OrthoEdit: Principled and Stable Knowledge Editing via Orthogonal Subspace Projection

Shanbao Qiao[1], Xuebing Liu[1], Akshat Gupta[3], Seung-Hoon Na[2] *
[1]Department of Computer Science and Artificial Intelligence,
Jeonbuk National University
[2]Graduate School of Artificial Intelligence,
Ulsan National Institute of Science and Technology [3]UC Berkeley
{joe,liuxuebing}@jbnu.ac.kr, nash@unist.ac.kr

Abstract

Large language models (LLMs) encode extensive factual knowledge through pretraining, yet often require targeted updates to correct errors, incorporate new information, or revise outdated facts. Recent approaches to knowledge editing, such as projection-based constraints, parameter pruning, and regularization, have proven effective in improving editing accuracy and stability. However, these methods often fail to maintain a clear separation between new edits and existing knowledge, leading to interference and degradation over time. We propose **OrthoEdit**, a principled framework for stable and scalable knowledge editing that ensures each parameter update is orthogonal to both pre-existing and previously edited knowledge, remains strictly non-interfering and preserving the integrity of prior edits. OrthoEdit enables exact subspace control through three coordinated steps: progressive null space refinement, principal subspace extraction, and orthogonal projection. This yields compact and well-aligned updates that systematically satisfy all accumulated constraints. Comprehensive experiments across diverse models and benchmarks demonstrate that OrthoEdit consistently enhances editing accuracy and robustness while preserving general capabilities—even through extended sequences of batched edits. Our code is available at https://github.com/JoveReCode/OrthoEdit.git.

1 Introduction

Large language models (LLMs) excel at many NLP tasks by encoding vast factual knowledge during pretraining (Touvron et al., 2023; OpenAI, 2023; Yang et al., 2024; Guo et al., 2025). However, this knowledge can be inaccurate, outdated, or harmful (Onoe et al., 2022; Ji et al., 2023; Zhao et al., 2023; Gallegos et al., 2023; Farquhar et al.; Mousavi et al., 2024). Given the dynamic nature of real-world information, efficiently updating model content is increasingly crucial. Retraining or full-model fine-tuning imposes significant computational and temporal costs, rendering it unsuitable for rapid knowledge updates. Thus, knowledge editing methods (Yao et al., 2023; Zhang et al., 2024c; Xu et al., 2025) that enable targeted, lightweight model updates have attracted growing interest.

Existing knowledge editing methods can be broadly categorized into two paradigms: *parameter-updating* approaches and *knowledge-preserving* approaches. Parameter-updating methods (Meng et al., 2022b; Li et al., 2023; Qiao et al., 2024b) typically modify the parameters of feedforward network (FFN) layers to encode new facts, enabling durable updates and allowing for batch-wise editing. In contrast, knowledge-preserving methods (Hartvigsen et al., 2023; Qi et al., 2024; Qiao et al., 2024a) avoid weight changes by leveraging external modules or prompting strategies, but often incur additional memory costs or offer only temporary editing.

Editing tasks vary in scale, ranging from single-instance to large-scale batched sequential edits. The latter presents unique challenges: it demands both high update capacity and long-term stability. Knowledge-preserving approaches struggle to scale due to reliance on lengthy prompts(Qi et al., 2024) or retrieval(Hartvigsen et al., 2023), while parameter-updating methods risk interference and forgetting over extended edit sequences.

AlphaEdit (Fang et al., 2024) projects updates onto the null space of pre-existing knowledge, reducing interference and improving locality during sequential editing. However, it still preserves prior edits via conventional constraints, leaving them susceptible to interference and forgetting in sub-

* Corresponding author.

Transactions of the Association for Computational Linguistics, vol. 14, pp. 1140–1162, 2026. https://doi.org/10.1162/TACL.a.700
Action Editor: Daisuke Kawahara. Submission batch: 6/2025; Revision batch: 11/2025; Published 6/2026.

sequent updates. Another contemporaneous line of work, O-Edit (Cai and Cao, 2024), while also termed orthogonal subspace editing, introduces gradient-based regularization into the training loss to penalize the cosine similarity between the gradient directions of different knowledge. However, it imposes soft constraints with additional hyperparameters during training, rather than operating exactly on the update space.

To overcome these limitations, we propose **OrthoEdit**, a projection-based framework that formalizes principled orthogonal subspace editing for stable and scalable batched sequential knowledge editing. The core idea is to ensure that each parameter update is confined to a subspace that is simultaneously orthogonal to both the model's pre-existing knowledge and all previously edited knowledge keys. This dual orthogonality prevents interference across editing rounds and ensures that each update remains localized and non-disruptive.

Our method recursively constructs the shared null space via progressive projection that eliminates directions aligned with prior edits. It further extracts a compact principal subspace from the current target knowledge to reduce redundancy and conserve null space capacity for future edits. The final update is projected onto their orthogonal intersection, enabling precise updates that satisfy all accumulated constraints. Through this design, OrthoEdit provides a principled and efficient solution for stable and sustainable large-scale editing in large language models.

Our contributions are summarized as follows:

- We introduce OrthoEdit, a novel projection-based knowledge editing framework that supports stable and scalable batched sequential edits by explicitly eliminating interference from both pre-existing and previously edited knowledge.
- We design a structured subspace construction procedure comprising progressive null space refinement, principal subspace extraction, and orthogonal alignment, enabling precise updates and ensure robustness over long editing sequences.
- We provide a formal analysis demonstrating that OrthoEdit rigorously satisfies the required orthogonality constraints, and we validate its effectiveness through comprehensive experiments across multiple models and benchmarks, consistently surpassing prior works in editing accuracy and stability.

2 Related Works

Model editing methods, aimed at updating factual knowledge in large language models, are broadly categorized into parameter-updating and parameter-preserving approaches.

2.1 Parameter-Updating Editing Methods

Parameter-updating methods can be broadly categorized into locate-then-edit and meta-learning approaches. Locate-then-edit methods identify and modify parameters encoding specific facts using causal tracing. ROME (Meng et al., 2022a), MEMIT (Meng et al., 2022b), and PMET (Li et al., 2023) perform activation path analysis for localized updates, while GLAME (Zhang et al., 2024a) enhances this with knowledge graph signals. RECT (Gu et al., 2024b) regularizes weight updates during editing to prevent overfitting. PRUNE (Ma et al., 2024) enforces condition-number constraints on the edited weight matrix to bound numerical perturbations. AnyEdit (Jiang et al., 2025) generalizes editing across arbitrary formats, and AlphaEdit (Fang et al., 2024) introduces subspace projection to reduce interference by projecting updates with the null space of existing knowledge.

Meta-learning approaches instead train auxiliary hypernetworks to generate parameter updates from few-shot examples. KE (Cao et al., 2021), MEND (Mitchell et al., 2022a), and MALMEN (Tan et al., 2024) design generalizable editing pipelines, and InstructEdit (Zhang et al., 2024b) further enables task-specific behavior through instruction tuning.

Aligned with our high-level goal of interference-free editing, recent effort O-Edit (Cai and Cao, 2024) also explores orthogonal subspace editing, but via loss-level regularization that penalizes the cosine similarity between gradient directions. While similar in motivation, this approach is fundamentally different from ours. Our OrthoEdit achieves interpretable and principled control over interference via an efficient projection-based mechanism, faithfully capturing the essence of orthogonal subspace editing.

2.2 Parameter-Preserving Editing Methods

On the other hand, parameter-preserving methods avoid modifying pretrained weights to maintain the integrity of original knowledge. These fall into two categories: memory-augmented and inference-based. Memory-augmented methods incorporate new facts via lightweight external modules, for instance, MELO (Yu et al., 2024) uses additional LoRA (Hu et al., 2022) blocks, T-Patcher (Huang et al., 2023) introduces auxiliary neurons, SERAC (Mitchell et al., 2022b) adds a sub-network, and MEMoE (Wang and Li, 2024) employs a mixture-of-experts for routing. Inference-based methods, such as ICE (Qi et al., 2024), retain the model architecture and employ prompting and in-context reasoning to retrieve or infer edited knowledge.

2.3 Extended Knowledge Editing Tasks

Beyond the standard formulation of knowledge editing, a number of studies have explored alternative paradigms, including multi-hop editing (Zhong et al., 2023; Gu et al., 2024a; Bi et al., 2024; Lu et al., 2024), editing commonsense knowledge (Huang et al., 2024), modeling the ripple effects of edits (Cohen et al., 2024), event-level editing (Liu et al., 2024), and long-form evaluation (Rosati et al., 2024). These works broaden the scope of the field and introduce new challenges that further drive the development of knowledge editing research.

3 Preliminaries

In a large language model (LLM) that predicts the next token x based on the current context, the hidden representation of token x at layer l is denoted as h^l, which is computed as:

$$\begin{aligned} h^l &= h^{l-1} + a^l + m^l, m^l \\ &= \mathbf{W}^l_{out}\sigma(\mathbf{W}^l_{in}\gamma(h^{l-1} + a^l)), \end{aligned} \tag{1}$$

where a^l and m^l represent the outputs of the attention block and the feed-forward network (FFN) layer, respectively, σ is the non-linear activation function, and γ denotes the layer normalization. $\mathbf{W}^l_{in}$ and $\mathbf{W}^l_{out}$ are the weight matrices of the FFN layer. For simplicity and consistent with previous work, we use $\mathbf{W}$ to denote $\mathbf{W}^l_{out}$ in this context.

Following previous works (Meng et al., 2022b; Fang et al., 2024), where $\mathbf{W}$ is interpreted as a linear associative memory, linear operation $\mathbf{W}$ can act as a key–value store for a set of vector keys $\mathbf{K}$ and corresponding vector values $\mathbf{V}$, by approximately satisfying: $\mathbf{WK} \approx \mathbf{V}$. Specifically, suppose each edit involves updating u factual triples in the form of (s, r, o). The updated matrix $\mathbf{W}$ is expected to associate u new key–value pairs, where k and v encode the (s, r) and o of the new knowledge, respectively.

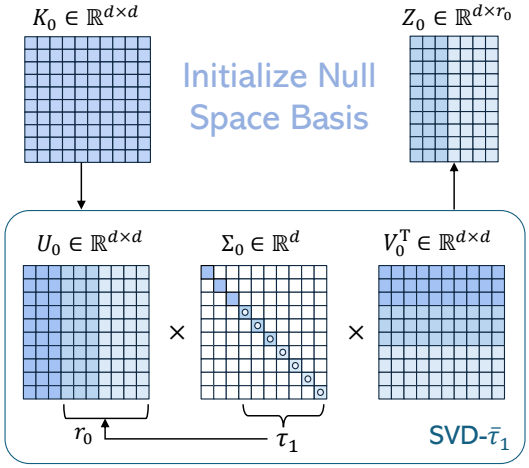

Figure 1: Illustration of null space basis initialization. SVD-$\bar{\tau}_1$ denotes performing SVD and selecting the columns from left singular matrix $\mathbf{U}_0$ corresponding to the singular values below the threshold τ_1. The resulting null space basis aligns with that used in AlphaEdit.

Parameter-updating methods, such as MEMIT (Meng et al., 2022b), follow a *locate-then-edit* paradigm in which an update matrix $\mathbf{\Delta}$ is directly added to the pre-trained model parameters $\mathbf{W}$. Ideally, the pretrained model satisfies: $\mathbf{WK} = \mathbf{V}$, where $\mathbf{K}$ and $\mathbf{V}$ denote the matrix of keys and values, respectively. For the target knowledge set $\mathbf{K}_t$ with desired values $\mathbf{V}_t$ that need to be updated, the model generally fails to satisfy the mapping, i.e., $\mathbf{WK}_t \neq \mathbf{V}_t$.

