FASTEDIT: LOW-RANK STRUCTURED REGULARIZATION FOR EFFICIENT MODEL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

When new knowledge emerges, it is crucial to efficiently update large language models (LLMs) to reflect the latest information. However, state-of-the-art methods widely adopted in the model editing community — such as MEMIT, PRUNE, and AlphaEdit — suffer from prohibitively slow editing speeds, often taking 6 to 14 hours to sequentially edit just 2000 facts on models like LLaMA-3-8B, making real-time updates impractical, especially as model scale increases. Moreover, they require extensive pre-computation to sample pre-edit knowledge — a step that can take over 24 hours — severely limiting their deployability. In this paper, we present **FastEdit**, a highly efficient editing framework that enables rapid and scalable model updates. Our key insight is to exploit the low-rank structure inherent in editing updates through a structured regularizer, allowing us to avoid costly inversions via the Sherman-Morrison-Woodbury (SMW) identity. This drastically accelerates the computation of update matrices while preserving edit quality. Crucially, **FastEdit** requires only a small number of pre-edit samples, reducing both memory and computational overhead. On 2000 sequential edits, **FastEdit** completes the process in just **1 hour** – an order of magnitude faster than prior work – without sacrificing accuracy. Our method significantly lowers the barrier to practical model editing, enabling timely and scalable knowledge updates in large models.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities in understanding and generating human language (Brown et al., 2020; Touvron et al., 2023). Yet, their knowledge remains largely static after training—updating even a single fact typically requires full retraining or incurs risks of corrupting unrelated knowledge (De Cao et al., 2021; Mitchell et al., 2022a; Meng et al., 2022; Zhao et al., 2023). This rigidity poses a fundamental challenge for applications in dynamic domains such as news, medicine, or education, where models must adapt quickly and precisely to new information (Leike et al., 2023; Vellal et al., 2024).

To enable fine-grained control over model knowledge, recent work has introduced *knowledge editing*: techniques that modify specific facts through localized weight updates while preserving general behavior (Meng et al., 2023; Ramesh et al., 2024; Gupta et al., 2024b; Fang et al., 2025). While conceptually appealing, these methods face two critical bottlenecks: **computational inefficiency** and **practical infeasibility**. Most approaches rely on expensive optimization procedures—such as inverting large $d \times d$ matrices (d: hidden dimension)—leading to $\mathcal{O}(d^3)$ time complexity per edit (Gupta et al., 2024a; Li & Chu, 2025; Ma et al., 2025). As a result, updating thousands of facts sequentially becomes impractical, especially for large-scale models. Moreover, many methods depend on extensive pre-computation using large sets of pre-edit data to estimate representation statistics. For example, collecting and processing samples for covariance estimation on LLaMA-3-8B (Meta, 2024) can take over 24 hours, severely limiting deployability (Meng et al., 2022;

2023; Ma et al., 2025). These costs stem from treating edits as dense, unstructured operations, without leveraging the underlying geometry of the model's latent space (Aghajanyan et al., 2021; Yu & Wu, 2023).

In this work, we ask: *Can we design a model editing framework that is both principled and truly efficient—enabling fast updates with low computational and pre-editing overhead?*

We propose **FastEdit**, a structure-aware editing framework that exploits the low-dimensional intrinsic subspace of pre-edit representations (Aghajanyan et al., 2021; Yu & Wu, 2023). Rather than performing updates in the full d-dimensional space, we model the key space with a low-rank plus diagonal structure, leading to an efficient closed-form update via the Sherman-Morrison-Woodbury (SMW) identity (Golub & Van Loan, 2013). This avoids $O(d^3)$ matrix inversion and reduces per-edit complexity to $\mathcal{O}(dr^2)$ time and $\mathcal{O}(dr)$ space, where $r \ll d$. For sequential editing, we further develop an incremental solver that maintains a low-rank summary of past updates, enabling constant-time updates with respect to the number of edits.

Our approach yields three key advances: (1) a principled alternative to dense update formulations through structured regularization; (2) a highly efficient closed-form update that drastically reduces computational overhead; and (3) a scalable, incremental solver for long-term knowledge integration. Experiments on counterfactual and factual editing benchmarks demonstrate that FastEdit achieves on-par or superior edit precision and generalization, while reducing the total time for 2000 sequential edits on Llama-3-8B from 6 to 14 hours across existing methods down to just 1 hour. These results highlight that exploiting the latent low-dimensional structure of neural representations provides a viable pathway toward efficient, reliable, and scalable model editing in practical settings.

2 Related Work

Knowledge editing aims to update specific factual knowledge in pre-trained language models without full retraining. Existing methods fall into two main paradigms. *Training-based* approaches construct tailored datasets to train auxiliary components for parameter updates. MEND (Mitchell et al., 2022a) and InstructEdit (Zhang et al., 2024) employ meta-learning to train hypernetworks that predict localized parameter modifications. SERAC (Mitchell et al., 2022b) introduces a memory-augmented architecture with a scope classifier and a counterfactual model to generate corrected outputs. T-Patcher (Huang et al., 2023) and MELO (Yu et al., 2024) insert feedforward memory modules to store and retrieve new factual associations during inference. *Memory-based* methods, a category of *training-free* editing, store edits externally and retrieve them at inference time via similarity-based lookup (Dong et al., 2022; Zheng et al., 2023; Hartvigsen et al., 2023; Jiang et al., 2024). These approaches decouple knowledge updates from model parameters, enabling efficient and reversible edits, where in-context learning is usually utilized (Bi et al., 2025). Another *training-free* paradigm is *Locate-then-edit*, which identifies and directly modifies knowledge-localized components within the model (Wang et al., 2024; Park et al., 2025; Gupta et al., 2023; Li et al., 2024a; Gupta et al., 2024b; 2025; Dai et al., 2025). ROME (Meng et al., 2022) pioneers this approach by modeling MLP layers as associative memory and applying causal tracing to guide rank-one weight updates. Subsequent work generalizes and improves this framework: MEMIT (Meng et al., 2023) and EMMET (Gupta et al., 2024b) extends ROME to batched editing for improved scalability, while AlphaEdit (Fang et al., 2025) constrains edits within the null space of existing knowledge representations to better preserve model integrity.

Several benchmarks have been introduced to assess not only the local correctness of edits but also their ability to support logical reasoning and generalization (Zhong et al., 2023; Gu et al., 2024a; Cohen et al., 2024), providing a more comprehensive evaluation of knowledge editing methods. As knowledge editing techniques advance and are applied in more complex or sequential scenarios, unintended side effects have increasingly come to light. These include knowledge conflict and distortion (Li et al., 2024b), gradual and catastrophic forgetting during large-scale editing (Gupta et al., 2024a), attenuation of edited knowledge over time (Li & Chu, 2024), overfitting (Zhang et al., 2025) and failure in lifelong editing due to knowledge su-

perposition in parameter space (Hu et al., 2025). To mitigate such issues, regularization strategies have been proposed. RECT (Gu et al., 2024b) restricts updates to a sparse subset of parameters, while PRUNE (Ma et al., 2025) and AdaEdit (Li & Chu, 2025) apply singular value decomposition (SVD) to preserve the dominant components of parameter changes, thereby enhancing stability and reducing interference.

3 PRELIMINARIES

We focus on the *locate-then-edit* framework for model editing, which aims to update specific knowledge in large language models (LLMs) by identifying and modifying relevant parameters. Recent studies have shown that factual knowledge is primarily stored in the feed-forward network (FFN) modules of Transformers (Geva et al., 2021). Further analysis via causal mediation has revealed that editing the second linear layer within the FFN of earlier Transformer blocks is particularly effective for knowledge update (Meng et al., 2022). Concretely, each such linear layer, parameterized by a weight matrix $\mathbf{W} \in \mathbb{R}^{\hat{d} \times d}$, associates input representations $\mathbf{k} \in \mathbb{R}^d$ with output vectors $\mathbf{v} \in \mathbb{R}^{\hat{d}}$, forming a key-value mapping that encodes knowledge.

To update a piece of knowledge, we seek a new output vector $\mathbf{v}'$ that produces the desired behavior. While $\mathbf{v}'$ can be learned via gradient-based optimization, the challenge lies in finding a parameter perturbation Δ such that the updated layer $\mathbf{W} + \Delta$ maps $\mathbf{k}$ to $\mathbf{v}'$, i.e., $(\mathbf{W} + \Delta)\mathbf{k} = \mathbf{v}'$. Crucially, we want this update to retain the model's original knowledge. That is, the perturbation Δ should introduce minimal interference to the model's pre-existing behavior on unrelated knowledge.

