{\theta}) := \mathrm{softmax}(\boldsymbol{z}_{j,t}(\boldsymbol{\theta}))$ and $d_{j,t}(\boldsymbol{z}) := D_{\text{KL}}(p_{j,t}(\boldsymbol{\theta}_0) \,\|\, \mathrm{softmax}(\boldsymbol{z}))$. For a categorical distribution parameterized by logits, the edit negative log-likelihood $\ell_y(\boldsymbol{z}) = -z_y + \log\sum_k e^{z_k}$ and the forward KL divergence $d_{j,t}$ have the same output-space Hessian,

$$\nabla_{\boldsymbol{z}}^2 \ell_y(\boldsymbol{z}) = \nabla_{\boldsymbol{z}}^2 d_{j,t}(\boldsymbol{z}) = \mathrm{Diag}(\boldsymbol{s}) - \boldsymbol{s}\boldsymbol{s}^\top, \qquad \boldsymbol{s} := \mathrm{softmax}(\boldsymbol{z}). \tag{31}$$

The right-hand side is the Fisher information of the categorical distribution with respect to its natural parameter, and it depends on neither the label y nor the reference distribution. Propagating it through $\boldsymbol{J}_{\boldsymbol{\theta}}(x)$ therefore produces the model Fisher information matrix in both cases, which gives both equalities in equation 9:

$$\begin{aligned}\boldsymbol{G} &= \mathbb{E}_x\left[\boldsymbol{J}_{\boldsymbol{\theta}}(x)^\top \left(\mathrm{Diag}(\boldsymbol{s}) - \boldsymbol{s}\boldsymbol{s}^\top\right) \boldsymbol{J}_{\boldsymbol{\theta}}(x)\right] \\ &= \mathbb{E}_x\, \mathbb{E}_{\tilde{y}\sim p_{\boldsymbol{\theta}}(\cdot|x)}\left[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tilde{y} \mid x)\, \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\tilde{y} \mid x)^\top\right] = \boldsymbol{F},\end{aligned} \tag{32}$$

with the expectation over x taken on the edit data for $(\boldsymbol{G}_{\mathrm{edit}}, \boldsymbol{F}_{\mathrm{edit}})$ and on the capability data for $(\boldsymbol{G}_{\mathrm{cap}}, \boldsymbol{F}_{\mathrm{cap}})$. The inner label expectation is taken under the model's predictive distribution at the curvature-evaluation point, so these matrices are model Fishers rather than empirical Fishers formed from requested or observed labels (Kunstner et al., 2019). Equation 32 is what licenses the sampled-label estimator in Appendix D: both curvature factors can be estimated from labels drawn under the model itself, without reference to the requested edit targets.

For reference text j with L_j evaluated positions, let $\boldsymbol{J}_{j,t} := \nabla_{\boldsymbol{\theta}} \boldsymbol{z}_{j,t}(\boldsymbol{\theta}_0)$ and $\boldsymbol{C}_{j,t} := \mathrm{Diag}(p_{j,t}(\boldsymbol{\theta}_0)) - p_{j,t}(\boldsymbol{\theta}_0) p_{j,t}(\boldsymbol{\theta}_0)^\top$. Averaging the forward-KL curvature over these positions gives

$$\boldsymbol{H}_{\mathrm{cap}} = \boldsymbol{G}_{\mathrm{cap}} = \boldsymbol{F}_{\mathrm{cap}} = \frac{1}{\sum_{j=1}^{N} L_j} \sum_{j=1}^{N} \sum_{t=1}^{L_j} \boldsymbol{J}_{j,t}^\top \boldsymbol{C}_{j,t} \boldsymbol{J}_{j,t}. \tag{33}$$

D K-FAC Realization and Numerical Details

D.1 Curvature Factors and Their Estimation

For a weight block $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ with input activation $\boldsymbol{a}$ and output-side gradient $\boldsymbol{\delta}$, the K-FAC factors in equation 12 are the second-moment matrices

$$\boldsymbol{A} = \mathbb{E}\left[\boldsymbol{a}\boldsymbol{a}^\top\right] \in \mathbb{R}^{n\times n}, \qquad \boldsymbol{B} = \mathbb{E}\left[\boldsymbol{\delta}\boldsymbol{\delta}^\top\right] \in \mathbb{R}^{m\times m}. \tag{34}$$

By equation 32, $\boldsymbol{\delta}$ is computed from next-token labels sampled from the model's own predictive distribution at the point where curvature is evaluated, for the edit and the capability factors alike. This sampling enters only the curvature estimates; the gradient that the adjustment acts on remains the gradient of the edit loss defined in Section 2.