Knowledge editing therefore seeks an update $\mathbf{\Delta}$ such that: $(\mathbf{W} + \mathbf{\Delta})\mathbf{K}_t = \mathbf{V}_t$, while preserving all unrelated pre-existing knowledge encoded in $\mathbf{K}_{\text{base}}$ with values $\mathbf{V}_{\text{base}}$. MEMIT realizes this by solving the following objective:

$$\begin{aligned} \mathbf{\Delta} = \arg\min_{\tilde{\mathbf{\Delta}}} &\left\|(\mathbf{W} + \tilde{\mathbf{\Delta}})\mathbf{K}_t - \mathbf{V}_t\right\|^2 \\ &+\left\|(\mathbf{W} + \tilde{\mathbf{\Delta}})\mathbf{K}_{\text{base}} - \mathbf{V}_{\text{base}}\right\|^2. \end{aligned} \tag{2}$$

In practice, $\mathbf{K}_{\text{base}} \in \mathbb{R}^{d\times 100{,}000}$ is constructed by encoding 100,000 factual samples from Wikipedia corpus, serving as an approximation of the

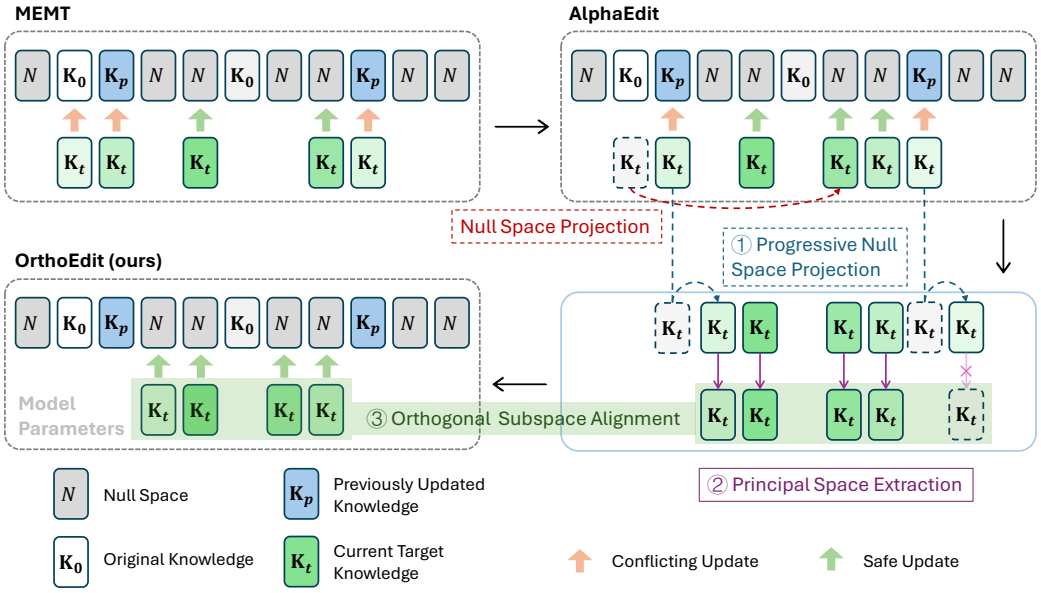

Figure 2: Overview of OrthoEdit. When integrating the target knowledge $\mathbf{K}_t$ into model parameters, MEMIT directly updates model parameters, often causing interference with original knowledge $\mathbf{K}_0$ and previously edited knowledge $\mathbf{K}_p$. AlphaEdit alleviates this by projecting updates onto the null space of $\mathbf{K}_0$, but prior edits $\mathbf{K}_p$ may still be affected. In contrast, OrthoEdit employs a progressive null space projection to map updates onto the joint null space of $\mathbf{K}_0$ and $\mathbf{K}_p$, while extracting the principal subspace of the current target knowledge $\mathbf{K}_t$. By aligning updates within this orthogonal subspace, OrthoEdit enables safe parameter modification that preserves both previously edited and original knowledge.

model's internal factual memory. Here, d is the size of the FFN hidden layer.

Directly preserving $\mathbf{K}_{\text{base}}$ via the above objective still suffers from interference when multiple edits accumulate. AlphaEdit (Fang et al., 2024) addresses this by constraining updates to the null space of $\mathbf{K}_{\text{base}}$. The null space is defined as follows: given two matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathbf{B}$ lies in the null space of $\mathbf{A}$ if and only if $\mathbf{BA} = \mathbf{0}$.

The motivation for applying a null space projection in model editing is to preserve existing knowledge during parameter updates. Specifically, when the update parameter $\mathbf{\Delta}$ is projected onto the null space of $\mathbf{K}_{base}$ (i.e., $\mathbf{\Delta_p}\mathbf{K}_{base} = \mathbf{0}$, where $\mathbf{\Delta_p}$ denotes the projected update parameter), adding it to the parameter matrix $\mathbf{W}$ yields:

$$(\mathbf{W} + \mathbf{\Delta}_p)\mathbf{K}_{base} = \mathbf{W}\mathbf{K}_{base} = \mathbf{V}_{base}. \quad (3)$$

This property ensures that the projected update parameter $\mathbf{\Delta}_p$ does not interfere with the key–value associations of the preserved knowledge pair $\{\mathbf{K}_{base}, \mathbf{V}_{base}\}$, thereby maintaining the integrity of the stored information. Therefore, before incorporating the update parameter into the model parameters, AlphaEdit first project it onto the null space of $\mathbf{K}_{base}$ to safeguard the existing knowledge. This mechanism effectively eliminates the need for an explicit regularization term dedicated to protecting the preserved knowledge, as such protection is inherently guaranteed by the projection itself.

Due to the high dimensionality of $\mathbf{K}_{\text{base}}$, it operates on its uncentered covariance matrix $\mathbf{K}_0 = \mathbf{K}_{\text{base}}\mathbf{K}_{\text{base}}^\top \in \mathbb{R}^{d\times d}$, which shares the same null space with $\mathbf{K}_{\text{base}}$. Specifically, it performs Singular Value Decomposition (SVD) on the symmetric matrix: $\mathbf{K}_0 = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$. Let $\hat{\mathbf{U}}$ denote the eigenvectors associated with near-zero[1] singular values. The null space projector is then defined as $\mathbf{P} = \hat{\mathbf{U}}\hat{\mathbf{U}}^\top$. The projected update is given by $\mathbf{\Delta}_{\text{Alpha}} = \mathbf{\Delta}\mathbf{P}$, which satisfies $\mathbf{\Delta}\mathbf{P}\mathbf{K}_{\text{base}} = 0$, thereby eliminating the $\mathbf{K}_{\text{base}}$ term from the update objective.

However, it is also crucial for sequential editing to preserve previously edited knowledge, denoted as $\mathbf{K}_p = \{\mathbf{K}_1, \ldots, \mathbf{K}_{t-1}\}$ across timesteps. Although AlphaEdit isolates updates from $\mathbf{K}_{\text{base}}$, it does not prevent interference with $\mathbf{K}_p$, which

[1] As eigenvalues are seldom strictly zero, it is common to use an approximation.

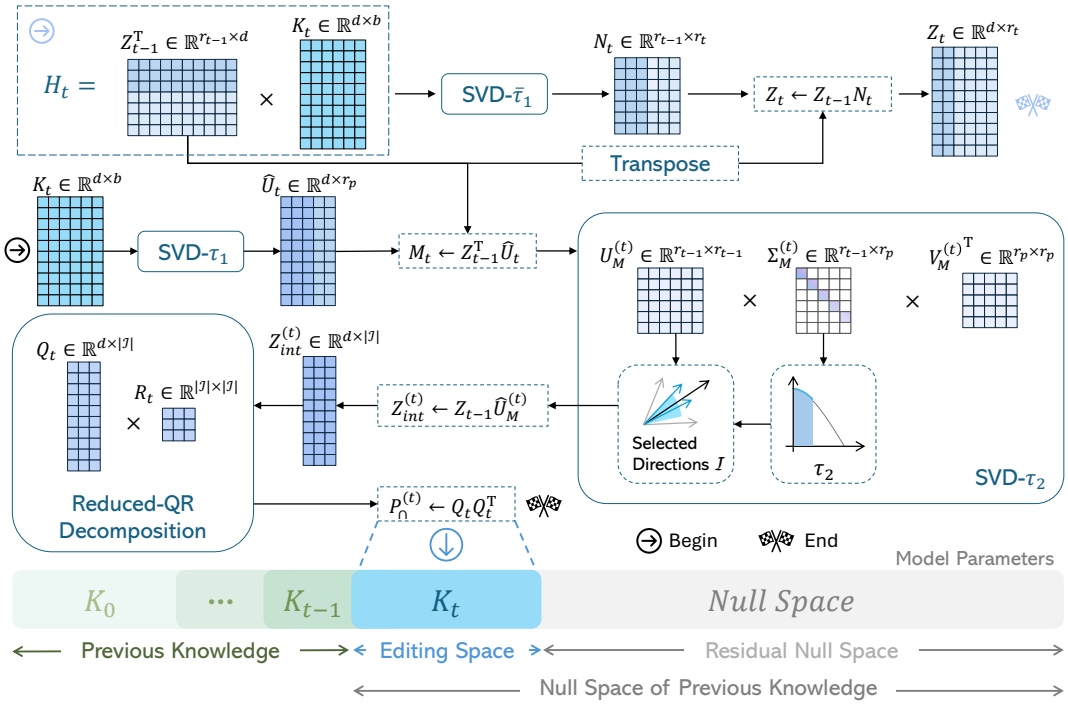

Figure 3: Detailed workflow of the orthogonal subspace projector construction. Given the current knowledge key matrix $\mathbf{K}_t$ and the null space basis $\mathbf{Z}_{t-1}$ from the last timestep, we compute the $\mathbf{H}_t = \mathbf{Z}_{t-1}^{\top}\mathbf{K}_t$, the upper stream illustrate progressive null space projection that updates $\mathbf{Z}_t$, while the lower stream shows principal space extraction and subspace alignment with orthogonalization. SVD-$\bar{\tau}_1$ denotes selecting the columns from left singular matrix with singular values below the threshold τ_1, SVD-τ_1 and SVD-τ_2 denote selecting those above the respective thresholds. The resulting projector $\mathbf{P}_{\cap}$ maps the parameter updates into the joint orthogonal null space of all prior edits and pre-existing knowledge.

is instead handled using a conventional constraint term: $\left\|(\mathbf{W}+\mathbf{\Delta P})\mathbf{K}_p - \mathbf{V}_p\right\|^2$[2]. As edits accumulate and $\mathbf{K}_p$ grows over time, maintaining compatibility with all accumulated constraints becomes increasingly difficult. Consequently, earlier edits may still be disrupted by subsequent updates, limiting long-term stability in sequential settings.

Compared with AlphaEdit, our proposed OrthoEdit (1) replaces soft preservation of previous edits with an incremental joint null-space that spans both the base knowledge $\mathbf{K}_0$ and all prior edited keys $\{\mathbf{K}_p\}$, thereby eliminating cross-batch interference; (2) performs principal subspace extraction to compress each update into the most informative directions of the current key $\mathbf{K}_t$, reducing redundancy and conserving null-space capacity for future edits; (3) computes an exact orthogonal intersection via subspace alignment, yielding a closed-form idempotent, symmetric projector $\mathbf{P}_{\cap}$ that simultaneously satisfies accumulated constraints; and (4) delivers stronger long-term stability, extending reliable sequential editing while sustaining accuracy and general capability under more challenging settings. The overview of our framework as shown in Figure 2.

4 Method

We propose OrthoEdit, which enforces a stronger non-interference constraint by progressively projecting the update parameter $\mathbf{\Delta}$ into the joint null space of all encoded knowledge keys, $\mathbf{K}_{\text{base}}$ and $\mathbf{K}_p$, thereby ensuring that both pre-existing knowledge and previous edits remain intact, thus improving overall robustness and stability. Furthermore, our orthogonal subspace projection reduces the dimensional cost of progressive projection and constrains the update to occur within a compact and informative subspace, allowing for precise and stable knowledge editing.

[2]Since the knowledge from previous batches has already been updated and thus satisfies $\mathbf{W}\mathbf{K}_p = \mathbf{V}_p$, the term $\left\|(\mathbf{W}+\mathbf{\Delta P})\mathbf{K}_p - \mathbf{V}_p\right\|^2$ can be simplified to $\left\|\tilde{\mathbf{\Delta}}\mathbf{P}_0\mathbf{K}_p\right\|^2$.

4.1 Progressive Null Space Projection

To prevent interference with both pre-existing knowledge and previously edited knowledge, we constrain the update parameter $\mathbf{\Delta}$ to lie in the joint null space of $\mathbf{K}_{\text{base}}$ and all prior edited keys $\mathbf{K}_p = \{\mathbf{K}_i\}_{i=1}^{t-1}$. A straightforward approach would project $\mathbf{\Delta}$ into the null space of a concatenated matrix:

$$\mathbf{Y}_{\leq t} = [\mathbf{K}_0\, \mathbf{K}_1\, \cdots\, \mathbf{K}_{t-1}] \in \mathbb{R}^{d\times B}, \quad (4)$$

where $\mathbf{K}_0 \in \mathbb{R}^{d\times d}$ and each $\mathbf{K}_i \in \mathbb{R}^{d\times b}$, b is the editing batch size, and $B = b(t-1)+d$. However, as the number of editing steps increases, the size of $\mathbf{Y}_{\leq t}$ grows linearly with time, computing its null space becomes increasingly costly.

To address this scalability challenge, we propose Progressive Null Space Projection, which incrementally updates the null space basis by incorporating one knowledge constraint $\mathbf{K}_i$ at a time. This eliminates the need to explicitly construct the large concatenated matrix $\mathbf{Y}_{\leq t}$, while ensuring that $\mathbf{\Delta}$ remains orthogonal to all previously edits.