To formalize this, suppose we have b_0 pieces of preserved knowledge, encoded as input-output pairs $(\mathbf{K}_0, \mathbf{V}_0)$, where $\mathbf{K}_0 \in \mathbb{R}^{d \times b_0}$ and $\mathbf{V}_0 \in \mathbb{R}^{\hat{d} \times b_0}$ satisfy $\mathbf{W}\mathbf{K}_0 = \mathbf{V}_0$. Additionally, let b_1 new knowledge edits be represented by $(\mathbf{K}_1, \mathbf{V}_1)$, with $\mathbf{K}_1 \in \mathbb{R}^{d \times b_1}$ and $\mathbf{V}_1 \in \mathbb{R}^{\hat{d} \times b_1}$. We then formulate the update as an optimization problem that balances faithful editing with knowledge preservation (Meng et al., 2023):

$$\Delta = \arg\min_{\tilde{\Delta}} \left(\left\| (\mathbf{W} + \tilde{\Delta})\mathbf{K}_1 - \mathbf{V}_1 \right\|^2 + \left\| (\mathbf{W} + \tilde{\Delta})\mathbf{K}_0 - \mathbf{V}_0 \right\|^2 \right). \tag{1}$$

where $\|\cdot\|$ is the Frobenius norm. Using the fact that $\mathbf{W}\mathbf{K}_0 = \mathbf{V}_0$, the second term simplifies to $\left\| \tilde{\Delta}\mathbf{K}_0 \right\|^2$. Applying the normal equation (Lang, 2012), the closed-form solution (when the inverse exists) is:

$$\begin{aligned} \Delta &= \arg\min_{\tilde{\Delta}} \left(\left\| (\mathbf{W} + \tilde{\Delta})\mathbf{K}_1 - \mathbf{V}_1 \right\|^2 + \left\| \tilde{\Delta}\mathbf{K}_0 \right\|^2 \right) \\ &= (\mathbf{V}_1 - \mathbf{W}\mathbf{K}_1)\mathbf{K}_1^\top \left(\mathbf{K}_0\mathbf{K}_0^\top + \mathbf{K}_1\mathbf{K}_1^\top \right)^{-1}. \end{aligned} \tag{2}$$

This solution provides a principled way to update model parameters while preserving existing knowledge, forming the foundation of many recent model editing methods. However, the inversion usually takes a lot of time with a time complexity of $O(d^3)$, particularly for large language models (large d).

4 METHOD

4.1 EFFICIENT REGULARIZATION VIA LATENT STRUCTURE MODELING

We build upon the *locate-then-edit* paradigm (Meng et al., 2022; 2023) and formulate knowledge editing as a regularized optimization problem: modify the model weights $\mathbf{W}$ by an update Δ to satisfy a new knowledge constraint $(\mathbf{K}_1, \mathbf{V}_1)$, while minimizing interference with existing knowledge $\mathbf{K}_0$. The objective is:

$$\mathcal{L} = \|(\mathbf{W} + \Delta)\mathbf{K}_1 - \mathbf{V}_1\|_F^2 + \lambda \|\Delta\mathbf{K}_0\|_F^2, \tag{3}$$

where $\lambda > 0$ is a regularization coefficient that balances the trade-off between satisfying the new knowledge and minimizing interference with existing representations. A larger λ enforces stronger invariance over $\mathbf{K}_0$, reducing side effects at the potential cost of underfitting the edit. The term $\|\Delta\mathbf{K}_0\|_F^2$ measures how the update Δ affects existing representations. However, directly using it in optimization can be computationally expensive. To simplify this, we can naively apply the submultiplicative property:

$$\|\Delta\mathbf{K}_0\|_F^2 \leq \|\mathbf{K}_0\|_2^2 \|\Delta\|_F^2, \tag{4}$$

which decouples Δ from $\mathbf{K}_0$ and leads to a scalar-weighted Frobenius norm. While computationally efficient, this upper bound is *structurally blind*: it penalizes all directions of Δ equally, regardless of their semantic impact on the model's latent space. To design a *semantically aware* and *efficient* regularizer, we instead consider the expected influence of Δ under a structured probabilistic model of the key distribution.

Low-Rank Plus Diagonal (LR+D) Factor Model. Specifically, we assume that the pre-editing hidden representation $\mathbf{k}$ follows a low-rank plus diagonal (LR+D) structure (Fan et al., 2013), motivated by empirical findings that Transformer representations often lie in a low-dimensional subspace (Aghajanyan et al., 2021; Yu & Wu, 2023). We further justify the low-rank nature of these representations from a mathematical perspective; see Appendix A.1 for a formal analysis. The model is given by:

$$\mathbf{k} = \boldsymbol{\mu} + \mathbf{U}\mathbf{z} + \boldsymbol{\varepsilon}, \tag{5}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean (assumed to be $\mathbf{0}$ after centering), $\mathbf{z} \in \mathbb{R}^{r_0}$ is a low-dimensional latent variable ($r_0 \ll d$) with $\mathbb{E}[\mathbf{z}] = \mathbf{0}$ and $\mathrm{Cov}(\mathbf{z}) = \mathbf{I}$, $\mathbf{U} \in \mathbb{R}^{d\times r_0}$ captures dominant semantic directions (e.g., topics or relations), and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{D})$ represents isotropic or anisotropic noise with diagonal covariance $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_d)$, independent of $\mathbf{z}$. This model subsumes several important special cases. When $\mathbf{U} = \mathbf{0}$, it reduces to a diagonal-covariance Gaussian: $\mathbf{k} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{D})$. Further, if $\mathbf{D} = \sigma^2\mathbf{I}$, the covariance becomes isotropic ($\mathbf{C} = \sigma^2\mathbf{I}$), and the expected penalty $\mathbb{E}\left[\|\Delta\mathbf{k}\|_F^2\right]$ becomes $\sigma^2\|\Delta\|_F^2$, recovering the scalar-scaled Frobenius norm in Equation 4.

Under this model, the expected regularization term in Equation 3 can be derived as:

$$\mathbb{E}_{\mathbf{K}_0}\left[\|\Delta\mathbf{K}_0\|_F^2\right] \propto \mathbb{E}_{\mathbf{k}}\left[\|\Delta\mathbf{k}\|_2^2\right] = \mathrm{Trace}\left(\Delta^\top\Delta\left(\mathbf{U}\mathbf{U}^\top + \mathbf{D}\right)\right),$$

See Appendix A.2 for a detailed derivation. This expectation reveals that the impact of Δ is governed not merely by its magnitude, but by its alignment with the underlying structure of the key space. Specifically, edits that align with the semantic subspace spanned by $\mathbf{U}$—i.e., directions of high data variance—have greater influence on existing representations and are thus more disruptive. In contrast, perturbations in the orthogonal complement $\mathbf{U}^\perp$—i.e., the null space of $\mathbf{U}^\top$—affect lower-variance directions and incur less interference. Consequently, the expectation implicitly encodes the geometry of the latent representation space, assigning higher penalty to changes along semantically salient directions. Replacing the empirical Frobenius norm $\|\Delta\mathbf{K}_0\|_F^2$ in Equation 3 with this expected regularizer leads to the modified objective:

$$\Delta = \arg\min_{\hat{\Delta}} \left\|(\mathbf{W} + \hat{\Delta})\mathbf{K}_1 - \mathbf{V}_1\right\|_F^2 + \lambda \cdot \mathrm{Trace}\left(\hat{\Delta}^\top\hat{\Delta}(\mathbf{U}\mathbf{U}^\top + \mathbf{D})\right). \tag{6}$$

Letting $\mathbf{R} = \mathbf{V}_1 - \mathbf{W}\mathbf{K}_1$, the closed-form solution is given by:

$$\Delta = \mathbf{R}\mathbf{K}_1^\top\mathbf{M}^{-1}, \quad \text{where } \mathbf{M} = \mathbf{K}_1\mathbf{K}_1^\top + \lambda(\mathbf{U}\mathbf{U}^\top + \mathbf{D}). \tag{7}$$

Here lies our core optimization: we exploit low-rank structure in both the pre-editing keys $\mathbf{K}_0$ and the editing keys $\mathbf{K}_1$ to achieve **dual benefits**: *semantic-aware regularization* and *computational efficiency*. $\mathbf{U} \in \mathbb{R}^{d\times r_0}$ captures the dominant semantic directions of $\mathbf{K}_0$ with $r_0 \ll d$, and $\mathbf{K}_1 \in \mathbb{R}^{d\times b_1}$ typically has small batch size $b_1 \ll d$ in sequential editing tasks, so both $\mathbf{U}\mathbf{U}^\top$ and $\mathbf{K}_1\mathbf{K}_1^\top$ are low-rank.This structure enables efficient inversion of the matrix $\mathbf{M}$ via the Sherman-Morrison-Woodbury (SMW) identity (Golub &

Van Loan, 2013). Specifically, $\mathbf{M}$ can be viewed as a low-rank perturbation ($\mathbf{K}_1\mathbf{K}_1^\top + \lambda\mathbf{U}\mathbf{U}^\top$) added to a diagonal matrix $\lambda\mathbf{D}$. Applying the SMW identity, we obtain:

$$\mathbf{M}^{-1} = (\lambda\mathbf{D})^{-1} - (\lambda\mathbf{D})^{-1}\mathbf{A}\left(\mathbf{I} + \mathbf{A}^\top(\lambda\mathbf{D})^{-1}\mathbf{A}\right)^{-1}\mathbf{A}^\top(\lambda\mathbf{D})^{-1}, \tag{8}$$

where $\mathbf{A} = \left[\mathbf{K}_1,\ \sqrt{\lambda}\mathbf{U}\right] \in \mathbb{R}^{d\times(b_1+r_0)}$ combines the new knowledge and pre-editing knowledge directions. The application of SMW identity reduces the inversion complexity of $\mathbf{M}$ from $O(d^3)$ to $O(d(b_1+r_0)^2)$ and the memory from $O(d^2)$ to $O(d(b_1+r_0))$, making the method scalable to large models.