D.2 Separability of the Relative Curvature

For nonsingular edit factors, the Kronecker inverse and mixed-product identities give $(\boldsymbol{A}_{\mathrm{edit}} \otimes \boldsymbol{B}_{\mathrm{edit}})^{-1} = \boldsymbol{A}_{\mathrm{edit}}^{-1} \otimes \boldsymbol{B}_{\mathrm{edit}}^{-1}$ and hence the approximation in equation 13. The generalized eigensystems in equation 14 make the edit factors identity matrices and the capability factors diagonal in their respective bases. For $\boldsymbol{Q}_{\otimes} = \boldsymbol{Q}_A \otimes \boldsymbol{Q}_B$, the mixed-product identity gives

$$\begin{aligned}\boldsymbol{Q}_{\otimes}^\top (\boldsymbol{A}_{\mathrm{edit}} \otimes \boldsymbol{B}_{\mathrm{edit}}) \boldsymbol{Q}_{\otimes} &= (\boldsymbol{Q}_A^\top \boldsymbol{A}_{\mathrm{edit}} \boldsymbol{Q}_A) \otimes (\boldsymbol{Q}_B^\top \boldsymbol{B}_{\mathrm{edit}} \boldsymbol{Q}_B) = \boldsymbol{I}, \\ \boldsymbol{Q}_{\otimes}^\top (\boldsymbol{A}_{\mathrm{cap}} \otimes \boldsymbol{B}_{\mathrm{cap}}) \boldsymbol{Q}_{\otimes} &= (\boldsymbol{Q}_A^\top \boldsymbol{A}_{\mathrm{cap}} \boldsymbol{Q}_A) \otimes (\boldsymbol{Q}_B^\top \boldsymbol{B}_{\mathrm{cap}} \boldsymbol{Q}_B) = \boldsymbol{D}_A \otimes \boldsymbol{D}_B.\end{aligned} \tag{35}$$

These exact identities for the Kronecker factors yield the approximate relations in equation 15 for the original curvatures. The diagonal entries $b_i a_j$ are the relative curvatures of Proposition 1 for the K-FAC approximation.

D.3 Derivation of the Elementwise Adjustment

Index the block parameters by their row–column pairs (i, j). The entry of $\boldsymbol{A} \otimes \boldsymbol{B}$ coupling (i, j) with (k, ℓ) is $A_{j\ell} B_{ik}$, so its action on a block-shaped matrix $\boldsymbol{X}$ has entries

$$\sum_{k=1}^{m} \sum_{\ell=1}^{n} B_{ik} A_{j\ell} X_{k\ell} = (\boldsymbol{B}\boldsymbol{X}\boldsymbol{A}^\top)_{ij}. \tag{36}$$

Multiplying equation 10 by $\boldsymbol{I} + \lambda \boldsymbol{G}_{\mathrm{cap}} \boldsymbol{G}_{\mathrm{edit}}^{-1}$ and substituting the K-FAC approximation in equation 13 therefore yields the following matrix equation for the implemented adjustment:

$$\boldsymbol{g}_{\mathrm{edit}}^{\mathrm{adj}} + \lambda \boldsymbol{B}_{\mathrm{cap}} \boldsymbol{B}_{\mathrm{edit}}^{-1} \boldsymbol{g}_{\mathrm{edit}}^{\mathrm{adj}} \boldsymbol{A}_{\mathrm{edit}}^{-1} \boldsymbol{A}_{\mathrm{cap}} = \boldsymbol{g}_{\mathrm{edit}}. \tag{37}$$