At each timestep t, we construct a projector $\mathbf{P}_{\text{null}}^{(t)}$ that ensures the update parameter $\mathbf{\Delta}$ remains orthogonal to all previously edited knowledge keys. Formally, the projection satisfies:

$$\mathbf{\Delta}\mathbf{P}_{\text{null}}^{(t)}\mathbf{K}_i = \mathbf{0}, \quad \text{for all } i < t. \quad (5)$$

We adopt the null space projection from AlphaEdit (Fang et al., 2024) as our initialization step. Specifically, we perform Singular Value Decomposition (SVD) on the uncentered covariance matrix $\mathbf{K}_0$, which shares the same null space as $\mathbf{K}_{\text{base}}$, yielding $\mathbf{K}_0 = \mathbf{U}_0\mathbf{\Sigma}_0\mathbf{U}_0^\top$. We then retain $\mathbf{Z}_0 \in \mathbb{R}^{d\times r_0}$, the submatrix of $\mathbf{U}_0$ whose columns correspond to singular values below the threshold τ_1, as the initial null space basis, where $r_0 = d - \text{rank}_{\tau_1}(\mathbf{\Sigma}_0)$, $\text{rank}_{\tau_1}(\cdot)$ counts singular values greater than a predefined threshold τ_1, as illustrated in Figure 1.

Whereas AlphaEdit uses a fixed projector $\mathbf{P}_{\text{null}} = \mathbf{Z}_0\mathbf{Z}_0^\top$ throughout the entire sequential editing, we progressively update the null space by integrating each new edit batch $\mathbf{K}_i$ into the constraint, enabling the null space basis to evolve with each editing step.

To achieve this, at step t, we update the null space basis $\mathbf{Z}_{t-1}$ by eliminating its components that interfere with $\mathbf{K}_t$. We first construct the *interference matrix*:

$$\mathbf{H}_t = \mathbf{Z}_{t-1}^\top\mathbf{K}_t \in \mathbb{R}^{r_{t-1}\times b}, \quad (6)$$

which captures the interaction between the current null space basis and the new target knowledge. Here, r_{t-1} denotes the column dimension of $\mathbf{Z}_{t-1}$. We then perform SVD: $\mathbf{H}_t = \mathbf{U}_H\mathbf{\Sigma}_H\mathbf{V}_H^\top$, and extract the left null space of $\mathbf{H}_t$ by selecting the columns of $\mathbf{U}_H$ corresponding to singular values below the threshold τ_1. The resulting submatrix is denoted as $\mathbf{N}_t \in \mathbb{R}^{r_{t-1}\times r_t}$, where $r_t = r_{t-1} - \text{rank}_{\tau_1}(\mathbf{\Sigma}_H)$. The null space basis is recursively updated as:

$$\mathbf{Z}_t = \mathbf{Z}_{t-1}\mathbf{N}_t. \quad (7)$$

This recursive process allows the null space basis $\mathbf{Z}_t$ to evolve at each editing step, and the progressive null space projector can then be constructed as:

$$\mathbf{P}_{\text{null}}^{(t)} = \mathbf{Z}_{t-1}\mathbf{Z}_{t-1}^\top. \quad (8)$$

We provide a detailed proof in the Appendix D demonstrating that our construction procedure satisfies the constraint in Equation 5.

Moreover, our method reduces the computational complexity of the joint null space from quadratic to linear in time-step T. Specifically, for the naive matrix concatenation approach, at the t-th editing step, the concatenated matrix will have a dimension of $d_t = d + B \times t$. Summing over all T batches, the overall computational complexity becomes approximately:

$$\sum_{t=1}^{T}\mathcal{O}(d + B\times t) = \mathcal{O}(dT + BT^2) \quad (9)$$

which scales quadratically with T. In contrast, our incremental null space computation caches Z_t at each step and only requires computing the SVD on a matrix of size B, resulting in a total computational complexity of:

$$\sum_{t=1}^{T}\mathcal{O}(B) = \mathcal{O}(BT) \quad (10)$$

thereby reducing the overall complexity from quadratic to linear in T.

4.2 Principal Space Extraction

Although the update parameter $\mathbf{\Delta}$ has been constrained to the joint null space basis $\mathbf{Z}_t$, this subspace may still contain redundant directions that are not essential for updating the current target knowledge $\mathbf{K}_t$. The effectiveness of parameter

pruning methods such as Gu et al. (2024b); Qiao et al. (2025); Li and Chu (2024) also suggests that update parameters typically exhibit redundancy.

Moreover, as each batch of edit consumes a portion of the null space dimensions, it is crucial to compress the update space into a lower-dimensional principal subspace tailored to $\mathbf{K}_t$. This dimensional compression confines parameter updates within a compact and informative subspace, simultaneously mitigating parameter redundancy and preserving more null space capacity for subsequent edits, thereby enhancing the sustainability for long-term editing.

To construct this principal subspace, we perform SVD as: $\mathbf{K}_t = \mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^\top$, and retain the top r_p left singular vectors corresponding to the largest singular values. We denote this resulting matrix as $\hat{\mathbf{U}}_t \in \mathbb{R}^{d\times r_p}$, where $r_p = \text{rank}_{\tau_1}(\mathbf{\Sigma}_t)$. The columns of $\hat{\mathbf{U}}_t$ form an orthonormal basis capturing the most expressive directions of the current knowledge key in the parameter space.

This principal subspace serves as an intermediate constraint, enabling subsequent alignment and orthogonalization procedures to precisely control the update direction.

4.3 Orthogonal Projector from Subspace Alignment

After obtaining the principal subspace basis $\hat{\mathbf{U}}_t$ from the current target knowledge and the updated null space basis $\mathbf{Z}_{t-1}$, we aim to unify them into a coherent subspace to precisely control the update direction. However, since $\hat{\mathbf{U}}_t$ and $\mathbf{Z}_{t-1}$ are generally not orthogonal, naïvely combining them may violate previously enforced constraints and potentially compromising both precision and edit locality. To address this issue, we quantify their interaction by computing the *alignment matrix* as:

$$\mathbf{M}_t = \mathbf{Z}_{t-1}^\top\hat{\mathbf{U}}_t \in \mathbb{R}^{r_{t-1}\times r_p}, \tag{11}$$

where r_{t-1} and r_p are the column dimensions of $\mathbf{Z}_{t-1}$ and $\hat{\mathbf{U}}_t$, respectively. Each element of $\mathbf{M}_t$ reflects the cosine alignment between directions from the established null space and those of the current principal space.

Since perfect alignment between these subspaces is rare, we extract highly aligned directions by applying SVD to the alignment matrix: $\mathbf{M}_t = \mathbf{U}_M\mathbf{\Sigma}_M\mathbf{V}_M^\top$, and retaining the singular vectors with singular values above a dynamic threshold τ_2. Given the variability of alignment distributions across layers and editing batches, we detect the elbow point via second-order differences of the ordered singular values λ_i in $\mathbf{\Sigma}_M$, selecting the index set $\mathcal{I} = \{i \mid \lambda_i > \tau_2(\mathbf{\Sigma}_M)\}$, This strategy adaptively selects aligned components across varying scenarios. The corresponding intersection basis is constructed as:

$$\mathbf{Z}_{\text{int}}^{(t)} = \mathbf{Z}_{t-1}\hat{\mathbf{U}}_M^{(t)} \in \mathbb{R}^{d\times|\mathcal{I}|}, \tag{12}$$

where $\hat{\mathbf{U}}_M^{(t)}$ denotes the submatrix of $\mathbf{U}_M$ consisting of columns indexed by $\mathcal{I}$.

Nevertheless, the intersection basis $\mathbf{Z}_{\text{int}}^{(t)}$ may contain vectors that are nearly collinear or numerically unstable, potentially undermining the precision and stability of the projection. To ensure numerical robustness and precise orthogonal projection, we apply reduced QR decomposition for orthogonalization: $\mathbf{Z}_{\text{int}}^{(t)} = \mathbf{Q}_t\mathbf{R}_t$, where the columns of $\mathbf{Q}_t \in \mathbb{R}^{d\times|\mathcal{I}|}$ form an orthonormal basis for a robust subspace representation.

Finally, the orthogonal projector from subspace alignment is defined as:

$$\mathbf{P}_{\cap}^{(t)} = \mathbf{Q}_t\mathbf{Q}_t^\top, \tag{13}$$

a symmetric and idempotent operator that precisely confines parameter updates to the aligned and numerically stable intersection of the principal and null spaces. This guarantees accumulative constraint preservation while enabling accurate integration of new knowledge. The detailed workflow is illustrated in Figure 3 and full procedure is outlined in Algorithm 1.

4.4 Solving for $\mathbf{\Delta}$ in OrthoEdit

To sloving the update parameter $\mathbf{\Delta}$, we begin by revisiting the general objective for batched sequential editing[3], which is formulated to accommodate sequential batch editing:

$$\begin{aligned}\mathbf{\Delta} = \arg\min_{\tilde{\mathbf{\Delta}}} &\left\|(\mathbf{W}+\tilde{\mathbf{\Delta}})\mathbf{K}_t - \mathbf{V}_t\right\|^2 \\ &+ \left\|\tilde{\mathbf{\Delta}}\mathbf{K}_p\right\|^2 + \left\|\tilde{\mathbf{\Delta}}\mathbf{K}_0\right\|^2,\end{aligned} \tag{14}$$

where $\mathbf{K}_0$ encodes pre-existing knowledge, and $\mathbf{K}_p$ encodes previously edited knowledge. However, this update objective simultaneously attempts to update $\mathbf{K}_t$ and preserve $\mathbf{K}_0$ and $\mathbf{K}_p$, which can induce conflicting gradients and make optimization difficult.

[3]When the objective of MEMIT is extended to the batched sequential editing setting (by introducing $\mathbf{K}_p$), we obtain this generalized formulation.

	Method	Score ↑	Efficacy↑	Generalization↑	Locality↑	Consistency↑	Fluency↑
LLaMA3-8B	Unedited	11.77	$7.13_{\pm 0.26}$	$9.66_{\pm 0.25}$	$89.66_{\pm 0.19}$	$635.58_{\pm 0.11}$	$24.31_{\pm 0.08}$
	FT	66.09	$96.40_{\pm 0.19}$	$89.20_{\pm 0.26}$	$42.00_{\pm 0.35}$	$258.59_{\pm 0.43}$	$14.20_{\pm 0.09}$
	MEMIT	61.47	$71.50_{\pm 0.45}$	$67.94_{\pm 0.40}$	$49.76_{\pm 0.35}$	$506.92_{\pm 1.42}$	$7.61_{\pm 0.11}$
	PRUNE	60.45	$72.38_{\pm 0.45}$	$66.56_{\pm 0.40}$	$48.10_{\pm 0.33}$	$573.16_{\pm 0.44}$	$5.83_{\pm 0.08}$
	RECT	64.97	$72.00_{\pm 0.45}$	$70.62_{\pm 0.41}$	$55.18_{\pm 0.35}$	$525.04_{\pm 0.68}$	$13.28_{\pm 0.11}$
	AlphaEdit	80.43	$96.75_{\pm 0.18}$	$92.10_{\pm 0.22}$	$62.09_{\pm 0.31}$	$595.98_{\pm 0.29}$	$32.71_{\pm 0.12}$
	OrthoEdit	**85.96**	**99.13**$_{\pm 0.09}$	**93.42**$_{\pm 0.21}$	**70.89**$_{\pm 0.27}$	**620.93**$_{\pm 0.18}$	**33.75**$_{\pm 0.12}$
Qwen2.5-7B	Unedited	19.32	$12.82_{\pm 0.33}$	$15.23_{\pm 0.31}$	$85.96_{\pm 0.22}$	$625.46_{\pm 0.10}$	$25.58_{\pm 0.09}$
	FT	61.50	$88.75_{\pm 0.32}$	$66.77_{\pm 0.40}$	$44.38_{\pm 0.35}$	$234.20_{\pm 0.20}$	$5.99_{\pm 0.05}$
	MEMIT	76.92	$89.13_{\pm 0.31}$	$76.14_{\pm 0.35}$	$68.27_{\pm 0.28}$	$611.80_{\pm 0.54}$	$31.38_{\pm 0.11}$
	PRUNE	60.09	$66.17_{\pm 0.47}$	$62.44_{\pm 0.40}$	$53.19_{\pm 0.30}$	$573.13_{\pm 0.44}$	$9.20_{\pm 0.07}$
	RECT	66.42	$71.22_{\pm 0.45}$	$67.43_{\pm 0.40}$	$61.37_{\pm 0.32}$	$496.18_{\pm 1.45}$	$14.4_{\pm 0.15}$
	AlphaEdit	83.66	$97.57_{\pm 0.15}$	$84.98_{\pm 0.29}$	$72.24_{\pm 0.27}$	$624.15_{\pm 0.10}$	$32.12_{\pm 0.11}$
	OrthoEdit	**88.99**	**99.30**$_{\pm 0.08}$	**91.57**$_{\pm 0.23}$	**78.61**$_{\pm 0.25}$	**624.21**$_{\pm 0.10}$	**33.20**$_{\pm 0.10}$
InternLM-7B	Unedited	23.17	$16.00_{\pm 0.37}$	$18.22_{\pm 0.34}$	$82.55_{\pm 0.27}$	$553.35_{\pm 0.25}$	$33.10_{\pm 0.09}$
	FT	63.04	$92.73_{\pm 0.26}$	$70.18_{\pm 0.39}$	$44.34_{\pm 0.35}$	$489.65_{\pm 0.50}$	$28.13_{\pm 0.10}$
	MEMIT	84.14	$93.13_{\pm 0.25}$	$89.62_{\pm 0.27}$	$72.68_{\pm 0.29}$	$553.03_{\pm 0.25}$	**33.97**$_{\pm 0.09}$
	PRUNE	84.29	$93.48_{\pm 0.25}$	$90.15_{\pm 0.27}$	$72.45_{\pm 0.29}$	$552.20_{\pm 0.25}$	$33.80_{\pm 0.09}$
	RECT	84.94	$92.33_{\pm 0.27}$	$88.45_{\pm 0.28}$	$75.86_{\pm 0.28}$	$551.21_{\pm 0.25}$	$33.77_{\pm 0.09}$
	AlphaEdit	85.86	$95.38_{\pm 0.21}$	$92.48_{\pm 0.24}$	$73.30_{\pm 0.29}$	$551.07_{\pm 0.26}$	$33.57_{\pm 0.09}$
	OrthoEdit	**88.41**	**97.82**$_{\pm 0.15}$	**93.82**$_{\pm 0.20}$	**76.62**$_{\pm 0.28}$	**553.64**$_{\pm 0.25}$	$33.93_{\pm 0.09}$