Estimation of $\mathbf{U}$ and $\mathbf{D}$. Given the low-rank plus diagonal (LR+D) structure in Equation 5, the population covariance of a pre-editing key vector $\mathbf{k}$ is (see Appendix A.3 for a detailed derivation):

$$\mathrm{Cov}(\mathbf{k}) = \mathbb{E}\left[(\mathbf{k}-\boldsymbol{\mu})(\mathbf{k}-\boldsymbol{\mu})^\top\right] = \mathbf{U}\mathbb{E}[\mathbf{z}\mathbf{z}^\top]\mathbf{U}^\top + \mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top] = \mathbf{U}\mathbf{U}^\top + \mathbf{D}, \tag{9}$$

Since the true covariance is unknown, we estimate this structure from the sampled pre-editing keys $\mathbf{K}_0 \in \mathbb{R}^{d\times b_0}$ by computing the sample covariance $\mathbf{C}_{\text{data}} = \frac{1}{b_0-1}(\mathbf{K}_0-\hat{\mu})(\mathbf{K}_0-\hat{\mu})^\top$, where $\hat{\mu}$ denotes the empirical mean of the pre-editing keys. However, when b_0 is small or the sampled keys are noisy, $\mathbf{C}_{\text{data}}$ may provide a poor estimate of the true latent structure, leading to unstable or semantically misaligned regularization. To improve robustness, we incorporate a structural prior derived from the MLP down-projection weights $\mathbf{W}$, whose right singular vectors span the input directions of maximal variance for the MLP output. Let $\mathbf{W}_{\text{down}} = \mathbf{P}\mathbf{S}\mathbf{Q}^\top$ be its SVD, and let $\mathbf{V}_r = \mathbf{Q}_{:,1:r_0} \in \mathbb{R}^{d\times r_0}$ denote the top-r_0 right singular vectors. We define the prior covariance as:

$$\mathbf{C}_{\text{prior}} = \mathbf{V}_r\boldsymbol{\Lambda}_v\mathbf{V}_r^\top, \tag{10}$$

where $\boldsymbol{\Lambda}_v$ is a diagonal matrix of prior weights (e.g., identity or squared singular values). To ensure numerical compatibility, we normalize $\mathbf{C}_{\text{prior}}$ such that $\|\mathbf{C}_{\text{prior}}\|_F = \|\mathbf{C}_{\text{data}}\|_F$. The fused covariance is:

$$\mathbf{C}_{\text{fused}} = (1-\alpha)\cdot\mathbf{C}_{\text{data}} + \alpha\cdot\mathbf{C}_{\text{prior}}, \quad \alpha\in[0,1], \tag{11}$$

where $\alpha = 0$ recovers the data-driven estimate, and $\alpha = 1$ uses only the prior. Given the fused covariance, we compute its eigendecomposition $\mathbf{C}_{\text{fused}} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^\top$, and set:

$$\mathbf{U} = \mathbf{P}_{:,1:r_0}\boldsymbol{\Lambda}^{1/2}_{1:r_0,1:r_0}, \tag{12}$$

$$\mathbf{D} = \mathrm{diag}\left(\mathbf{C}_{\text{fused}} - \mathbf{U}\mathbf{U}^\top\right). \tag{13}$$

Selecting the top-r_0 eigenvectors is justified by the Eckart–Young–Mirsky theorem (Golub & Van Loan, 2013), which states that the truncated eigendecomposition provides the best rank-r_0 approximation to $\mathbf{C}_{\text{fused}}$ in the Frobenius norm. This ensures that $\mathbf{U}\mathbf{U}^\top$ captures the most significant shared variation in the key space, while the diagonal $\mathbf{D}$ absorbs residual noise and idiosyncratic variations.

4.2 Efficient Sequential Editing via Periodic Spectral Compression

Real-world knowledge editing often occurs sequentially: new facts arrive in batches, requiring updates without reprocessing all prior data. Let $\{(\mathbf{K}_t,\mathbf{V}_t)\}_{t=1}^T$ denote a sequence of edit requests. At each step t, we aim to satisfy $(\mathbf{W}_{t-1}+\Delta_t)\mathbf{K}_t = \mathbf{V}_t$ (with $\mathbf{W}_0$ the initial weights), while preserving both *previously edited* and *pre-editing* knowledge. This requires computing the inverse of the following matrix:

$$\mathbf{M}_t = \sum_{i=1}^{t}\mathbf{K}_i\mathbf{K}_i^\top + \lambda(\mathbf{U}\mathbf{U}^\top + \mathbf{D}),$$

which generalizes the single-step matrix $\mathbf{M}$ in Equation 7. A straightforward approach would reapply the Sherman–Morrison–Woodbury (SMW) identity using all accumulated keys $\mathbf{K}_{1:t} = [\mathbf{K}_1,\ldots,\mathbf{K}_t]$

($\mathbf{K}_{1:t}\mathbf{K}_{1:t}^\top = \sum_{i=1}^{t} \mathbf{K}_i\mathbf{K}_i^\top$). This results in a per-step computational cost of $O(dr_t^2)$, where denotes the number of column vectors in $\mathbf{K}_{1:t}$. As t increases, r_t grows linearly with the number of edits, making the update progressively more expensive. When r_t becomes large—particularly as it approaches the model dimension d—the inversion of SMW inner system scales as $O(r_t^3)$, causing the overall cost to approach $O(d^3)$ and effectively negating the efficiency gains from the low-rank assumption.

To maintain efficiency, we apply periodic low-rank compression: every τ incoming key vectors, we perform SVD on the accumulated keys and retain only the top singular components that preserve most of the directional energy. Let $\mathbf{K}_{\text{comp}}$ denote the compressed key matrix from previous cycles (or empty initially), and let $\mathbf{K}_{\text{all}} = [\mathbf{K}_{\text{comp}}, \mathbf{K}_{\text{buff}}]$ be the full set of keys to compress, where $\mathbf{K}_{\text{buff}}$ is the current buffer of unprocessed keys. We compute the SVD $\mathbf{K}_{\text{all}} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ and retain the largest r components such that

$$r = \min\left\{ k : \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r'} \sigma_i^2} \geq \gamma,\ k \leq r_{\max} \right\},$$

where σ_i are the singular values, $r' = \text{rank}(\mathbf{K}_{\text{all}})$, $\gamma \in (0, 1]$ is the energy retention threshold, and $r_{\max}$ caps the maximum rank to prevent unbounded growth. The compressed key matrix is then updated as $\mathbf{K}_{\text{comp}} \leftarrow \mathbf{U}_{:,1:r}\mathbf{S}_{1:r,1:r}$, and the buffer is reset. The accumulated keys $\mathbf{K}_{\text{all}}$ is used in the SMW updates at each step, ensuring per-step computational cost remains bounded at $O(dr^2)$. The full procedure is summarized in Algorithm 1.

5 Experiments

5.1 Experimental Setup

We outline the experimental setup, including the models, editing methods, datasets, and evaluation metrics used in our study. Further details—such as hyperparameters and baseline implementations for the sequential editing task—are provided in Appendix C.

Language Models & Editing Methods. We conduct experiments on three decoder-only language models: GPT2-XL (1.5B parameters), GPT-J (6B parameters), and Llama-3 (8B parameters). The key vector dimensionality is 6400, 16384, and 14348, respectively. These models vary in architecture and training data, allowing us to evaluate the generalization of editing methods across diverse LLMs. We compare against a range of state-of-the-art locate-then-edit methods: MEMIT (Meng et al., 2023), PMET (Li et al., 2024a), EMMET (Gupta et al., 2024b), AlphaEdit (Fang et al., 2025), RECT (Gu et al., 2024b), PRUNE (Ma et al., 2025), and AdaEdit (Li & Chu, 2025). We do not include ROME (Meng et al., 2022) as it is a special case of EMMET with batch size one. Among these, RECT, PRUNE, and AdaEdit are post-regularization methods that explicitly constrain updates to preserve behavior on unrelated inputs.

Editing Datasets & Evaluation Metrics. We evaluate on two standard factual editing benchmarks: ZsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022). Each edit is defined by an input-output pair (x, y). In ZsRE, x is a question (e.g., *What university did Watts Humphrey attend?*) and y is the target answer (e.g., *Illinois Institute of Technology*). In CounterFact, x is a cloze prompt (e.g., *The mother tongue of Danielle Darrieux is*) and y is the new fact (e.g., *English*), with the original fact y_o (e.g., *French*) provided. We assess performance using three primary metrics: **Efficacy (Eff*)** measures whether the model generates y as the top prediction given x, i.e., $y = \arg\max_{y'} P(y' \mid x)$. **Generality (Gen*)** evaluates robustness to input variation by measuring success on paraphrased inputs x_g, i.e., $y = \arg\max_{y'} P(y' \mid x_g)$. **Specificity (Spe*)** measures locality by the percentage of unrelated fact pairs (x_s, y_s) that remain correctly predicted after editing. For CounterFact, since the original fact y_o is provided, we additionally report probability-increase-based metrics, where an edit is considered successful if $P(y \mid x) > P(y_o \mid x)$, following prior work (Meng et al., 2023; Fang et al., 2025). The corresponding metrics are denoted as **Eff.**, **Gen.**, and **Spe.**.

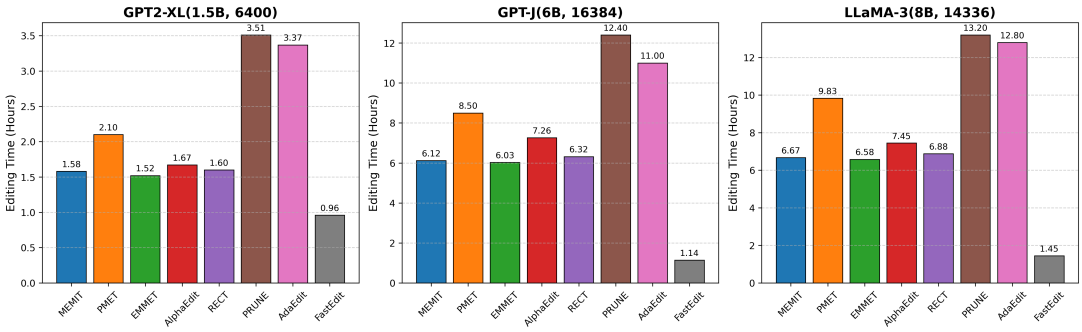

Figure 1: Editing time comparison for 2,000 sequential edits on the COUNTERFACT dataset across different models. Each subplot is labeled with the model name, parameter count, and key space dimension d.