The generalized eigensystems in equation 14 imply

$$\begin{aligned} \boldsymbol{A}_{\text{edit}} &= \boldsymbol{Q}_A^{-\top}\boldsymbol{Q}_A^{-1}, \quad \boldsymbol{A}_{\text{cap}} = \boldsymbol{Q}_A^{-\top}\,\text{diag}(a_j)\boldsymbol{Q}_A^{-1}, \\ \boldsymbol{B}_{\text{edit}} &= \boldsymbol{Q}_B^{-\top}\boldsymbol{Q}_B^{-1}, \quad \boldsymbol{B}_{\text{cap}} = \boldsymbol{Q}_B^{-\top}\,\text{diag}(b_i)\boldsymbol{Q}_B^{-1}. \end{aligned} \tag{38}$$

Let $\boldsymbol{S} := \boldsymbol{Q}_B^{\top}\boldsymbol{g}_{\text{edit}}^{\text{adj}}\boldsymbol{Q}_A$, so that $\boldsymbol{g}_{\text{edit}}^{\text{adj}} = \boldsymbol{Q}_B^{-\top}\boldsymbol{S}\boldsymbol{Q}_A^{-1}$. Substitute this expression and equation 38 into equation 37, then multiply on the left by $\boldsymbol{Q}_B^{\top}$ and on the right by $\boldsymbol{Q}_A$ to obtain

$$\boldsymbol{S} + \lambda \boldsymbol{D}_B \boldsymbol{S} \boldsymbol{D}_A = \boldsymbol{Q}_B^{\top}\boldsymbol{g}_{\text{edit}}\boldsymbol{Q}_A. \tag{39}$$

Because $\boldsymbol{D}_A$ and $\boldsymbol{D}_B$ are diagonal, this system separates entrywise:

$$(1 + \lambda b_i a_j)\, S_{ij} = \left(\boldsymbol{Q}_B^{\top}\boldsymbol{g}_{\text{edit}}\boldsymbol{Q}_A\right)_{ij}, \tag{40}$$

and solving for S_{ij} yields equation 17. The inverse basis transformations in equation 17 can be applied without explicitly inverting the generalized eigenvector matrices, because

$$\boldsymbol{Q}_B^{-\top} = \boldsymbol{B}_{\text{edit}}\boldsymbol{Q}_B, \qquad \boldsymbol{Q}_A^{-1} = \boldsymbol{Q}_A^{\top}\boldsymbol{A}_{\text{edit}}. \tag{41}$$

D.4 Numerical Procedure

We add positive damping to $\boldsymbol{A}_{\text{edit}}$ and $\boldsymbol{B}_{\text{edit}}$ before solving each generalized eigensystem, which makes them positive definite and bounds the resulting relative curvatures. To solve the input-side system, let $\boldsymbol{A}_{\text{edit}} = \boldsymbol{L}_A\boldsymbol{L}_A^{\top}$ be a Cholesky factorization and diagonalize the symmetric matrix $\boldsymbol{L}_A^{-1}\boldsymbol{A}_{\text{cap}}\boldsymbol{L}_A^{-\top}$; mapping its orthonormal eigenvectors through $\boldsymbol{L}_A^{-\top}$ produces $\boldsymbol{Q}_A$, and the output-side factors are handled analogously. This procedure preserves the symmetric structure and never forms $\boldsymbol{A}_{\text{cap}}\boldsymbol{A}_{\text{edit}}^{-1}$ or $\boldsymbol{B}_{\text{cap}}\boldsymbol{B}_{\text{edit}}^{-1}$ explicitly.

D.5 Computational Complexity

We compare the dense and K-FAC realizations for one weight block $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, with curvature fixed and the decomposition reused throughout an edit. The following costs concern the linear algebra after curvature estimation; data passes for estimating curvature and the storage of model parameters and optimizer state are excluded.

In the dense realization, each curvature matrix has size $mn \times mn$. Cholesky whitening followed by a symmetric eigendecomposition costs $O((mn)^3) = O(m^3n^3)$ once. Each subsequent adjustment applies the forward and inverse gradient-coordinate transformations, each a dense matrix–vector product in dimension mn, for a total of $O((mn)^2) = O(m^2n^2)$. The curvature matrices and bases require $O(m^2n^2)$ memory, which dominates the $O(mn)$ gradient storage.