Table 1: Performance on the CounterFact dataset across different models under the setting of 6,000 edits (batch size 300). Best results are in bold.

AlphaEdit (Fang et al., 2024) partially addresses this challenge by introducing a null-space projector $\mathbf{P}_0 = \mathbf{Z}_0\mathbf{Z}_0^\top$, ensuring that updates remain independent to $\mathbf{K}_0$. Defining $\mathbf{\Delta}_{\text{Alpha}} = \mathbf{\Delta}\mathbf{P}_0$, since we have $\mathbf{\Delta}\mathbf{P}_0\mathbf{K}_0 = \mathbf{0}$ by construction, the update objective simplifies to:

$$\mathbf{\Delta}_{\text{Alpha}} = \arg\min_{\tilde{\mathbf{\Delta}}} \left\|(\mathbf{W} + \tilde{\mathbf{\Delta}}\mathbf{P}_0)\mathbf{K}_t - \mathbf{V}_t\right\|^2 + \left\|\tilde{\mathbf{\Delta}}\mathbf{P}_0\mathbf{K}_p\right\|^2 + \left\|\tilde{\mathbf{\Delta}}\mathbf{P}_0\right\|^2, \quad (15)$$

where the term $||\mathbf{\Delta}\mathbf{P}_0||^2$ acts as a regularizer to ensure stable convergence. This reformulation avoids explicitly preserving pre-existing knowledge $\mathbf{K}_0$, as parameter updates occur exclusively within its null space. Nevertheless, it still enforces explicitly preservation of previously edited knowledge $\mathbf{K}_p$, retaining issue similar interference challenges in subsequent editing.

To further streamline the optimization and remove the explicit preservation of prior edits encoded in $\mathbf{K}_p$, we incorporate our Progressive Null Space Projection into the update scheme. As discussed before, our projector $\mathbf{P}_{\text{null}}$ satisfies both $\mathbf{P}_{\text{null}}\mathbf{K}_0 = \mathbf{0}$ and $\mathbf{P}_{\text{null}}\mathbf{K}_p = \mathbf{0}$. The final orthogonal projector $\mathbf{P}_\cap$ derived from $\mathbf{P}_{\text{null}}$ inherits these crucial properties. A formal proof is provided in the Appendix E.

Defining the **OrthoEdit** update as $\mathbf{\Delta}_{\text{Ortho}} = \mathbf{\Delta}\mathbf{P}_\cap$, the update objective elegantly simplifies to:

$$\mathbf{\Delta}_{\text{Ortho}} = \arg\min_{\tilde{\mathbf{\Delta}}} \left\|(\mathbf{W} + \tilde{\mathbf{\Delta}}\mathbf{P}_\cap)\mathbf{K}_t - \mathbf{V}_t\right\|^2 + \left\|\tilde{\mathbf{\Delta}}\mathbf{P}_\cap\right\|^2. \quad (16)$$

This allows the updating process to focus solely on accurately incorporating the current target knowledge $\mathbf{K}_t$, without concern for interfering with either pre-existing or previously edited knowledge.

To facilitate a concise and consistent presentation with prior work (e.g., Meng et al. (2022b) and Fang et al. (2024)), we denote the residual vector as $\mathbf{R} = \mathbf{V}_t - \mathbf{W}\mathbf{K}_t$. Substituting this into the objective in Eqn. 16 and differentiating to derive the normal equation yields:

$$(\mathbf{\Delta}\mathbf{P}_\cap\mathbf{K}_t - \mathbf{R})\mathbf{K}_t^\top\mathbf{P}_\cap + \mathbf{\Delta}\mathbf{P}_\cap = 0 \quad (17)$$

Solving this normal equation we obtain the following closed-form solution for OrthoEdit:

$$\mathbf{\Delta}_{\text{Ortho}} = \mathbf{R}\mathbf{K}_t^\top\mathbf{P}_\cap\left(\mathbf{K}_t\mathbf{K}_t^\top + \mathbf{I}\right)^{-1}. \quad (18)$$

Method	Pref. Dist.			Rel. Spec.			Subj. Spec.		
	EOS↑	CAP↑	DP↓	EOS↑	CAP↑	DP↓	EOS↑	CAP↑	DP↓
MEMIT	44.82	15.80	17.33	66.03	11.28	5.85	60.48	15.41	10.60
RECT	54.40	22.37	14.50	76.03	16.34	4.70	64.43	21.71	**5.11**
PRUNE	44.63	16.05	18.03	66.10	11.32	6.28	62.40	16.79	10.27
AlphaEdit	50.19	19.20	16.59	74.00	15.46	6.25	61.14	19.94	8.80
OrthoEdit	**58.89**	**23.14**	**11.93**	**81.14**	**17.51**	**4.18**	**65.94**	**22.22**	7.00

Table 2: Performance comparison on EVOKE benchmark using LLaMA3-8B model. **Pref. Dist.** denote Prefix distraction, **Rel. Spec.** denote Subject Specificity, **Subj. Spec.** is Relation Specificity.

A detailed derivation is provided in the Appendix G. This final expression provides a concise closed-form solution for the update parameter. It leverages the orthogonality properties of the projector $\mathbf{P}_{\cap}$, ensuring that the current knowledge updates are fully independent of both pre-existing knowledge and previously edited knowledge.

5 Experiments

5.1 Baselines and Models

We compare OrthoEdit with existing methods that support batched sequential knowledge editing, including fine-tuning (FT) (Rawat et al., 2021), MEMIT (Meng et al., 2022b), RECT (Gu et al., 2024b), PRUNE (Ma et al., 2024), and AlphaEdit (Fang et al., 2024). Experiments are conducted on multiple LLMs, including LLaMA3-8B, Qwen2.5-7B, and InternLM-7B. We provide the hyperparameter settings in Appendix B.

5.2 Datasets and Metrics

We evaluate editing methods on three knowledge editing benchmarks: the widely used CounterFact (Meng et al., 2022a) and ZsRE (Levy et al., 2017) datasets, and the recently introduced QAEdit (Yang et al., 2025), EVOKE (Zhang et al., 2025) datasets. To assess general capability of the post-edited model, we further evaluate the performance on six downstream tasks from the GLUE benchmark (Wang et al., 2018).

For the knowledge editing datasets, we consider three core metrics: **Efficacy** (or Reliability), measuring the success rate of edits; **Generalization** (or Paraphrase), evaluating success on paraphrased prompts; and **Locality** (or Specificity), quantifying the preservation of original knowledge. Formal definitions are in Appendix A. For CounterFact, we additionally report the harmonic mean of these metrics as **Score**, along with **Consistency** (reference score) and **Fluency** (generation entropy). For QAEdit, beyond standard metrics, we include LLM-judged **real-world** QA accuracy across three question types (via GPT-4o-mini). The GLUE benchmark includes six downstream tasks: SST (Socher et al., 2013), MMLU (Hendrycks et al., 2021), MRPC (Dolan and Brockett, 2005), CoLA (Warstadt et al., 2019), RTE (Bentivogli et al., 2009), and NLI (Williams et al., 2018). For each task, we report the **F1 Score** of the edited models.

6 Findings and Results

We conduct comprehensive experiments spanning multiple datasets, models, and editing settings to systematically evaluate the effectiveness and robustness of OrthoEdit and provide detailed analyses and insights into its performance.

6.1 Improved Editing Performance and Strong Locality

OrthoEdit's joint null-space construction and exact intersection projection guarantee that each update is confined to interference-free directions, preserving untouched facts. To validate this, we evaluate on the classical CounterFact and ZsRE benchmarks. In both cases, OrthoEdit achieves substantially higher edit success and locality scores than prior methods, confirming our theoretical expectations.

We first evaluate OrthoEdit and baselines on CounterFact using LLaMA3-8B, Qwen-2.5-7B, and InternLM-7B models. As shown in Table 1, OrthoEdit consistently outperforms all baselines, including the previous state-of-the-art AlphaEdit, with significant gains across all metrics. These results highlight the effectiveness of orthogonal subspace projection in isolating target updates from both pre-existing and prior edited knowledge, preserving earlier information and simplifying the up-

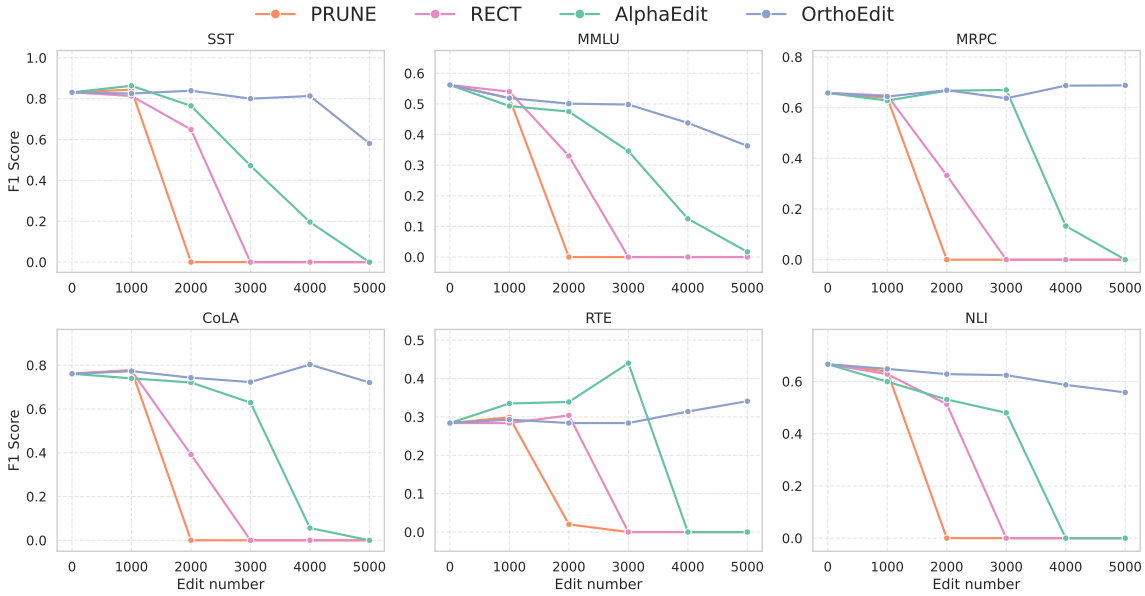

Figure 4: General capability across six tasks with increasing number of edits on the LLaMA3-8B model.