5.2 EDITING EFFICIENCY & EFFICACY

Editing time. We evaluate computational efficiency by measuring the time to perform 2000 edits on the CounterFact dataset. The effectiveness of our low-rank update strategy is empirically supported by the long-tailed singular value distribution of both the pre-edit keys $\mathbf{K}_0$ and the sequence of edited keys $\mathbf{K}_{[1:t]}$, as shown in Figure 2, indicating that the knowledge dynamics during editing are intrinsically low-rank. This justifies our use of a rank-r update with $r \ll d$, which avoids costly matrix inversions. As shown in Figure 1, FastEdit achieves remarkable speedups across all models—GPT2-XL (1.5B), GPT-J (6B), and LLaMA-3 (8B)—reducing editing time to just **0.96 hours** on GPT2-XL, **1.14 hours** on GPT-J, and **1.45 hours** on LLaMA-3, significantly outperforming all baselines. It runs approximately $5\times$ faster than the next fastest method on larger models, and up to $10\times$ faster than PRUNE and AdaEdit. This dramatic improvement stems from our key optimization: replacing the $O(d^3)$ inverse in FFN editing with an $O(dr^2)$ update via the Sherman-Morrison-Woodbury identity. The results confirm that FastEdit scales efficiently with model size, making it highly practical for real-world deployment.

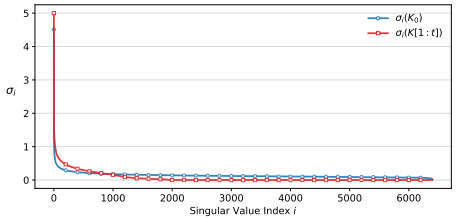

Figure 2: Singular values of the pre-edit key matrix $\mathbf{K}_0$ and the concatenated edited keys $\mathbf{K}_{[1:t]}$ over 2000 sequential edits on COUNTERFACT using GPT2-XL, illustrating the inherent low-rank structure.

Pre-computation time. Locate-then-edit methods typically require extensive sampling of pre-editing keys (e.g., $\sim 4 \times 10^7$ in prior work) to estimate the semantic subspace, taking over 24 hours for LLaMA-3. In contrast, FastEdit requires only $\sim 4 \times 10^4$ samples, reducing pre-computation time to a few minutes. This efficiency is enabled by our robust covariance estimation procedure (detailed in Section 4), which fuses data-driven statistics with a structural prior from the MLP down-projection weights. This allows stable estimation of $\mathbf{U}$ and $\mathbf{D}$ even with limited samples, eliminating the need for massive key collection and enabling rapid deployment on new large language models.

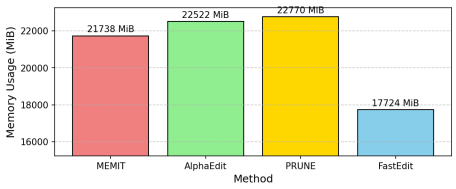

Figure 3: GPU memory usage comparison on LLaMA-3 (bfloat16, bs=1).

Table 1: Editing performance on GPT-J and LLaMA-3 across COUNTERFACT and ZSRE datasets. ↑: higher is better; ↕: closer to pre-edit values is better.

Method		CounterFact						ZsRE		
		Eff.↑	Gen.↑	Spe.↑	Eff*.↑	Gen*.↑	Spe*.↕	Eff*.↑	Gen*.↑	Spe*.↕
Pre-edited		15.4	18.0	83.3	0.40	0.60	13.7	27.7	27.1	27.4
MEMIT	GPT-J	99.2	89.9	**77.5**	96.4	51.3	9.90	98.8	93.3	**27.5**
PMET	GPT-J	99.7	95.0	73.3	98.0	69.9	10.6	99.7	**97.8**	28.4
EMMET	GPT-J	99.6	94.4	75.3	98.1	60.6	10.4	**99.8**	97.2	28.6
AlphaEdit	GPT-J	**99.8**	93.8	76.6	**99.0**	61.2	9.90	**99.8**	**97.8**	28.7
RECT	GPT-J	99.2	89.0	**77.5**	95.2	49.1	9.30	98.7	92.9	27.6
PRUNE	GPT-J	99.4	96.1	73.2	98.4	**74.4**	**12.5**	99.3	95.9	30.9
AdaEdit	GPT-J	99.7	96.0	72.7	98.5	71.5	10.6	99.5	94.6	26.4
FastEdit	GPT-J	**99.8**	**96.3**	71.4	98.6	71.6	12.2	99.6	96.5	28.2
Pre-edited		7.80	10.4	89.3	0.30	0.50	21.3	38.2	37.6	38.6
MEMIT	LLaMA3	99.0	92.9	71.5	97.0	71.3	18.4	**99.2**	95.4	**43.9**
PMET	LLaMA3	99.2	**95.7**	66.2	97.0	**75.6**	16.5	99.1	**96.5**	45.4
EMMET	LLaMA3	51.1	50.8	48.9	0.00	0.00	0.00	97.3	93.9	46.0
AlphaEdit	LLaMA3	56.1	53.9	48.5	2.90	1.80	0.60	98.0	94.2	45.7
RECT	LLaMA3	99.0	93.0	71.0	96.8	71.6	18.5	99.1	96.0	44.0
PRUNE	LLaMA3	85.2	77.0	64.6	39.9	33.3	7.40	94.3	91.3	46.8
AdaEdit	LLaMA3	77.6	70.3	56.1	36.8	27.7	0.07	93.2	90.6	46.6
FastEdit	LLaMA3	**99.6**	93.6	**74.4**	**98.4**	70.5	**19.8**	99.0	95.1	45.3

Memory Cost. We measure the peak GPU memory consumption during editing on LLaMA-3 using bfloat16 precision with a batch size of 1. As shown in Figure 3, FastEdit consumes only **17,724 MiB**, notably lower than MEMIT (21,738 MiB), AlphaEdit (22,522 MiB), and PRUNE (22,770 MiB). This efficiency stems from our $O(dr)$ parameterization of the update, which avoids the $O(d^2)$ memory overhead of dense matrix storage. All measurements were conducted on an NVIDIA A6000 with 48GB GPU memory. While the memory saving is not critical on such high-end devices, FastEdit's reduced footprint enhances feasibility in resource-constrained environments and enables potential extensions to larger models or higher batch sizes where memory becomes a bottleneck.

Editing Performance. We evaluate editing performance on GPT-J and LLaMA-3 using the CounterFact and ZsRE datasets, with results for GPT2-XL provided in Appendix E. As shown in Table 1, FastEdit achieves strong overall performance. On CounterFact, it obtains the highest efficacy (99.6%/98.4%) and specificity (74.4%/19.8%), outperforming all baseline methods while maintaining competitive generality. Notably, methods such as EMMET and AlphaEdit exhibit severe performance degradation, achieving the lowest scores across all metrics, which indicates poor stability under sustained editing pressure. On ZsRE, FastEdit performs competitively, with minor gaps behind MEMIT and RECT. These results confirms that high editing efficiency can be achieved without compromising editing performance.

5.3 SAFETY-AWARE EDITING EVALUATION

To evaluate the safety of model edits, we leverage the low-rank semantic subspace $\mathbf{U} \in \mathbb{R}^{d \times r_0}$ ($r_0 \ll d$) learned from pre-edit FFN key distributions. We introduce two complementary geometric metrics that assess edit safety from different but consistent perspectives. First, to assess the geometric alignment of each edit with underutilized regions of the key space, we compute the *orthogonal energy* of the new key

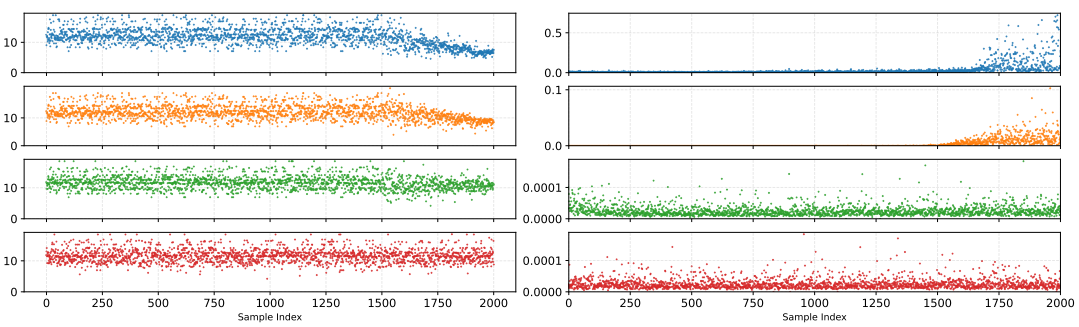

Figure 4: Temporal evolution of edit safety metrics e_t (left, higher is better) and s_t (right, lower is better) over 2000 sequential edits on COUNTERFACT using LLaMA-3. Each row corresponds to a method, from top to bottom: ALPHAEDIT, EMMET, MEMIT, FASTEDIT.