With K-FAC, the corresponding Cholesky factorizations and eigendecompositions operate on the $m \times m$ and $n \times n$ factors, costing $O(m^3 + n^3)$ once. For the forward transformation $\boldsymbol{Q}_B^{\top}\boldsymbol{g}_{\text{edit}}\boldsymbol{Q}_A$, the left multiplication costs $O(m^2n)$ and the right multiplication costs $O(mn^2)$. The inverse transformation in equation 17 has the same cost, while elementwise attenuation costs $O(mn)$, giving $O(m^2n + mn^2)$ per adjustment. The factors and their bases occupy $O(m^2 + n^2)$ memory, and the gradient and intermediate matrices occupy $O(mn)$. Neither the full curvature matrices nor the joint basis $\boldsymbol{Q}_{\otimes}$ are formed, so the total auxiliary memory is $O(m^2 + n^2 + mn)$.

For a square block with $m = n = d$, K-FAC reduces the decomposition cost from $O(d^6)$ to $O(d^3)$, the cost per gradient from $O(d^4)$ to $O(d^3)$, and auxiliary memory from $O(d^4)$ to $O(d^2)$. Since the block contains d^2 weights, the last reduction makes the retained curvature state scale linearly with the parameter count.

E Cross-Model Evaluation

We apply the same 3,000 edits per dataset to Qwen2.5-7B-Instruct to evaluate editing performance on another model family. We compare RhoEdit with the same baselines as in Table 1: FT, LoRA, MEND, MEMIT, AlphaEdit, UltraEdit, LocFT-BF, and CrispEdit. All methods use the same judge, prompts, and capability benchmarks as the Llama-3-8B-Instruct experiment.

Table 3: **Cross-model evaluation on Qwen2.5-7B-Instruct with 3,000 edits per dataset.** The datasets, baselines, and evaluation metrics match Table 1. The unedited model is shown only as a reference. Best and second-best scores among editing methods are bold and underlined. The two Avg. columns are lightly shaded.

Data	Method	Edited Capabilities					Base Capabilities					
		QA Context		No Context		Avg.	MMLU	IFEval	TruthfulQA	ARC-C	GSM8K	Avg.
		Rel.	Gen.	Rel.	Gen.							
	Qwen2.5-7B-Instruct	4.3	3.9	3.5	2.8	3.6	75.6	72.1	63.0	58.6	76.1	69.1
ZsRE	FT	50.2	35.3	37.4	26.8	37.4	27.3	17.6	49.0	34.1	2.0	26.0
	LoRA	4.0	3.3	23.5	5.0	9.0	72.9	66.2	62.2	54.9	76.5	66.5
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	22.7	0.0	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	27.2	15.5	48.1	23.6	0.0	22.9
	AlphaEdit	29.7	25.1	19.0	16.2	22.5	41.8	44.7	52.1	37.3	51.8	45.5
	UltraEdit	14.4	12.3	21.3	14.2	15.6	**75.7**	**72.3**	**62.4**	**58.5**	77.0	**69.2**
	LocFT-BF	**79.6**	**64.9**	46.9	37.8	57.3	73.5	71.0	60.4	52.9	77.0	67.0
	CrispEdit	66.5	42.6	48.3	30.2	46.9	74.7	71.2	58.2	52.9	73.0	66.0
	RHOEDIT	75.6	49.6	**74.4**	**49.9**	**62.4**	75.4	70.4	53.9	55.0	**78.9**	66.7
	Qwen2.5-7B-Instruct	1.6	0.5	1.1	1.0	1.0	75.6	72.1	63.0	58.6	76.1	69.1
CounterFact	FT	27.3	9.2	10.6	7.4	13.6	33.4	29.6	55.4	29.9	1.1	29.9
	LoRA	3.7	1.9	21.8	4.4	8.0	73.6	65.1	**63.9**	57.6	73.5	66.7
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	22.7	0.0	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	24.6	14.6	46.4	22.4	0.0	21.6
	AlphaEdit	13.1	13.0	27.9	21.3	18.8	25.3	8.1	47.3	20.6	1.3	20.5
	UltraEdit	15.8	8.2	25.0	13.6	15.7	**75.6**	**72.6**	61.7	**58.5**	**77.0**	**69.1**
	LocFT-BF	**82.1**	**56.7**	**74.2**	**60.3**	**68.3**	72.2	70.6	59.6	56.3	76.4	67.0
	CrispEdit	37.2	12.9	70.3	20.8	35.3	74.5	69.1	59.7	57.1	73.4	66.8
	RHOEDIT	51.0	19.2	63.2	36.5	42.5	75.1	69.7	53.7	49.8	74.1	64.5
	Qwen2.5-7B-Instruct	19.8	19.3	15.3	16.6	17.8	75.6	72.1	63.0	58.6	76.1	69.1
WikiBigEdit	FT	32.2	26.0	36.0	34.2	32.1	36.9	33.1	52.2	42.1	15.5	35.9
	LoRA	25.2	22.9	61.4	33.4	35.7	73.7	66.9	62.4	**60.2**	**80.5**	68.7
	MEND	0.0	0.0	0.0	0.0	0.0	22.9	14.6	0.0	22.7	0.0	12.0
	MEMIT	0.0	0.0	0.0	0.0	0.0	24.6	14.8	48.1	22.3	0.0	21.9
	AlphaEdit	1.3	1.0	2.3	2.2	1.7	24.0	8.0	48.5	23.3	0.3	20.8
	UltraEdit	42.2	37.9	52.6	45.3	44.5	**75.5**	71.0	61.8	58.5	76.9	68.7
	LocFT-BF	**86.6**	**76.1**	75.8	**68.2**	**76.7**	74.3	71.9	61.7	59.0	78.3	69.1
	CrispEdit	55.7	48.5	62.5	57.2	56.0	**75.5**	69.7	61.4	58.1	71.1	67.2
	RHOEDIT	71.8	57.9	**76.0**	61.4	66.8	75.4	**75.6**	**63.5**	58.4	77.9	**70.1**