Method	LLaMA3-8B			Qwen2.5-7B		
	Efficacy	General.	Locality	Efficacy	General.	Locality
-w/o PNS & PRI	$96.75_{\pm 0.18}$	$91.10_{\pm 0.22}$	$62.09_{\pm 0.31}$	$97.57_{\pm 0.15}$	$84.98_{\pm 0.29}$	$72.24_{\pm 0.27}$
-w/o PRI	$98.87_{\pm 0.11}$	$93.32_{\pm 0.21}$	$68.28_{\pm 0.28}$	$97.85_{\pm 0.15}$	$85.78_{\pm 0.29}$	$75.00_{\pm 0.26}$
OrthoEdit	$99.13_{\pm 0.09}$	$93.42_{\pm 0.21}$	$70.89_{\pm 0.27}$	$99.30_{\pm 0.08}$	$91.57_{\pm 0.23}$	$78.61_{\pm 0.25}$

Table 3: Ablation Study on CounterFact. PNS denote the progressive null space projection, and PRI is the principal space projection. The full version used the orthigonalised intersection space of them.

date objective for each new batch of edit.

Table 5 further shows robustness under more challenging settings on LLaMA3-8B. With 5,000 total edits and batch size 100, where updates occur more frequently, OrthoEdit maintains high and stable performance, while strong baselines like AlphaEdit degrade significantly, highlighting OrthoEdit's resilience in long-term editing.

We further evaluate editing performance on the ZsRE dataset under two different editing configurations using the LLaMA3-8B model. As shown in Table 7, for 6,000 edits with a batch size of 300, MEMIT and PRUNE perform poorly across all metrics, indicating low stability. RECT also exhibits severe degradation. While AlphaEdit achieves relatively strong performance, but OrthoEdit consistently achieves the best results.

Under the more challenging setting of 5,000 edits with a batch size of 100, where parameter updates are applied more frequently, MEMIT, PRUNE, and RECT all collapse catastrophically. AlphaEdit remains relatively stable but still shows a noticeable decline. In contrast, OrthoEdit maintains consistently high performance without signs of degradation, further validating its robustness and reliability in extended sequential editing.

6.2 Real-World Robustness and Less Overfitting

OrthoEdit generalizes well to more complex and realistic editing settings, as evidenced by its consistently strong performance on both QAEdit and EVOKE under extensive sequential edits.

We further evaluate OrthoEdit on QAEdit, a recent benchmark built from multiple real-world QA datasets. Beyond standard editing metrics, QAEdit includes LLM-based judgment to assess factual alignment, posing a more realistic and challenging editing scenario. Figures 5 show results under both **Standard** and **Real-World** evaluation. Interestingly, AlphaEdit suffers a severe performance collapse, largely due to its cumulative and numerically additive retention of prior edits $\mathbf{K}_p$, which causes escalating interference, particularly pronounced on QAEdit where real-world knowledge more strongly perturbs model param-

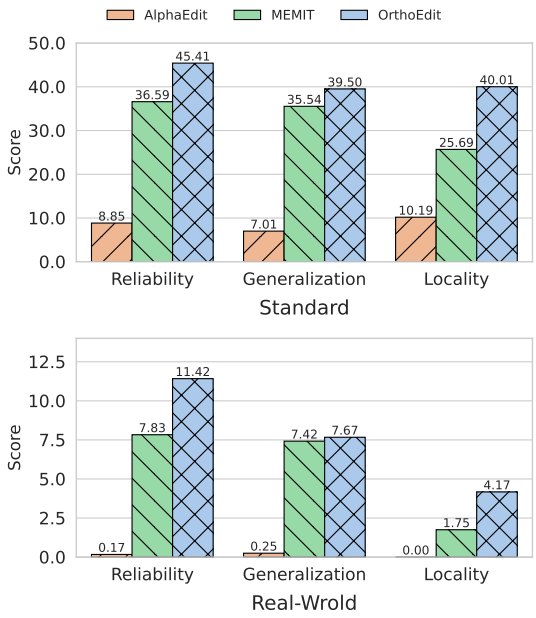

Figure 5: Performance on QAEdit dataset.

eters. In contrast, although MEMIT lacks any mechanism to retain $\mathbf{K}_p$, it avoids such collapse. OrthoEdit avoided this interference by explicitly eliminating $\mathbf{K}_p$ from the update objective via progressive null space projection. Crucially, OrthoEdit consistently outperforms both baselines, demonstrating superior stability and accuracy under realistic editing demands.

To assess overfitting in model editing, we additionally evaluated OrthoEdit on the recently proposed EVOKE (Zhang et al., 2025) benchmark, which is explicitly designed to probe overfitting in knowledge editing. EVOKE includes challenging settings such as *Prefix Distraction*, *Subject Specificity* and *Relation Specificity*, the detailed description is provided in Appendix C. As shown in the results in Table 2, OrthoEdit continues to achieve the best overall performance and exhibiting less overfitting according to the overfitting-specific evaluations. These results suggest that OrthoEdit effectively alleviates the overfitting issue and generalizes beyond the training edit instances.

Finally, to provide an intuitive view of editing stability, we visualize the distribution of hidden representations before and after editing, which can reveal both internal drift and potential overfitting. Specifically, we randomly sample 1,200 CounterFact examples, extract their hidden representations from LLaMA3-8B, and visualize them using t-SNE (van der Maaten and Hinton, 2008) with scatter plots with covariance ellipses. Figure 6 and Figure 7 in Appendix H show results after 5,000 edits (batch size 100) and 6,000 edits (batch size 300), respectively. comparing MEMIT, AlphaEdit, and OrthoEdit. MEMIT induces a substantial shift in the representation space, indicating severe internal drift and overfitting, whereas AlphaEdit shows a smaller yet still noticeable deviation. In contrast, OrthoEdit maintains a distribution closely aligned with the unedited model, suggesting improved stability and reduced overfitting.

These results offer empirical evidence for OrthoEdit's locality and robustness, explaining its ability to preserve general capabilities across extended editing.

6.3 Long-Term Stability and General Capability

By constraining updates to orthogonal subspaces, OrthoEdit theoretically prevents cumulative drift in model parameters. We verify this on the six GLUE tasks after thousands of sequential edits, showing strong stability in F1 scores while competing methods fail. Excessive parameter updates can lead to catastrophic forgetting (Gupta et al., 2024; Huang et al.), severely impairing a model's general capability on downstream tasks. To assess the stability of knowledge editing, we evaluate post-editing performance on the GLUE benchmark, which spans six representative NLP tasks.

Figure 4 presents the F1 scores after up to 5,000 edits with a batch size of 100. PRUNE and RECT exhibit rapid degradation, with general capability collapsing by 2,000 and 3,000 edits, respectively. AlphaEdit maintains moderate performance until 3,000–4,000 edits but ultimately fails at 5,000 edits. In contrast, OrthoEdit preserves stable general performance throughout, with only marginal declines on SST and MMLU, and near-original performance on the remaining four tasks. Figure 8 in Appendix 8 further corroborates these findings under a 6,000-edit setting with batch size of 300. OrthoEdit consistently maintains the highest general capability, underscoring its robustness in extended sequential editing.

Notably, the gap between AlphaEdit and OrthoEdit is consistent with their different treatments of accumulated prior edits. AlphaEdit isolates updates from pre-existing knowledge, but previously edited knowledge $\mathbf{K}_p$ is still preserved through an explicit objective term. As $\mathbf{K}_p$ grows across batches, optimization increasingly entangles the

current edit with constraints from all past edits, which can induce cumulative drift in shared FFN directions that also support downstream tasks. OrthoEdit instead eliminates $\mathbf{K}_p$ from the update objective via progressive null-space projection, so each update remains orthogonal to all prior edited keys by construction, substantially reducing cumulative drift and improving long-term stability.

6.4 Ablation Studies

To assess the contribution of each component in our framework, we conduct ablation studies on the LLaMA3-8B and Qwen2.5-7B models. Disabling both the Progressive Null Space Projection (PNS) and Principal Space Extraction (PRI) reduces our method to the AlphaEdit baseline, denoted as "$-w/o$ **PNS & PRI**". To isolate the effect of PNS, we enable only this component while disabling PRI and orthogonalization, denoted as "$-w/o$ **PRI**". Enabling all components yields the complete OrthoEdit framework.

As shown in Table 3, the PNS module alone consistently improves performance over the baseline across all metrics and models. Incorporating principal space extraction and orthogonalization further enhances results, demonstrating the cumulative benefit of each component.

To evaluate how the threshold in SVD influences the performance, we have conducted ablation study on the threshold τ_2, where we manually vary τ_2 across multiple scales, also shown in Table 8 in Appeldix M. The results reveal a clear and interpretable trade-off: Larger values of τ_2 yield a more restrictive principal subspace, which improves locality but may reduce efficacy and generalization. Smaller values of τ_2 produce a more expressive update subspace, leading to higher editing success but with increased interference. These results demonstrate that τ_2 effectively controls the balance between update capacity and stability. The automatically detected τ_2 consistently achieves balanced performance across all metrics, confirming the reliability of the dynamic selection strategy.

We also provide in-batch and cross-batch performance under increased editing frequency to shown OrthoEdit can identify and work on proper subspace, details are presented in Appendix N.

6.5 Efficiency Analysis

We evaluate the efficiency of OrthoEdit by measuring the average editing time per sample and peak GPU memory usage. While OrthoEdit involves several additional SVD operations on $\mathbf{K}_0$,$\mathbf{H}_t$,$\mathbf{M}_t$, these are performed on low-dimensional matrices (at most size b) and incur negligible overhead. Other matrix operations and parameter updates are similarly lightweight, and the reduced update space may even accelerates optimization. As shown in Table 4, OrthoEdit achieves similar memory usage to MEMIT and AlphaEdit, with slightly increased runtime than MEMIT but faster than AlphaEdit. These results indicate that OrthoEdit is both effective and computationally efficient.

	Method	AVG. Time	MEM. Cost
LLaMA	MEMIT	7.84 s	38049 MB
	AlphaEdit	8.46 s	36189 MB
	OrthoEdit	8.32 s	37254 MB
Qwen	MEMIT	3.35 s	41009 MB
	AlphaEdit	4.69 s	37911 MB
	OrthoEdit	4.64 s	38174 MB
Int.LM	MEMIT	4.51 s	36200 MB
	AlphaEdit	4.93 s	35515 MB
	OrthoEdit	4.83 s	36193 MB

Table 4: Efficiency Analysis. **AVG. Time** is the per-sample editing time, and **MEM. Cost** is the peak GPU memory usage during editing.

7 Conclusion

In this work, we introduced **OrthoEdit**, a principled framework for batched sequential knowledge editing in large language models. By explicitly enforcing orthogonality to both pre-existing and previously edited knowledge, OrthoEdit prevents interference across editing iterations and preserves the model's general capability. Our method combines progressive null space construction, principal subspace compression, and subspace orthogonalization to produce compact, well-constrained parameter updates. This design improves editing fidelity and ensures sustainability under long-term batched edits. Empirical evaluations across diverse benchmarks validate the stability and effectiveness of our approach. The ultimate editing capacity remains an open question dependent on model size and editing conditions, motivating future work on mechanisms such as periodic constraint-subspace re-initialization. OrthoEdit provides a principled foundation for future research in scalable and robust knowledge editing.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2020-II201336, Artificial Intelligence graduate school support(UNIST)) and IITP grant funded by the Korea government(MSIT) (No.RS-2023-00216011, Development of artificial complex intelligence for conceptually understanding and inferring like human). Shanbao Qiao and Xuebing Liu were also supported by China Scholarship Council (CSC).

References

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1.

Baolong Bi, Shenghua Liu, Yiwei Wang, Lingrui Mei, Hongcheng Gao, Junfeng Fang, and Xueqi Cheng. 2024. Struedit: Structured outputs enable the fast and accurate knowledge editing for large language models. *arXiv preprint arXiv:2409.10132*.

Yuchen Cai and Ding Cao. 2024. O-edit: Orthogonal subspace editing for language model sequential editing. *arXiv preprint arXiv:2410.11469*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

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

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

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

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. 630(8017):625–630.

Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2023. Bias and fairness in large language models: A survey.

Hengrui Gu, Kaixiong Zhou, Xiaotian Han, Ninghao Liu, Ruobing Wang, and Xin Wang. 2024a. 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)*, pages 8069–8083, Bangkok, Thailand. Association for Computational Linguistics.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024b. Model editing can hurt general abilities of large language models. *CoRR*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiusheng Huang, Jiaxiang Liu, Yequan Wang, and Kang Liu. Reasons and solutions for the decline in model performance after editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xiusheng Huang, Yequan Wang, Jun Zhao, and Kang Liu. 2024. Commonsense knowledge editing based on free-text in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14870–14880, Miami, Florida, USA. Association for Computational Linguistics.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. *arXiv preprint arXiv:2301.09785*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-seng Chua. 2025. Anyedit: Edit any knowledge encoded in language models. *arXiv preprint arXiv:2502.05628*.