$e_t = \|(\mathbf{I} - \mathbf{P}_\mathbf{U})\mathbf{k}_t\|^2$, where $\mathbf{P}_\mathbf{U} = \mathbf{U}(\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top$ is the orthogonal projection onto $\mathrm{span}(\mathbf{U})$, and $\mathbf{k}_t$ is the new key introduced at the t-th sequential edit. Higher e_t indicates a safer edit, as it reflects greater alignment with the "quiet" subspace orthogonal to existing semantic directions—regions where interference is less likely. Second, we define the *subspace interference* metric $s_t = \|\Delta_t\mathbf{U}\|_F$, which measures the component of the weight update Δ_t at step t that lies within the dominant directions of existing knowledge. A larger s_t indicates stronger interference with the model's semantic structure, suggesting a higher risk of undesirable side effects. More details are provided in Appendix D.

We compute s_t and e_t over 2000 sequential edits on the COUNTERFACT dataset using four editing methods: ALPHAEDIT, EMMET, MEMIT, and FASTEDIT, all applied to LLaMA-3. As shown in Figure 4, during the initial phase (edits 1–1500), all methods maintain high e_t and low s_t, indicating that edits are successfully confined to orthogonal, low-interference directions. However, after edit #1500, a clear divergence emerges:

- ALPHAEDIT and EMMET exhibit a **sharp decline in** e_t and a corresponding **rise in** s_t, indicating that their updates begin to leak into the semantic subspace, increasing the risk of interference.
- In contrast, MEMIT and FASTEDIT continue to maintain high e_t and low s_t, suggesting that they preserve access to orthogonal directions and avoid corrupting existing knowledge.

This stark contrast demonstrates the ability of our metrics to detect potential safety issues in sequential editing. The concurrent degradation in both e_t (key space) and s_t (weight space) provides strong evidence that $\mathbf{U}$ captures meaningful structure in the model's pre-editing knowledge manifold. These metrics form a robust diagnostic framework for distinguishing between transiently effective and truly sustainable editing methods. Crucially, e_t can be computed *before* applying the edit and does not require access to the update matrix Δ_t, making it a practical prior for assessing edit safety across diverse editing methods.

6 CONCLUSION

We presented FastEdit, an efficient model editing framework that exploits the low-rank structure of FFN key spaces to replace costly $O(d^3)$ updates with fast $O(dr^2)$ revisions, achieving up to 10× speedup over existing methods while maintaining competitive editing accuracy. To assess editing safety, we introduced a set of interpretable metrics that capture semantic interference in the model's pre-editing knowledge space. FastEdit thus enables efficient, safe, and scalable knowledge editing—bringing continuous model updating one step closer to reality. Future work may generalize this low-rank approach to other editing frameworks (e.g., AlphaEdit (Fang et al., 2025)), study its compositional behavior under long-term massive-scale updates.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, 2021.

Baolong Bi, Shenghua Liu, Lingrui Mei, Yiwei Wang, Junfeng Fang, Pengliang Ji, and Xueqi Cheng. Decoding by contrasting knowledge: Enhancing large language model confidence on edited facts. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17198–17208, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. *NeurIPS*, 2020.

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.

Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. Namet: Robust massive model editing via noise-aware memory optimization. *arXiv preprint arXiv:2505.11876*, 2025.

Nicola De Cao, Tejas Kipf, Patrick Lewis, et al. Editing large language models: Problems, methods, and opportunities. In *EMNLP*, 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.

Jianqing Fan, Yuan Liao, and Marine Mincheva. High-dimensional covariance matrix estimation in approximate factor models. *The Annals of Statistics*, 39(6):3320–3356, 2013.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

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*. Association for Computational Linguistics, 2021.

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

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. Pokemqa: Programmable knowledge editing for multi-hop question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8069–8083, 2024a.

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, 2024b.

Akshat Gupta, Anurag Rao, and Gopala Anumanchipalli. Model editing at scale leads to gradual and catastrophic forgetting. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 15202–15232, 2024a.

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

Akshat Gupta, Maochuan Lu, Thomas Hartvigsen, and Gopala Anumanchipalli. Efficient knowledge editing via minimal precomputation. *arXiv preprint arXiv:2506.04226*, 2025.

Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8214–8232, 2023.

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.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Knowledge in superposition: Unveiling the failures of lifelong knowledge editing for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24086–24094, 2025.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*, 2023.

Yuxin Jiang, Yufei Wang, Chuhan Wu, Wanjun Zhong, Xingshan Zeng, Jiahui Gao, Liangyou Li, Xin Jiang, Lifeng Shang, Ruiming Tang, et al. Learning to edit: Aligning llms with knowledge editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4689–4705, 2024.

Serge Lang. *Introduction to linear algebra*. Springer Science & Business Media, 2012.

Jan Leike, Jonathan Uesato, Natasha Monga, et al. Towards long-term safety for reinforcement learning agents, 2023. Anthropic Blog.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *21st Conference on Computational Natural Language Learning, CoNLL 2017*, pp. 333–342. Association for Computational Linguistics (ACL), 2017.

Qi Li and Xiaowen Chu. Can we continually edit language models? on the knowledge attenuation in sequential model editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5438–5455, 2024.

Qi Li and Xiaowen Chu. Adaedit: Advancing continuous knowledge editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4127–4149, 2025.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18564–18572, 2024a.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. Unveiling the pitfalls of knowledge editing for large language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *The Thirteenth International Conference on Learning Representations*, 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, 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.

Meta. Llama 3. Large language model release, 2024. URL `https://llama.meta.com/llama3/`. Accessed: 2025-09-20.

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.

Haewon Park, Gyubin Choi, Minjun Kim, and Yohan Jo. Context robust knowledge editing for language models. *arXiv preprint arXiv:2505.23026*, 2025.

Aditya Ramesh, Prafulla Dhariwal, Nick Lourie, et al. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2204.07705*, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Akshay Vellal, Xiang Li, Christopher Mitchell, et al. Temporal knowledge in language models: A survey. In *ICLR*, 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, 2024.

Hao Yu and Jianxin Wu. Compressing transformers: features are low-rank, but weights are not! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11007–11015, 2023.

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.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Shu Wu, Pengjie Ren, and Zhumin Chen. Uncovering overfitting in large language model editing. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. Instructedit: instruction-based knowledge editing for large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6633–6641, 2024.

Shuning Zhao, Anna Roberts, Frank Li, et al. Do llms really separate knowledge from inference? In *ACL*, 2023.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. Can we edit factual knowledge by in-context learning? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4862–4876, 2023.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 15686–15702, 2023.

A MODELING THE PRE-EDITING KEYS VIA LR+D FACTOR MODEL

A.1 RANK ANALYSIS OF THE SECOND LINEAR LAYER INPUT

In this section, we provide a mathematical justification for the low-rank structure observed in the input to the second linear layer of the Feed-Forward Network (FFN) in Transformers. Specifically, we show that due to architectural constraints, the effective dimensionality of these activations is inherently limited.

Consider a single token's hidden representation $\mathbf{x} \in \mathbb{R}^d$ as input to the FFN block. The FFN applies the following transformation:

$$\mathrm{FFN}(\mathbf{x}) = \mathbf{W}_2 \cdot \mathrm{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2, \tag{14}$$

where $\mathbf{W}_1 \in \mathbb{R}^{4d\times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times 4d}$ are weight matrices, and $\mathbf{b}_1, \mathbf{b}_2$ are biases (we omit biases for simplicity in the analysis below).

Let $\mathbf{a} = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x}) \in \mathbb{R}^{4d}$ denote the activation vector that serves as input to the second linear layer ($\mathbf{W}_2$). We analyze the rank properties of the set of all possible such activations.

Linear Transformation Stage. The first stage computes $\mathbf{z} = \mathbf{W}_1\mathbf{x}$. Since $\mathbf{W}_1 \in \mathbb{R}^{4d\times d}$, its column space (image) satisfies:

$$\dim\left(\mathrm{Im}(\mathbf{W}_1)\right) = \mathrm{rank}(\mathbf{W}_1) \le d. \tag{15}$$

Thus, $\mathbf{z} = \mathbf{W}_1\mathbf{x}$ lies in a subspace of $\mathbb{R}^{4d}$ with dimension at most d, regardless of the specific $\mathbf{x} \in \mathbb{R}^d$.

Nonlinear Activation Stage. The ReLU function $\mathrm{ReLU}(\cdot) = \max(\cdot, 0)$ is applied element-wise to $\mathbf{z}$, yielding $\mathbf{a} = \mathrm{ReLU}(\mathbf{z})$. While ReLU is nonlinear and breaks the linear subspace structure, it does not increase the intrinsic dimensionality of the mapping. Specifically:

The image of the map $f : \mathbb{R}^d \to \mathbb{R}^{4d}$, defined by $f(\mathbf{x}) = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x})$, has topological dimension at most d.

Proof. Since $\mathbf{W}_1$ is a linear map from $\mathbb{R}^d$ to $\mathbb{R}^{4d}$, it is continuous and its image is contained in a d-dimensional subspace. The ReLU function is continuous and piecewise linear. The composition $f = \mathrm{ReLU} \circ \mathbf{W}_1$ is therefore a continuous map from a d-dimensional domain to $\mathbb{R}^{4d}$. By standard results in topology, the image of a d-dimensional manifold under a continuous map cannot exceed d in topological dimension. Hence, the set $\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^d\}$ forms a d-dimensional (or lower) subset of $\mathbb{R}^{4d}$, i.e., a d-dimensional manifold. □

Implication for Batched Inputs. Now consider a batch of n inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, and let $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{4d\times n}$ be the matrix of activations, where $\mathbf{a}_i = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x}_i)$. Then:

$$\mathrm{rank}(\mathbf{A}) \le d, \tag{16}$$

since each column $\mathbf{a}_i$ is determined by $\mathbf{x}_i \in \mathbb{R}^d$, and the mapping is deterministic. Even if $n > d$, the rank cannot exceed d due to the bottleneck imposed by the input dimension and the fixed $\mathbf{W}_1$.