F EXPERIMENTAL DETAILS

Table 4 lists the optimization settings shared by both models and all three datasets.

Table 4: **Optimization settings shared by both models and all datasets.**

Setting	Value
Edited module	MLP down-projection
Optimizer	Adam
Training steps	25
Batch size	32
Capability set size	10,000
K-FAC damping	1×10^{-5}

For Llama-3-8B-Instruct, we edit layers 14–23. We set λ to 0.01, 0.1, and 1.0 on ZsRE, CounterFact, and WikiBigEdit, respectively. For Qwen2.5-7B-Instruct, we edit layers 12–19. We set λ to 0.01 on ZsRE and CounterFact and 0.1 on WikiBigEdit.

Judge prompt. Every generated continuation is scored by the prompt in Figure 3, which asks for a single binary verdict and admits no partial credit. The same prompt, judge model, and parser are used for every method, dataset, and evaluation condition, so any judge bias applies uniformly across the comparison. Outputs that cannot be parsed as one of the two verdicts are re-queried once and then counted as incorrect.

Prompt for LLM-as-a-Judge

Your job is to look at a question, a gold target, and a predicted answer, and then assign a grade → of either ["CORRECT", "INCORRECT"].

The following are examples of CORRECT predicted answers.

```
Question: What are the names of Barack Obama's children?
Gold target: Malia Obama and Sasha Obama
Predicted answer 1: sasha and malia obama
Predicted answer 2: Malia and Sasha Obama are the names of Barack
    Obama's children.
```

These predicted answers are all CORRECT because:

- They fully contain the important information in the gold target.
- They do not contain any information that contradicts the gold target.

The following are examples of INCORRECT predicted answers.

```
Question: What are the names of Barack Obama's children?
Gold target: Malia and Sasha
Predicted answer 1: Malia.
Predicted answer 2: Malia, Sasha, and Susan.
Predicted answer 3: Malia and Sasha, Malia and Sasha,
              Malia and Sasha, Malia and Sasha (repeated answer)
```

These predicted answers are all INCORRECT because:

- A factual statement in the answer contradicts the gold target or contains repeated content.

Here is a sample. Simply reply with either CORRECT or INCORRECT.

```
Question: {question}
Gold target: {target}
Predicted answer: {predicted_answer}
```

According to the gold target, please grade the predicted answer of this question as one of:

- A: CORRECT
- B: INCORRECT

Just return the letters "A" or "B", with no text around it.

Figure 3: **Binary correctness prompt given to DeepSeek-V4-Flash-Preview.**