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

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

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2023. Pmet: Precise model editing in a transformer. *arXiv preprint arXiv:2308.08742*.

Jiateng Liu, Pengfei Yu, Yuji Zhang, Sha Li, Zixuan Zhang, Ruhi Sarikaya, Kevin Small, and Heng Ji. 2024. EVEDIT: Event-based knowledge editing for deterministic knowledge propagation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 4907–4926, Miami, Florida, USA. Association for Computational Linguistics.

Yifan Lu, Yigeng Zhou, Jing Li, Yequan Wang, Xuebo Liu, Daojing He, Fangming Liu, and Min Zhang. 2024. Knowledge editing with dynamic knowledge graphs for multi-hop question answering. *arXiv preprint arXiv:2412.13782*.

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

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35.

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

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

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. 2024. Dyknow: dynamically verifying time-sensitive factual knowledge in llms. *arXiv preprint arXiv:2404.08700*.

Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity cloze by date: What LMs know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 693–702, Seattle, United States. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Siyuan Qi, Bangcheng Yang, Kailin Jiang, Xiaobo Wang, Jiaqi Li, Yifan Zhong, Yaodong Yang, and Zilong Zheng. 2024. In-context editing: Learning knowledge from self-induced distributions.

Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. 2024a. COMEM: In-context retrieval-augmented mass-editing memory in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2333–2347, Mexico City, Mexico. Association for Computational Linguistics.

Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. 2024b. DistillMIKE: Editing distillation of massive in-context knowledge editing in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7639–7654, Bangkok, Thailand. Association for Computational Linguistics.

Shanbao Qiao, Xuebing Liu, and Seung-Hoon Na. 2025. Wasserstein distance constraint and parameter sparsification for batched and iterative knowledge editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25019–25028.

Ankit Singh Rawat, Chen Zhu, Daliang Li, Felix Yu, Manzil Zaheer, Sanjiv Kumar, and Srinadh Bhojanapalli. 2021. Modifying memories in transformer models. In *International Conference on Machine Learning (ICML)*, volume 2020.

Domenic Rosati, Robie Gonzales, Jinkun Chen, Xuemin Yu, Yahya Kayani, Frank Rudzicz, and Hassan Sajjad. 2024. Long-form evaluation of model editing. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3749–3780, Mexico City, Mexico. Association for Computational Linguistics.

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

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language model via meta learning. In *The Twelfth International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Renzhi Wang and Piji Li. 2024. Memoe: Enhancing model editing with mixture of experts adaptors. *arXiv preprint arXiv:2405.19086*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Ziwen Xu, Shuxun Wang, Kewei Xu, Haoming Xu, Mengru Wang, Xinle Deng, Yunzhi Yao,

Guozhou Zheng, Huajun Chen, and Ningyu Zhang. 2025. Easyedit2: An easy-to-use steering framework for editing large language models.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. 2025. The mirage of model editing: Revisiting evaluation in the wild.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *CoRR*, abs/2305.13172.

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

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Knowledge graph enhanced large language model editing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22647–22662, Miami, Florida, USA. Association for Computational Linguistics.

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

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024b. Instructedit: instruction-based knowledge editing for large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, Siyuan Cheng, Ziwen Xu, Xin Xu, Jia-Chen Gu, Yong Jiang, Pengjun Xie, Fei Huang, Lei Liang, Zhiqiang Zhang, Xiaowei Zhu, Jun Zhou, and Huajun Chen. 2024c. A comprehensive study of knowledge editing for large language models.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

A Definition: Details of Metrics

Given an editing request in the form of $(subject, relation, object)$, denoted as (s, r, o), the goal of knowledge editing is to modify the model such that, when provided with the input (s, r), it changes its output from the original answer o to the new answer o^*. We denote the edited model as $\mathcal{M}_{\boldsymbol{\Delta}}$, and the model probability as $\mathbb{P}$, then:

Efficacy measures the accuracy of the editing process, specifically reflecting the successful modification of the factual knowledge statement in the dataset, which can be defined as:

$$\mathbb{E}[\mathbb{P}_{\mathcal{M}_{\boldsymbol{\Delta}}}(o^*|(s,r)) > \mathbb{P}_{\mathcal{M}_{\boldsymbol{\Delta}}}(o|(s,r))]. \quad (19)$$

Generalization evaluates whether the edit can be effectively applied to paraphrased or contextually related sentences within the dataset. Note the paraphrased query as $p(s, r)$, the formal definition can be expressed as:

$$\mathbb{E}[\mathbb{P}_{\mathcal{M}_{\boldsymbol{\Delta}}}(o^*|p(s,r)) > \mathbb{P}_{\mathcal{M}_{\boldsymbol{\Delta}}}(o|p(s,r))]. \quad (20)$$

Locality refers to the preservation of original knowledge that unrelated to editing requests, ensuring it remains intact. This is evaluated using irrelevant natural questions or neighborhood questions in the dataset. Note the unrelated neighborhood question as $n(s,r)$, the formal definition is:

$$\mathbb{E}[\mathbb{P}_{\mathcal{M}_{\mathbf{\Delta}}}(o^*|n(s,r)) < \mathbb{P}_{\mathcal{M}_{\mathbf{\Delta}}}(o|n(s,r))]. \quad (21)$$

Consistency measures the semantic consistency of $\mathcal{M}(\mathbf{\Delta})$'s generations, which is evaluated by giving the model $\mathcal{M}_{\mathbf{\Delta}}$ a subject s and computing the cosine similarity between the TF-IDF vectors of the model-generated text and a reference Wikipedia text about o.

Fluency measuring the excessive repetition in model outputs, it uses the weighted average of bi- and tri-gram entropies, which can be defined as:

$$-\frac{2}{3}\sum_k g_2(k)\log_2 g_2(k)+\frac{4}{3}\sum_k g_3(k)\log_2 g_3(k), \quad (22)$$

where $g_n(\cdot)$ is the n-gram frequency distribution, k denotes an individual n-gram.

Paraphrase shares the same definition as "Generalization" in the CounterFact dataset, we adopt this term in the ZsRE benchmark to maintain consistency with previous work.

Specificity shares the same definition as "Locality" in the CounterFact dataset; we adopt this term in the ZsRE benchmark to maintain consistency with previous work.

Reliability is to evaluate whether the post-edited model can provide the target answer for a given context, i.e. $\mathbb{E}[\mathcal{M}_{\mathbf{\Delta}}(s,r) = o^*]$.

Editing Overfit Score (EOS) evaluates the performance of the edited model on complex questions where the correct answer is not o^*. It serves as a primary indicator of the model's overfitting and overall performance. The score is calculated as the proportion of cases where the model overfitting by favoring the edit target o^* over the correct answer o_c, formalized as:

$$\mathbb{E}[\mathbb{I}[\mathbb{P}_{\mathcal{M}_{\mathbf{\Delta}}}(o_c|(s,r)) > \mathbb{P}_{\mathcal{M}_{\mathbf{\Delta}}}(o^*|(s,r))]]. \quad (23)$$

Correct Answer Probability (CAP) denotes the probability that the model generates the correct answer o_c for a given prompt (s,r), formalized as: $\mathbb{P}(o_c|(s,r))$.

Direct Probability (DP) denotes the likelihood that the model produces the edit target o^*, expressed as $\mathbb{P}(o^*|(s,r))$.

B Hyperparameter Settings

We set the fixed threshold τ_1 to 0.02, following the configuration used in AlphaEdit (Fang et al., 2024). For LLaMA3-8B models, edits are applied to layers {4, 5, 6, 7, 8}. The hyperparameter λ is set to 15,000. Each edit is optimized for 25 steps with a learning rate of 0.1 and a weight decay of 1×10^{-3}.

All experiments are conducted on RTX A6000 (48GB) GPUs. The LLMs are loaded using HuggingFace Transformers.

C Details of EVOKE Benchmark

EVOKE benchmark includes a variety of challenging and realistic tasks. Specifically, it contains the following tasks:

Prefix Distraction which evaluates whether an edited model can maintain its ability to provide correct answers to unrelated questions when the newly introduced knowledge is used as a prefix. It examines the model's capacity to preserve factual accuracy and resist interference from the edited information. This evaluation also helps determine whether the model has overfitted to the newly learned association between the subject, relation, and object, providing a more explicit measure of overfitting than multi-hop reasoning.

Subject Specificity which evaluates the edited model using questions that share the same subject as the edited sample but involve different relations. These questions concern the same subject but require different correct answers from the newly introduced target. This setting is used to assess whether the model has overfitted to the association between the subject and the newly learned object.

Relation Specificity which evaluates the edited model using questions that involve different subjects from the edited sample while maintaining the same relation. These questions focus on the same relational context but require different correct answers from the newly introduced target. This evaluation is designed to determine whether the model has overfitted to the association between the relation and the newly learned object.

D Proof: Property of Progressive Null Space Projector $\mathbf{P}_{\text{null}}$

Let $\mathbf{K}_0 \in \mathbb{R}^{d\times d}$ denote the pre-existing knowledge matrix, and let $\{\mathbf{K}_i\}_{i=1}^{t-1}$ be the sequence of previously edited knowledge keys, each $\mathbf{K}_i \in \mathbb{R}^{d\times b}$. Our goal is to construct a projection matrix $\mathbf{P}_{\text{null}}^{(t)}$ such that:

$$\mathbf{\Delta}\mathbf{P}_{\text{null}}^{(t)}\mathbf{K}_i = \mathbf{0}, \quad \text{for all } i < t. \tag{24}$$

Step 1: Initial Null Space Basis from $\mathbf{K}_0$ We compute the SVD of the uncentered covariance matrix: $\mathbf{K}_0 = \mathbf{U}_0\mathbf{\Sigma}_0\mathbf{U}_0^\top$. Let $\mathbf{Z}_0 \in \mathbb{R}^{d\times r_0}$ be the submatrix of $\mathbf{U}_0$ corresponding to zero singular values. Then:

$$\mathbf{Z}_0^\top\mathbf{K}_0 = \mathbf{0}, \quad \text{and} \quad \mathbf{P}_{\text{null}}^{(0)} = \mathbf{Z}_0\mathbf{Z}_0^\top. \tag{25}$$

Step 2: Progressive Construction of $\mathbf{Z}_{t-1}$ For each edited knowledge key $\mathbf{K}_i$ with $i = 1, \ldots, t-1$, we incrementally refine the null space basis.

Given the null space basis from the previous step $\mathbf{Z}_{i-1} \in \mathbb{R}^{d\times r_{i-1}}$, we define the interference matrix:

$$\mathbf{H}_i = \mathbf{Z}_{i-1}^\top\mathbf{K}_i \in \mathbb{R}^{r_{i-1}\times b}. \tag{26}$$

We perform SVD on $\mathbf{H}_i$: $\mathbf{H}_i = \mathbf{U}_i\mathbf{\Sigma}_i\mathbf{V}_i^\top$, and identify the left null space basis of $\mathbf{H}_i$ by selecting columns of $\mathbf{U}_i$ corresponding to zero singular values. Let $\hat{\mathbf{U}}_i \in \mathbb{R}^{r_{i-1}\times r_i}$ denote this submatrix. We then update the null space basis:

$$\mathbf{Z}_i = \mathbf{Z}_{i-1}\hat{\mathbf{U}}_i \in \mathbb{R}^{d\times r_i}. \tag{27}$$

Since the matrix $\hat{\mathbf{U}}_i \in \mathbb{R}^{r_{i-1}\times r_i}$ consists of the left-singular vectors of $\mathbf{H}_i$ corresponding to zero singular values. This implies:

$$\mathbf{H}_i^\top\hat{\mathbf{U}}_i = \mathbf{0} \quad \Leftrightarrow \quad \hat{\mathbf{U}}_i^\top\mathbf{H}_i = \mathbf{0}. \tag{28}$$

By construction, $\mathbf{H}_i = \mathbf{Z}_{i-1}^\top\mathbf{K}_i$ and $\mathbf{Z}_{i-1}\hat{\mathbf{U}}_i = \mathbf{Z}_i$, we have:

$$\begin{aligned}\mathbf{Z}_i^\top\mathbf{K}_i &= (\mathbf{Z}_{i-1}\hat{\mathbf{U}}_i)^\top\mathbf{K}_i \\ &= \hat{\mathbf{U}}_i^\top(\mathbf{Z}_{i-1}^\top\mathbf{K}_i) = \hat{\mathbf{U}}_i^\top\mathbf{H}_i = \mathbf{0}.\end{aligned} \tag{29}$$

Furthermore, since $\mathbf{Z}_{i-1}^\top\mathbf{K}_j = \mathbf{0}$ for all $j < i$ by the induction hypothesis, we have:

$$\begin{aligned}\mathbf{Z}_i^\top\mathbf{K}_j &= \hat{\mathbf{U}}_i^\top(\mathbf{Z}_{i-1}^\top\mathbf{K}_j) \\ &= \hat{\mathbf{U}}_i^\top \cdot \mathbf{0} = \mathbf{0}, \quad \forall j < i.\end{aligned} \tag{30}$$

Hence, we conclude:

$$\mathbf{Z}_i^\top\mathbf{K}_j = \mathbf{0}, \quad \forall j \le i. \tag{31}$$

Step 3: Final Projector After $t-1$ steps, the basis $\mathbf{Z}_{t-1}$ satisfies:

$$\mathbf{Z}_{t-1}^\top\mathbf{K}_i = \mathbf{0}, \quad \forall i < t. \tag{32}$$

We define the orthogonal projector:

$$\mathbf{P}_{\text{null}}^{(t)} = \mathbf{Z}_{t-1}\mathbf{Z}_{t-1}^\top. \tag{33}$$

Thus, for any update matrix $\mathbf{\Delta}$ projected as $\mathbf{\Delta}\mathbf{P}_{\text{null}}^{(t)}$, we have:

$$\mathbf{\Delta}\mathbf{P}_{\text{null}}^{(t)}\mathbf{K}_i = \mathbf{0}, \quad \forall i < t. \tag{34}$$

Conclusion This recursive construction of $\mathbf{P}_{\text{null}}$ guarantees that $\mathbf{\Delta}$ lies in the joint null space of all previously edited knowledge keys, satisfies all desired constraints.