Conclusion. This analysis shows that the input to the second linear layer, $\mathbf{a}$, is constrained to a low-dimensional manifold in $\mathbb{R}^{4d}$, with intrinsic dimension at most $d \ll 4d$. This structural property justifies our use of a low-rank plus diagonal (LR+D) model for the covariance of these activations in Section 4, as the full $4d$-dimensional space is not fully utilized.

A.2 Expected Regularization Term: Detailed Derivation

In this section, we derive the expected value of the empirical regularization term $\|\Delta\mathbf{K}_0\|_F^2$ under the assumption that the pre-edit keys $\mathbf{K}_0 \in \mathbb{R}^{d\times b_0}$ are independently sampled from a zero-mean distribution with covariance structure $\mathbf{\Sigma} = \mathbf{U}\mathbf{U}^\top + \mathbf{D}$, as defined in the LR+D model (Equation equation 5).

Let $\mathbf{K}_0 = [\mathbf{k}_1, \ldots, \mathbf{k}_{b_0}]$, where each $\mathbf{k}_i \in \mathbb{R}^d$ is an i.i.d. sample satisfying:

$$\mathbb{E}[\mathbf{k}_i] = 0, \quad \mathbb{E}[\mathbf{k}_i\mathbf{k}_i^\top] = \mathbf{\Sigma} = \mathbf{U}\mathbf{U}^\top + \mathbf{D}.$$

We aim to compute:

$$\mathbb{E}_{\mathbf{K}_0}\left[\|\Delta\mathbf{K}_0\|_F^2\right],$$

where $\Delta \in \mathbb{R}^{d\times d}$ is a fixed update matrix.

The squared Frobenius norm can be expanded column-wise:

$$\|\Delta\mathbf{K}_0\|_F^2 = \sum_{i=1}^{b_0} \|\Delta\mathbf{k}_i\|_2^2.$$

Taking expectation over $\mathbf{K}_0$ (which is equivalent to taking expectation over each $\mathbf{k}_i$ due to independence):

$$\mathbb{E}_{\mathbf{K}_0}\left[\|\Delta\mathbf{K}_0\|_F^2\right] = \mathbb{E}_{\mathbf{K}_0}\left[\sum_{i=1}^{b_0} \|\Delta\mathbf{k}_i\|_2^2\right] = \sum_{i=1}^{b_0} \mathbb{E}_{\mathbf{k}_i}\left[\|\Delta\mathbf{k}_i\|_2^2\right].$$

Since all $\mathbf{k}_i$ are identically distributed, we denote $\mathbf{k} \sim p(\mathbf{k})$ as a generic key vector, and write:

$$\mathbb{E}_{\mathbf{K}_0}\left[\|\Delta\mathbf{K}_0\|_F^2\right] = b_0 \cdot \mathbb{E}_{\mathbf{k}}\left[\|\Delta\mathbf{k}\|_2^2\right].$$

Now, observe that:

$$\|\Delta\mathbf{k}\|_2^2 = (\Delta\mathbf{k})^\top(\Delta\mathbf{k}) = \mathbf{k}^\top\Delta^\top\Delta\mathbf{k}.$$

Thus,

$$\mathbb{E}_{\mathbf{k}}\left[\|\Delta\mathbf{k}\|_2^2\right] = \mathbb{E}_{\mathbf{k}}\left[\mathbf{k}^\top\Delta^\top\Delta\mathbf{k}\right].$$

We now apply the following standard result for the expectation of a quadratic form:

[Expectation of Quadratic Form] For a random vector $\mathbf{x} \in \mathbb{R}^d$ with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$, and a fixed symmetric matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we have:

$$\mathbb{E}[\mathbf{x}^\top\mathbf{A}\mathbf{x}] = \mathrm{Tr}(\mathbf{A}\mathbf{\Sigma}) + \boldsymbol{\mu}^\top\mathbf{A}\boldsymbol{\mu}.$$

In our case: - $\mathbf{x} = \mathbf{k}$, - $\mathbf{A} = \Delta^\top\Delta$ (symmetric and positive semi-definite), - $\boldsymbol{\mu} = \mathbb{E}[\mathbf{k}] = 0$, - $\mathbf{\Sigma} = \mathbf{U}\mathbf{U}^\top + \mathbf{D}$.

Therefore,

$$\mathbb{E}_{\mathbf{k}}\left[\mathbf{k}^\top\Delta^\top\Delta\mathbf{k}\right] = \mathrm{Tr}\left(\Delta^\top\Delta\cdot(\mathbf{U}\mathbf{U}^\top + \mathbf{D})\right) + 0^\top(\cdots)0 = \mathrm{Tr}\left(\Delta^\top\Delta\,(\mathbf{U}\mathbf{U}^\top + \mathbf{D})\right).$$

Substituting back, we obtain:

$$\mathbb{E}_{\mathbf{K}_0}\left[\|\Delta\mathbf{K}_0\|_F^2\right] = b_0 \cdot \mathrm{Tr}\left(\Delta^\top\Delta\,(\mathbf{U}\mathbf{U}^\top + \mathbf{D})\right).$$

This shows that the expected regularization term is proportional to the trace expression, with proportionality constant b_0 (the number of pre-edit keys). In practice, this constant is absorbed into the regularization coefficient λ when forming the final objective, yielding the effective regularizer:

$$\mathrm{Tr}\left(\Delta^\top \Delta \left(\mathbf{U}\mathbf{U}^\top + \mathbf{D}\right)\right).$$

This derivation justifies replacing the empirical term $\|\Delta \mathbf{K}_0\|_F^2$ with the expected regularizer in the main optimization objective.

A.3 Derivation of the Covariance Structure

Under the factor model $\mathbf{k} = \boldsymbol{\mu} + \mathbf{U}\mathbf{z} + \boldsymbol{\varepsilon}$ equation 5, with $\mathbb{E}[\mathbf{z}] = \mathbf{0}$, $\mathrm{Cov}(\mathbf{z}) = \mathbf{I}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{D})$, and $\mathbf{z} \perp \boldsymbol{\varepsilon}$, the centered key vector is $\mathbf{k} - \boldsymbol{\mu} = \mathbf{U}\mathbf{z} + \boldsymbol{\varepsilon}$. The covariance is:

$$\mathrm{Cov}(\mathbf{k}) = \mathbb{E}\left[(\mathbf{U}\mathbf{z} + \boldsymbol{\varepsilon})(\mathbf{U}\mathbf{z} + \boldsymbol{\varepsilon})^\top\right] \tag{17}$$

$$= \mathbb{E}\left[\mathbf{U}\mathbf{z}\mathbf{z}^\top\mathbf{U}^\top\right] + \mathbb{E}\left[\mathbf{U}\mathbf{z}\boldsymbol{\varepsilon}^\top\right] + \mathbb{E}\left[\boldsymbol{\varepsilon}\mathbf{z}^\top\mathbf{U}^\top\right] + \mathbb{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top\right]. \tag{18}$$

Using independence and zero means:

- $\mathbb{E}[\mathbf{U}\mathbf{z}\mathbf{z}^\top\mathbf{U}^\top] = \mathbf{U}\,\mathbb{E}[\mathbf{z}\mathbf{z}^\top]\,\mathbf{U}^\top = \mathbf{U}\mathbf{U}^\top$,
- $\mathbb{E}[\mathbf{U}\mathbf{z}\boldsymbol{\varepsilon}^\top] = \mathbf{U}\,\mathbb{E}[\mathbf{z}]\mathbb{E}[\boldsymbol{\varepsilon}^\top] = \mathbf{0}$,
- $\mathbb{E}[\boldsymbol{\varepsilon}\mathbf{z}^\top\mathbf{U}^\top] = \mathbb{E}[\boldsymbol{\varepsilon}]\mathbb{E}[\mathbf{z}^\top]\mathbf{U}^\top = \mathbf{0}$,
- $\mathbb{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top] = \mathbf{D}$.

Summing the terms yields:

$$\mathrm{Cov}(\mathbf{k}) = \mathbf{U}\mathbf{U}^\top + \mathbf{D}, \tag{19}$$

as desired.

B Algorithmic Details of Periodic Spectral Compression

We provide the full algorithmic description of the periodic spectral compression method used in our sequential knowledge editing framework. This procedure maintains an efficient low-rank approximation of accumulated edit keys by periodically compressing them via truncated singular value decomposition (SVD). Specifically, incoming key vectors are accumulated until their total dimension reaches a threshold τ, at which point a global SVD is performed and only the top singular components—preserving at least a fraction γ of the total energy—are retained, up to a maximum rank $r_{\max}$. The compressed key matrix is then used in subsequent Sherman–Morrison–Woodbury (SMW) updates to ensure that each editing step remains computationally efficient, with cost bounded by $O(dr^2)$ where $r = \mathrm{rank}(\mathbf{K}_{\mathrm{comp}})$. The complete procedure is summarized in Algorithm 1.

C Experimental Setup

In this section, we provide an overview of the key model editing methods used as baselines in our evaluation. We then describe how these methods are adapted to *sequential editing tasks* by incorporating mechanisms to protect previously edited knowledge (Fang et al., 2025), which significantly improves their performance over long edit sequences.

Algorithm 1 Periodic Spectral Compression with SMW Update

Require: $\mathbf{D} \in \mathbb{R}^d$, $\mathbf{U} \in \mathbb{R}^{d \times r_u}$, $\mathbf{K}_{\text{new}} \in \mathbb{R}^{d \times b}$, buffer threshold τ, energy threshold γ, max rank $r_{\max}$
Ensure: $\mathbf{K}_{\text{new}}^\top (\mathbf{A} + \mathbf{K}_{\text{new}} \mathbf{K}_{\text{new}}^\top)^{-1}$, where $\mathbf{A} = \mathbf{D} + \mathbf{U}\mathbf{U}^\top + \mathbf{K}_{\text{comp}} \mathbf{K}_{\text{comp}}^\top$

Initialize $\mathbf{K}_{\text{comp}} \leftarrow \mathbf{0}^{d \times 0}$, $\text{count} \leftarrow 0$
$\mathbf{K}_{\text{comp}} \leftarrow [\mathbf{K}_{\text{comp}}, \mathbf{K}_{\text{new}}]$ ▷ Append new keys
$\text{count} \leftarrow \text{count} + b$
if $\text{count} \geq \tau$ **then**
    Compute SVD: $\mathbf{K}_{\text{comp}} = \mathbf{U}_s \mathbf{S}_s \mathbf{V}_s^\top$
    $r \leftarrow \min \left\{ k : \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{\text{rank}} \sigma_i^2} \geq \gamma,\ k \leq r_{\max} \right\}$
    $\mathbf{K}_{\text{comp}} \leftarrow \mathbf{U}_s[:, 1:r] \mathbf{S}_s[1:r, 1:r]$ ▷ Compress and update
    $\text{count} \leftarrow 0$
end if
Form total basis: $\mathbf{U}_{\text{total}} \leftarrow [\mathbf{U}, \mathbf{K}_{\text{comp}}, \mathbf{K}_{\text{new}}]$ ▷ Include protected, compressed, and new keys
Compute $\mathbf{V} \leftarrow \mathbf{D}^{-1/2} \mathbf{U}_{\text{total}}$ ▷ Element-wise: $\mathbf{U}_{\text{total}} / \sqrt{\mathbf{D}}$
$\mathbf{M} \leftarrow \mathbf{I} + \mathbf{V}^\top \mathbf{V}$
$\mathbf{M}^{-1} \leftarrow \text{inv}(\mathbf{M})$
return $\mathbf{K}_{\text{new}}^\top \left(\mathbf{D}^{-1} - \mathbf{D}^{-1} \mathbf{U}_{\text{total}} \mathbf{M}^{-1} \mathbf{U}_{\text{total}}^\top \mathbf{D}^{-1} \right)$

C.1 Baseline Model Editing Methods

We summarize the core ideas of the following seven representative methods:

- **MEMIT** (Meng et al., 2023): Extends single-fact editing (e.g., ROME (Meng et al., 2022)) to batched updates via a least-squares optimization, enabling efficient integration of multiple facts through direct weight modification.
- **PMET** (Li et al., 2024a): Improves editing precision by analyzing information flow in Transformer layers. It observes that Multi-Head Self-Attention (MHSA) encodes general reasoning patterns and should remain unaltered. PMET optimizes hidden states of both MHSA and FFN but only uses FFN states to update weights for more targeted edits.
- **EMMET** (Gupta et al., 2024b): Unifies ROME and MEMIT under a common preservation-memorization objective. While ROME uses equality constraints for single edits, EMMET supports batched editing with the same constraint type, achieving comparable performance with theoretical consistency.
- **AlphaEdit** (Fang et al., 2025): Addresses knowledge disruption in sequential editing by projecting perturbations into the null space of preserved knowledge. This ensures that outputs on unedited facts remain unchanged, significantly improving edit retention with minimal overhead.
- **RECT** (Gu et al., 2024b): Highlights that excessive weight changes during editing degrade general capabilities (e.g., reasoning, inference). It regularizes updates based on the relative change in weights to preserve model functionality while maintaining edit success.
- **PRUNE** (Ma et al., 2025): Identifies the condition number of the edit matrix as a key factor affecting stability in sequential editing. By constraining this value, PRUNE limits parameter perturbation and preserves general knowledge over many edits.
- **AdaEdit** (Li & Chu, 2025): Tackles performance decay in continuous editing by promoting disentangled and sparse representations of edited knowledge, enabling robust performance in large-scale editing scenarios.

C.2 Adaptation to Sequential Editing

To enable fair comparison in sequential editing scenarios, we adapt all baseline methods to preserve previously edited knowledge. Let:

- $\mathbf{K}_0$: initial model keys (pre-editing),
- $\mathbf{K}_t$: current batch of edit keys,
- $\mathbf{K}_{1:t-1} = [\mathbf{K}_1, \ldots, \mathbf{K}_{t-1}]$: keys from all previous edits.

Below we describe the adapted update rules.

MEMIT and MEMIT-based Methods (PMET, RECT, PRUNE, AdaEdit): The original MEMIT update is:

$$\Delta_{\text{MEMIT}} = \mathbf{R}\mathbf{K}_t^T \left(\mathbf{K}_0\mathbf{K}_0^T + \mathbf{K}_t\mathbf{K}_t^T\right)^{-1}. \tag{20}$$

To protect previously edited knowledge, we extend the regularization term:

$$\Delta_{\text{MEMIT}} = \mathbf{R}\mathbf{K}_t^T \left(\mathbf{K}_0\mathbf{K}_0^T + \mathbf{K}_{1:t-1}\mathbf{K}_{1:t-1}^T + \mathbf{K}_t\mathbf{K}_t^T\right)^{-1}. \tag{21}$$

This adaptation is also applied to PMET, RECT, PRUNE, and AdaEdit, as they are built upon the MEMIT framework.

EMMET: The original update rule is:

$$\Delta_{\text{EMMET}} = \mathbf{R}_t \left(\mathbf{K}_t^T(\mathbf{K}_0\mathbf{K}_0^T)^{-1}\mathbf{K}_t\right)^{-1} \mathbf{K}_t^T(\mathbf{K}_0\mathbf{K}_0^T)^{-1}. \tag{22}$$

We adapt it by updating the inverse covariance estimate to include prior edits:

$$\Delta_{\text{EMMET}} = \mathbf{R}_t \left(\mathbf{K}_t^T(\mathbf{K}_0\mathbf{K}_0^T + \mathbf{K}_{1:t-1}\mathbf{K}_{1:t-1}^T)^{-1}\mathbf{K}_t\right)^{-1} \mathbf{K}_t^T(\mathbf{K}_0\mathbf{K}_0^T + \mathbf{K}_{1:t-1}\mathbf{K}_{1:t-1}^T)^{-1}. \tag{23}$$

AlphaEdit: AlphaEdit inherently supports sequential editing by design. Its update already includes protection for previously edited knowledge:

$$\Delta_{\text{AlphaEdit}} = \mathbf{R}\mathbf{K}_t^T\mathbf{P} \left(\mathbf{I} + \mathbf{K}_{1:t-1}\mathbf{K}_{1:t-1}^T\mathbf{P} + \mathbf{K}_t\mathbf{K}_t^T\mathbf{P}\right)^{-1}, \tag{24}$$

where $\mathbf{P}$ is a projection matrix onto the null space of preserved knowledge. Hence, no further adaptation is required.

D Quantifying Edit Safety: A Geometric Criterion Based on Key Space Alignment

When performing knowledge editing, it is essential to assess not only whether an edit can be successfully implemented, but also whether it is likely to interfere with existing knowledge. In this section, we introduce a principled geometric criterion—the *Orthogonal Energy*—that quantifies the potential safety of an edit by measuring how much of the new key's direction lies in underutilized, orthogonal regions of the model's key space.

Recall from Section **??** that the pre-edit keys $\mathbf{K}_0$ exhibit a low-rank plus diagonal (LR+D) covariance structure:

$$\mathbf{\Sigma} = \mathbf{U}\mathbf{U}^\top + \mathbf{D},$$

where $\mathbf{U} \in \mathbb{R}^{d\times r}$ captures the top-r principal directions of variation (i.e., the semantic subspace), and $\mathbf{D} \in \mathbb{R}^{d\times d}$ is a diagonal matrix representing residual variances. This structure reflects the fact that natural inputs tend to concentrate in a low-dimensional manifold within the high-dimensional key space.

During editing, we regularize the weight update $\Delta \in \mathbb{R}^{o\times d}$ via:

$$\lambda \cdot \mathrm{Tr}\left(\Delta^\top \Delta \left(\mathbf{U}\mathbf{U}^\top + \mathbf{D}\right)\right),$$

which penalizes changes that act strongly along directions of high variance—i.e., within $\mathrm{span}(\mathbf{U})$. Intuitively, this means:

- Directions in $\mathrm{span}(\mathbf{U})$ are "expensive" to modify, as they are densely populated with existing knowledge.
- Directions in the orthogonal complement $\mathbf{U}^\perp$ are "cheap" to use, as they correspond to underutilized regions of the key space.

Therefore, edits whose keys lie primarily in $\mathbf{U}^\perp$ are less disruptive and more likely to preserve existing behavior.

D.1 Orthogonal Energy: Measuring Alignment with the Quiet Subspace

Given a new key $\mathbf{k}_1^{(i)} \in \mathbb{R}^d$, we assess its edit safety by measuring how much of its energy lies in the orthogonal complement $\mathbf{U}^\perp$. The core idea is:

The safer a key is, the more of its energy resides in $\mathbf{U}^\perp$, the underutilized "quiet" region of the key space.

Let $\mathbf{P}_\mathbf{U} = \mathbf{U}(\mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top$ denote the orthogonal projection onto $\mathrm{span}(\mathbf{U})$. Then $(\mathbf{I} - \mathbf{P}_\mathbf{U})$ projects onto $\mathbf{U}^\perp$.