E Proof: Orthogonality and Null-space Preservation of $\mathbf{P}_\cap$

We formally prove that the orthogonal projector $\mathbf{P}_\cap$, derived from the progressive null-space projector $\mathbf{P}_{\text{null}}$, inherits the crucial null-space properties:

$$\mathbf{P}_\cap\mathbf{K}_0 = \mathbf{0}, \quad \mathbf{P}_\cap\mathbf{K}_p = \mathbf{0}. \tag{35}$$

Step 1: Properties of Progressive Null-space Projector $\mathbf{P}_{\text{null}}$ Recall that the progressive null-space projector is constructed as:

$$\mathbf{P}_{\text{null}} = \mathbf{Z}_{t-1}\mathbf{Z}_{t-1}^\top, \tag{36}$$

where columns of $\mathbf{Z}_{t-1}$ form an orthonormal basis satisfying:

$$\mathbf{Z}_{t-1}^\top\mathbf{K}_0 = \mathbf{0}, \quad \mathbf{Z}_{t-1}^\top\mathbf{K}_p = \mathbf{0}. \tag{37}$$

Therefore, by construction:

$$\mathbf{P}_{\text{null}}\mathbf{K}_0 = \mathbf{0}, \quad \mathbf{P}_{\text{null}}\mathbf{K}_p = \mathbf{0}. \tag{38}$$

Step 2: Definition of $\mathbf{P}_\cap$ from Subspace Alignment The final orthogonal projector is defined as:

$$\mathbf{P}_\cap = \mathbf{Q}_t\mathbf{Q}_t^\top, \tag{39}$$

where the orthonormal columns of $\mathbf{Q}_t$ come from the intersection basis:

$$\mathbf{Z}_{\text{int}}^{(t)} = \mathbf{Z}_{t-1}\hat{\mathbf{U}}_M^{(t)}, \tag{40}$$

specifically, obtained via QR decomposition:

$$\mathbf{Z}_{\text{int}}^{(t)} = \mathbf{Q}_t\mathbf{R}_t. \tag{41}$$

Thus, columns of $\mathbf{Q}_t$ are linear combinations of columns of $\mathbf{Z}_{t-1}$. Formally, there exists a matrix $\mathbf{A}$ such that:

$$\mathbf{Q}_t = \mathbf{Z}_{t-1}\mathbf{A}, \quad \text{with } \mathbf{A} = \hat{\mathbf{U}}_M^{(t)}\mathbf{R}_t^{-1}. \tag{42}$$

Step 3: Preservation of Null-space Properties by $\mathbf{P}_\cap$ Since each column of $\mathbf{Q}_t$ is a linear combination of columns from $\mathbf{Z}_{t-1}$, it follows that:

$$\begin{aligned}\mathbf{Q}_t^\top \mathbf{K}_0 &= \mathbf{A}^\top \mathbf{Z}_{t-1}^\top \mathbf{K}_0 = \mathbf{0},\\ \mathbf{Q}_t^\top \mathbf{K}_p &= \mathbf{A}^\top \mathbf{Z}_{t-1}^\top \mathbf{K}_p = \mathbf{0}.\end{aligned} \tag{43}$$

Consequently, we have:

$$\begin{aligned}\mathbf{P}_\cap \mathbf{K}_0 &= \mathbf{Q}_t \mathbf{Q}_t^\top \mathbf{K}_0 = \mathbf{0},\\ \mathbf{P}_\cap \mathbf{K}_p &= \mathbf{Q}_t \mathbf{Q}_t^\top \mathbf{K}_p = \mathbf{0}.\end{aligned} \tag{44}$$

Step 4: Orthogonality and Idempotency of $\mathbf{P}_\cap$ The columns of $\mathbf{Q}_t$ are orthonormal by QR decomposition, thus:

$$\mathbf{Q}_t^\top \mathbf{Q}_t = \mathbf{I}. \tag{45}$$

Therefore, $\mathbf{P}_\cap$ is an orthogonal projection operator satisfying:

- **Symmetry:**

$$\mathbf{P}_\cap^\top = (\mathbf{Q}_t \mathbf{Q}_t^\top)^\top = \mathbf{Q}_t \mathbf{Q}_t^\top = \mathbf{P}_\cap. \tag{46}$$

- **Idempotency:**

$$\mathbf{P}_\cap^2 = (\mathbf{Q}_t \mathbf{Q}_t^\top)(\mathbf{Q}_t \mathbf{Q}_t^\top) = \mathbf{Q}_t \mathbf{Q}_t^\top = \mathbf{P}_\cap. \tag{47}$$

Conclusion: We have rigorously demonstrated that the orthogonal projector $\mathbf{P}_\cap$ derived from the progressive null-space projector $\mathbf{P}_{\text{null}}$ preserves the required null-space properties, ensuring orthogonality to both the pre-existing knowledge $\mathbf{K}_0$ and the previously edited knowledge $\mathbf{K}_p$. This completes the proof.

F Pseudo Code: Algorithm1 Procedure of Construct $\mathbf{P}_\cap$

G Derivation: Solving $\mathbf{\Delta}_{\text{Ortho}}$

We begin by defining the objective function as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{\Delta}) = &\left\|(\mathbf{W} + \mathbf{\Delta}\mathbf{P}_\cap)\mathbf{K}_t - \mathbf{V}_t\right\|^2 \\ &+ \left\|\mathbf{\Delta}\mathbf{P}_\cap\right\|^2,\end{aligned} \tag{48}$$

and let $\mathbf{R} = \mathbf{V}_t - \mathbf{W}\mathbf{K}_t$. Substituting this into the objective yields:

$$\mathcal{L}(\mathbf{\Delta}) = \left\|\mathbf{\Delta}\mathbf{P}_\cap \mathbf{K}_t - \mathbf{R}\right\|^2 + \left\|\mathbf{\Delta}\mathbf{P}_\cap\right\|^2. \tag{49}$$

Taking the derivative of $\mathcal{L}$ with respect to $\mathbf{\Delta}$ and setting it to zero yields the following normal equation:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{\Delta}} =&(\mathbf{\Delta}\mathbf{P}_\cap \mathbf{K}_t - \mathbf{R})(\mathbf{P}_\cap \mathbf{K}_t)^\top \\ &+ \mathbf{\Delta}\mathbf{P}_\cap \mathbf{P}_\cap^\top = 0.\end{aligned} \tag{50}$$

Since $\mathbf{P}_\cap$ is an idempotent and symmetric projection matrix (i.e., $\mathbf{P}_\cap^2 = \mathbf{P}_\cap$ and $\mathbf{P}_\cap = \mathbf{P}_\cap^\top$), the expression simplifies to:

$$(\mathbf{\Delta}\mathbf{P}_\cap \mathbf{K}_t - \mathbf{R})\mathbf{K}_t^\top \mathbf{P}_\cap + \mathbf{\Delta}\mathbf{P}_\cap = 0. \tag{51}$$

Solving for $\mathbf{\Delta}$ yields the closed-form solution:

$$\mathbf{\Delta}_{\text{Ortho}} = \mathbf{R}\mathbf{K}t^\top \mathbf{P}_\cap \left(\mathbf{K}_t \mathbf{K}_t^\top + \mathbf{I}\right)^{-1}. \tag{52}$$

H Figure 6 and 7: Visualized Hidden Representation Distribution After Editing

I Figure 8: General Capability on GLUE Benchmark After 6,000 Edits

J Table 5: Performance on the CounterFact Dataset (5,000 edits with batch size 100)

K A Direct Comparison with O-Edit

Since O-Edit's implementation details and code are not publicly available, we evaluated OrthoEdit on the same dataset (CounterFact) and with the same task configurations as those reported in the O-Edit's pre-print version, enabling a meaningful comparison with the results presented therein.

As shown in Table 6, under these conditions, OrthoEdit significantly outperforms O-Edit, while O-Edit's average performance lies between that of MEMIT and AlphaEdit. Moreover, O-Edit exhibits a pronounced decline in both efficacy and generalization after only 3,000 sequential edits. Although its locality remains relatively stable, the sharp decrease in efficacy and generalization accuracy highlights its limited scalability to larger editing scenarios. This observation further suggests that gradient-penalty–based approaches tend to restrict effective parameter updates. In contrast, projection-based methods, AlphaEdit and OrthoEdit, maintain stable efficacy and generalization, with OrthoEdit achieving both stable locality and the highest average performance. These findings demonstrate that our projection-based framework not only advances beyond AlphaEdit but also surpasses the latest gradient-penalty methods reported in the literature.

Algorithm 1: ORTHOEDIT: PROCEDURE OF CONSTRUCT $\mathbf{P}_{\cap}$

Input: Timestep $t, (t > 0)$; Previous knowledge key K_{t-1}, where $K_0 \in \mathbb{R}^{d\times d}$ is symmetric, and $K_i \in \mathbb{R}^{d\times b}$ for $i \geq 1$; Target knowledge key $K_t \in \mathbb{R}^{d\times b}$; thresholds τ_1, τ_2

Output: Interference-free projection $P_{\cap}^{(t)} \in \mathbb{R}^{d\times d}$

▷ **Step 1: Progressive Null Space Projection**

$U_0\Sigma_0 U_0^\top = \text{SVD}(K_0)$ // SVD of pre-existing knowledge key

$Z_0 \leftarrow U_0[:, r_0:]$, where $r_0 \leftarrow \text{rank}_{\tau_1}(\Sigma_0)$ // Initial null space basis

for $i \leftarrow 1$ **to** $t-1$ **do**

  Construct $H_i \leftarrow Z_{i-1}^\top K_i$

  $U_H\Sigma_H V_H^\top = \text{SVD}(H_i)$

  $N_i \leftarrow U_H[:, r_i:]$, where $r_i \leftarrow \text{rank}_{\tau_1}(\Sigma_H)$

  $Z_i \leftarrow Z_{i-1}N_i$ // Update null space basis

end

$P_{\text{null}}^{(t)} \leftarrow Z_{t-1}Z_{t-1}^\top$ // Null space projector

▷ **Step 2: Principal Space Extraction**

for $i \leftarrow 1$ **to** t **do**

  $U_i\Sigma_i V_i^\top = \text{SVD}(K_i)$

  $\hat{U}_i \leftarrow U_i[:, :r_p]$, where $r_p \leftarrow \text{rank}_{\tau_1}(\Sigma_i)$ // Principal space basis

end

$P_{\text{prin}}^{(t)} \leftarrow \hat{U}_t\hat{U}_t^\top$ // Principal space projector

▷ **Step 3: Subspace Alignment and Orthogonalization**

for $i \leftarrow 1$ **to** t **do**

  $M_i \leftarrow Z_{i-1}^\top \hat{U}_i$ // Principal-angle cosine matrix

  $\mathcal{I} \leftarrow \{j : \Sigma_M^j > \tau_2\}$, where $U_M\Sigma_M V_M^\top = \text{SVD}(M_i)$ // Select indices

  $Z_{\text{int}}^{(i)} \leftarrow Z_{i-1}U_M[:, \mathcal{I}]$ // Intersection basis

  $Q_i, R_i \leftarrow \text{QR}(Z_{\text{int}}^{(i)})$ // Orthogonalise

end

$P_{\cap}^{(t)} \leftarrow Q_tQ_t^\top$ // Final projector

return $P_{\cap}^{(t)}$

L Table 7 Results on ZsRE benchmark

M Table 8 Ablation Study on Threshold τ_2

N In-batch and Cross-batch Performance Under Increased Editing Frequency

To examine whether OrthoEdit can identify and work on proper subspace. We increased the editing frequency substantially by using a batch size of 50 across 100 sequential batches, which is 2–5× higher update frequency compared to the settings in the original manuscript. We evaluated both cross-batch performance (after all edits) and in-batch retention (stability of early-edited batches while later edits are applied).