[Orthogonal Energy] For a new key $\mathbf{k}_1^{(i)}$, its **Orthogonal Energy** is defined as:

$$e_i = \|(\mathbf{I} - \mathbf{P}_\mathbf{U})\mathbf{k}_1^{(i)}\|^2, \tag{25}$$

with $e_i = 0$ if $\mathbf{k}_1^{(i)} = 0$.

Interpretation:

- $e_i \gg 0$: The key has strong components in $\mathbf{U}^\perp$. It lies in a "quiet" region of the key space, and the edit can be implemented with minimal interference. *High safety.*
- $e_i \approx 0$: The key is nearly aligned with $\mathrm{span}(\mathbf{U})$. Modifying the model to fit it may disturb many existing representations. *Low safety.*
- Unlike normalized scores, e_i preserves the magnitude of the key, making it sensitive to both direction and scale—critical for detecting whether a large-magnitude edit is forced into the semantic subspace.

Geometric intuition: The quantity e_i measures the squared length of $\mathbf{k}_1^{(i)}$'s projection onto the orthogonal complement of the semantic subspace. Larger values indicate that the edit operates in a region where existing knowledge is sparse, reducing the risk of interference.

D.2 Subspace Interference: A Post-hoc Measure of Update Impact

While the orthogonal energy e_i provides a *prior* assessment of edit safety based on the input key, it is also valuable to measure the actual impact of an edit on the model's parameter space. To this end, we introduce the *subspace interference* metric:

$$s_i = \|\Delta_i \mathbf{U}\|_F,$$

where $\Delta_i \in \mathbb{R}^{o\times d}$ is the weight update applied at the i-th edit, and $\|\cdot\|_F$ denotes the Frobenius norm. This metric quantifies how much of the update Δ_i acts along the directions spanned by $\mathbf{U}$—i.e., within the semantic subspace where existing knowledge is concentrated.

A large s_i indicates that the edit strongly modifies parameters associated with core semantic directions, increasing the risk of interference with unrelated facts. In contrast, small s_i suggests that the update is confined to orthogonal or residual directions, preserving the integrity of existing representations.

Crucially, s_i and e_i are theoretically connected through the regularization objective. Recall from Section D that our regularizer penalizes updates via:

$$\lambda \cdot \mathrm{Tr}\left(\Delta^\top \Delta \left(\mathbf{U}\mathbf{U}^\top + \mathbf{D}\right)\right).$$

The component $\mathrm{Tr}(\Delta^\top \Delta\, \mathbf{U}\mathbf{U}^\top) = \|\Delta\mathbf{U}\|_F^2 = s_i^2$ directly measures the energy of the update in the semantic subspace. Thus, minimizing the regularized objective encourages *both* high e_i (via key alignment) *and* low s_i (via update sparsity in $\mathbf{U}$).

In practice, s_i serves as a *post-hoc diagnostic*: while e_i can be computed before applying the edit, s_i requires access to Δ_i and is used to verify whether the update respected the intended geometric constraints. When both metrics trend together—e.g., declining e_i and rising s_i over sequential edits—they provide converging evidence of deteriorating edit safety.

D.3 Why This Metric Works: Connection to Update Cost

To understand why e_i predicts edit safety, consider the minimal Frobenius-norm update Δ that satisfies $\Delta\mathbf{k}_1^{(i)} = \mathbf{r}_i$, where $\mathbf{r}_i = \mathbf{v}_1^{(i)} - \mathbf{W}\mathbf{k}_1^{(i)}$ is the residual. This update takes the outer-product form:

$$\Delta = \mathbf{r}_i \mathbf{k}_1^{(i)\top} / \|\mathbf{k}_1^{(i)}\|^2.$$

Each row Δ_j is proportional to $\mathbf{k}_1^{(i)\top}$, scaled by $(\mathbf{r}_i)_j$. The regularizer evaluates:

$$\mathrm{Tr}\left(\Delta^\top \Delta \left(\mathbf{U}\mathbf{U}^\top + \mathbf{D}\right)\right) \propto \sum_j (\mathbf{r}_i)_j^2 \cdot \mathbf{k}_1^{(i)\top}(\mathbf{U}\mathbf{U}^\top + \mathbf{D})\mathbf{k}_1^{(i)}.$$

We can decompose the quadratic form:

$$\mathbf{k}_1^{(i)\top}\mathbf{U}\mathbf{U}^\top\mathbf{k}_1^{(i)} = \|\mathbf{P}_\mathbf{U}\mathbf{k}_1^{(i)}\|^2 = \|\mathbf{k}_1^{(i)}\|^2 - \|(\mathbf{I}-\mathbf{P}_\mathbf{U})\mathbf{k}_1^{(i)}\|^2 = \|\mathbf{k}_1^{(i)}\|^2 - e_i.$$

Therefore, the regularization cost is:

$$\text{cost} \propto \sum_j (\mathbf{r}_i)_j^2 \cdot \left(\|\mathbf{k}_1^{(i)}\|^2 - e_i + \mathbf{k}_1^{(i)\top}\mathbf{D}\mathbf{k}_1^{(i)}\right).$$

For fixed $\mathbf{k}_1^{(i)}$ and $\mathbf{r}_i$, the cost is *minimized* when e_i is *maximized*. That is: ¿ **Higher orthogonal energy e_i leads to lower regularization cost**, meaning the edit can be implemented more easily and with less interference.

Thus, e_i serves as a prior indicator: edits with high e_i are geometrically favored by the regularization and are less likely to distort existing representations.

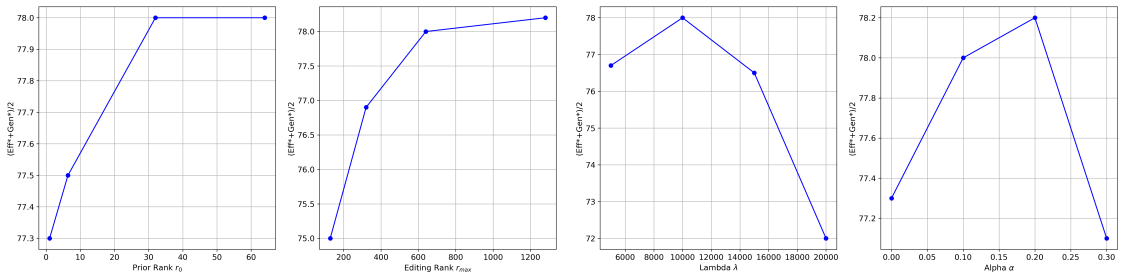

Figure 5: Hyperparameter sensitivity analysis on GPT2-XL (CounterFact).

D.4 Practical Usage and Interpretation

In practice, e_i serves as a *prior* indicator of edit safety:

- High e_i: Edit lies in a quiet region. Likely to succeed with minimal interference. Can be prioritized or applied with lower regularization.
- Low e_i: Edit is forced into the semantic subspace. High risk of interference. Should be monitored or rejected if safety is critical.

We recommend computing e_i for all new entries and using it to:

- Rank edits by safety (descending e_i).
- Set adaptive regularization (e.g., reduce λ for high-e_i edits).
- Diagnose failure cases (e.g., low e_i correlates with specificity drop or generalization failure).

E Results

E.1 Editing Performance on GPT2-XL

Table 2: Editing efficacy.

Method		CounterFact						ZsRE		
		Eff.↑	**Gen.↑**	**Spe.↑**	**Eff*.↑**	**Gen*.↑**	**Spe*.↕**	**Eff*.↑**	**Gen*.↑**	**Spe*.↕**
Pre-edited		22.1	24.4	78.0	0.10	0.40	10.6	23.7	22.8	25.0
MEMIT	GPT2-XL	98.0	88.6	65.7	91.4	58.4	10.6	94.7	88.4	26.9
PMET		97.8	90.0	61.3	90.6	62.6	8.80	96.1	91.4	26.1
EMMET		92.4	85.6	57.1	71.1	49.9	5.50	85.0	77.8	24.3
AlphaEdit		99.6	94.0	65.7	97.3	65.3	6.40	96.0	88.8	26.8
RECT		98.0	88.6	65.4	90.9	58.0	10.8	94.8	88.5	26.8
PRUNE		98.0	88.4	65.7	91.5	58.5	10.7	95.3	89.0	26.6
AdaEdit		96.2	87.9	60.8	86.9	60.4	9.20	93.8	88.3	26.0
FastEdit		98.2	91.3	61.4	92.2	64.2	9.10	94.8	89.2	26.0

E.2 HYPERPARAMETER INVESTIGATION

We analyze the sensitivity of FastEdit to four key hyperparameters: the subspace rank r_0 of $\mathbf{U}$, the maximum edit rank r_{max}, the regularization coefficient λ, and the prior fusion coefficient α. As shown in Figure 5, performance is robust for $r_0 \geq 30$, indicating that a small-dimensional semantic subspace suffices to capture essential knowledge structure in GPT2-XL. This validates the low-rank assumption underlying our safety metric and supports efficient pre-computation. The editing capacity, controlled by r_{max}, saturates around 600, beyond which further increases yield diminishing returns. This suggests that the representational complexity of editing knowledge—across diverse factual updates—is inherently low-rank, justifying our compressed update design. For regularization strength λ, we observe that when λ is too large ($> 1.5 \times 10^4$), the update is over-constrained, significantly limiting model adaptability and preventing effective knowledge injection. Finally, the prior fusion coefficient α controls the trade-off between data-driven covariance and structural prior $\mathbf{C}_{\text{prior}}$. Performance peaks at $\alpha \in [0.1, 0.2]$, confirming that combining empirical statistics with the MLP down-projection-based prior improves robustness—especially under limited sampling.