As shown in Table 9 and 10, the comparison show that under this high-frequency editing regime, even strong baselines such as AlphaEdit experience significant degradation, and both MEMIT and AlphaEdit exhibit pronounced forgetting of earlier edited knowledge across batches. In contrast, OrthoEdit maintains stable performance in both the in-batch and cross-batch evaluations. This indicates that the orthogonal subspace construction effectively prevents interference between sequential updates, allowing earlier edits to remain intact even under dense, repeated editing.

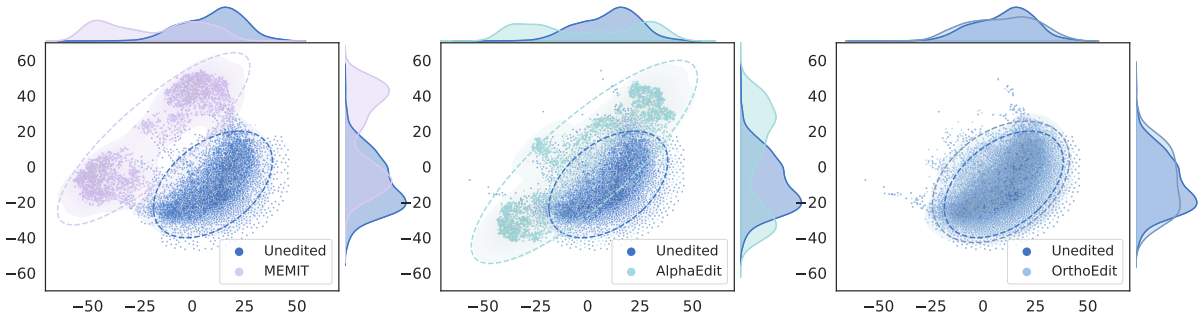

Figure 6: Visualized hidden representation of LLaMA3-8B under 5,000 edits with batch size 100.

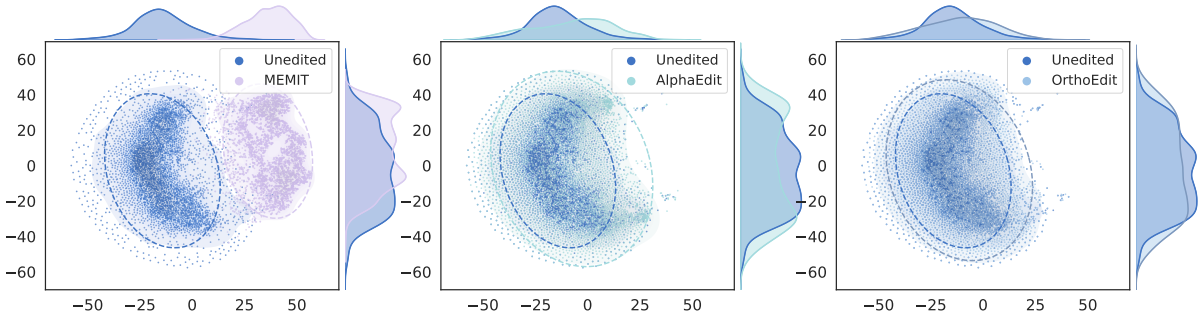

Figure 7: Visualized hidden representation of LLaMA3-8B under 6,000 edits with batch size 300.

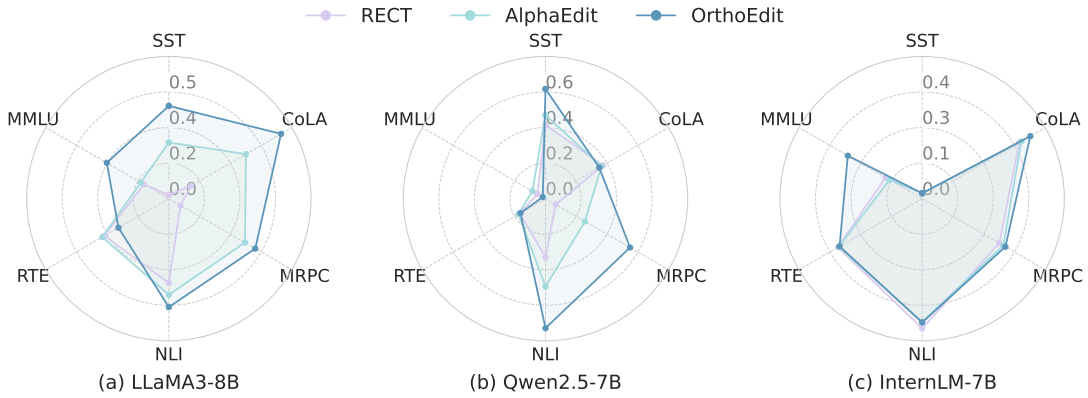

Figure 8: General capability across six tasks from the GLUE benchmark, evaluated on different models after 6,000 edits with batch size 300.

Method	Score ↑	Efficacy↑	Generalization↑	Locality↑	Consistency↑	Fluency↑
Unedited	11.64	$7.02_{\pm 0.26}$	$9.61_{\pm 0.25}$	$89.63_{\pm 0.19}$	$635.56_{\pm 0.11}$	$24.25_{\pm 0.09}$
FT	63.30	$95.38_{\pm 0.21}$	$85.79_{\pm 0.30}$	$39.60_{\pm 0.36}$	$270.72_{\pm 0.56}$	$16.84_{\pm 0.11}$
MEMIT	57.05	$66.66_{\pm 0.47}$	$56.28_{\pm 0.43}$	$50.47_{\pm 0.36}$	$517.38_{\pm 0.18}$	$3.17_{\pm 0.04}$
PRUNE	56.45	$65.70_{\pm 0.48}$	$56.72_{\pm 0.41}$	$49.28_{\pm 0.32}$	$443.66_{\pm 0.63}$	$1.83_{\pm 0.04}$
RECT	54.75	$60.52_{\pm 0.49}$	$55.51_{\pm 0.45}$	$49.37_{\pm 0.42}$	$526.15_{\pm 0.21}$	$2.19_{\pm 0.03}$
AlphaEdit	62.45	$69.90_{\pm 0.46}$	$63.55_{\pm 0.41}$	$55.57_{\pm 0.33}$	$533.09_{\pm 0.57}$	$4.34_{\pm 0.05}$
OrthoEdit	**86.91**	**98.62**$_{\pm 0.12}$	**91.01**$_{\pm 0.24}$	**74.67**$_{\pm 0.26}$	**624.76**$_{\pm 0.17}$	**34.52**$_{\pm 0.11}$

Table 5: Performance on the CounterFact dataset with LLaMA3-8B, evaluated under the setting of 5,000 edits with batch size 100. Best results are in bold.

Method	1500 Edits				3000 Edits			
	Eff.	Gen.	Loc.	Avg.	Eff.	Gen.	Loc.	Avg.
MEMIT	0.18	0.06	0.05	0.10	0.10	0.02	0.01	0.04
O-Edit	0.55	0.40	**0.27**	0.41	0.39	0.30	0.20	0.26
AlphaEdit	0.98	**0.75**	0.19	**0.64**	0.91	**0.69**	0.14	0.58
OrthoEdit	**0.99**	0.72	0.21	**0.64**	**0.99**	0.64	**0.21**	**0.61**

Table 6: Comparison with O-Edit on the CounterFact dataset using LLaMA3-8B model.

Method	**6000 Edits** (batch size=300)			**5000 Edits** (batch size=100)		
	Efficacy↑	Paraphrase↑	Specificity↑	Efficacy↑	Paraphrase↑	Specificity↑
Unedited	$36.09_{\pm 0.30}$	$35.50_{\pm 0.30}$	$31.84_{\pm 0.23}$	$36.35_{\pm 0.30}$	$35.73_{\pm 0.30}$	$31.84_{\pm 0.23}$
FT	$27.48_{\pm 0.26}$	$27.09_{\pm 0.26}$	$14.71_{\pm 0.17}$	$21.08_{\pm 0.24}$	$20.85_{\pm 0.24}$	$9.69_{\pm 0.14}$
MEMIT	$5.80_{\pm 0.19}$	$5.17_{\pm 0.17}$	$0.37_{\pm 0.02}$	$0.07_{\pm 0.01}$	$0.07_{\pm 0.01}$	$1.35_{\pm 0.04}$
PRUNE	$3.18_{\pm 0.12}$	$3.02_{\pm 0.11}$	$1.26_{\pm 0.05}$	$0.09_{\pm 0.01}$	$0.08_{\pm 0.01}$	$1.35_{\pm 0.04}$
RECT	$58.38_{\pm 0.36}$	$55.52_{\pm 0.36}$	$24.58_{\pm 0.21}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
AlphaEdit	$92.25_{\pm 0.17}$	$85.75_{\pm 0.25}$	$30.37_{\pm 0.22}$	$85.60_{\pm 0.24}$	$79.19_{\pm 0.29}$	$26.90_{\pm 0.22}$
OrthoEdit	$\mathbf{94.84}_{\pm 0.13}$	$\mathbf{90.39}_{\pm 0.20}$	$\mathbf{31.34}_{\pm 0.22}$	$\mathbf{94.83}_{\pm 0.13}$	$\mathbf{90.03}_{\pm 0.21}$	$\mathbf{31.27}_{\pm 0.22}$

Table 7: Performance on the ZsRE dataset with LLaMA3-8B. The best results are in bold.

Threshold τ_2	Score	Efficacy	Generalization	Locality	Consistency	Fluency
0.7	83.09	90.03	78.87	81.19	629.97	30.98
0.5	87.19	97.83	90.78	75.94	625.95	33.45
0.3	86.55	98.87	91.40	73.49	620.99	32.88
SD	85.96	99.13	93.42	70.89	620.93	33.75

Table 8: Ablation study of threshold τ_2 on the CounterFact dataset. **SD** denote second-order difference based threshold.

Method	Score	Efficacy	Generalization	Locality	Consistency	Fluency
MEMIT	50.66	$52.36_{\pm 0.50}$	$52.45_{\pm 0.49}$	$47.51_{\pm 0.49}$	$97.04_{\pm 0.14}$	$4.75_{\pm 0.04}$
AlphaEdit	55.14	$60.14_{\pm 0.49}$	$56.97_{\pm 0.43}$	$49.43_{\pm 0.38}$	$362.01_{\pm 0.79}$	$4.35_{\pm 0.06}$
OrthoEdit	86.31	$97.72_{\pm 0.15}$	$89.77_{\pm 0.26}$	$74.70_{\pm 0.26}$	$626.55_{\pm 0.15}$	$33.92_{\pm 0.11}$

Table 9: Cross-batch performance comparison of increased editing frequency on the CounterFact dataset. 5,000 edits was performed under the batch size of 50 and frequency of 100.

Batch	MEMIT			AlphaEdit			OrthoEdit		
	Eff.	Gen.	Loc.	Eff.	Gen.	Loc.	Eff.	Gen.	Loc.
@1	66.0	66.0	33.8	50.0	47.0	50.8	98.0	82.0	70.2
@10	56.0	56.0	41.8	50.0	55.0	43.0	98.0	91.0	71.0
@20	68.0	68.0	31.8	56.0	59.0	40.6	98.0	92.0	72.4
@30	64.0	65.0	36.4	52.0	51.0	53.4	96.0	86.0	80.4
@40	56.0	57.0	43.8	56.0	56.0	46.6	100.0	91.0	72.2
@50	64.0	62.0	36.8	48.0	50.0	51.8	100.0	92.0	80.8
@60	46.0	46.0	54.2	58.0	55.0	47.2	94.0	82.0	75.0
@70	62.0	63.0	36.0	52.0	54.0	46.0	98.0	94.0	71.2
@80	52.0	52.0	47.2	80.0	70.0	44.0	98.0	88.0	74.4
@90	64.0	61.0	39.0	88.0	76.0	49.8	98.0	85.0	71.8
@100	50.0	48.0	50.6	100.0	93.0	44.8	100.0	93.0	70.0

Table 10: In-batch performance comparison of increased editing frequency on the CounterFact dataset. **@k** means evaluation is performed on data samples from the k-th editing batch. **Eff.** denote Efficacy, **Gen.** denote Generalization, and **Loc.** is Locality.