Relaxing the Constraints: A Dual-Importance Projection Mechanism for Lifelong Model Editing

Zhenghai Chen[1], Senbin Xu[1], Jiaxi Tan[1], Xinhua Wu[1], Yan Zhang[1], Xiawu Zheng[1], Shengchuan Zhang[1]*, Ke Li[2], Sicheng Zhao[3], Liujuan Cao[1], Rongrong Ji[1]

[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, [2]Tencent Youtu Lab, [3]Tsinghua University

Correspondence: zsc_2016@xmu.edu.cn

Abstract

Factual knowledge stored in Large Language Models (LLMs) inevitably becomes outdated or erroneous over time, making it critical to update these models without incurring the high cost of retraining. Existing sequential knowledge editing methods predominantly rely on strict orthogonal projection to preserve previously edited knowledge. However, this excessive constraint limits gradient expressiveness, resulting in a significant degradation of model generalization and overall performance as the number of edits increases. To address this challenge, we propose Dual-Importance Projection Editing (DipEdit). This method leverages Singular Value Decomposition (SVD) to identify critical gradient subspaces and introduces a dual mechanism comprising "accumulated importance" and "projection importance." Unlike traditional approaches that enforce strict orthogonality, DipEdit dynamically scales gradient components parallel to key subspaces based on their projection importance rather than discarding them directly. This approach enhances the model's adaptability to new knowledge while maximally preserving historical knowledge. Extensive experiments conducted on five mainstream LLMs using the ZsRE and Counterfact datasets demonstrate that DipEdit effectively handles thousands of sequential edits. The proposed method achieves an average comprehensive performance improvement of 10.36% and effectively maintains the model's general capabilities on downstream tasks. Code is available at: https://github.com/czhhhla/DipEdit.

1 Introduction

Large language models (LLMs) demonstrate exceptional performance across diverse tasks, including text generation and question answering (Brown et al., 2020; OpenAI et al., 2024; Anil et al., 2023; DeepSeek-AI et al., 2025; Zhao et al., 2025). However, as real-world information evolves continuously, the factual knowledge encoded in these models often becomes outdated or erroneous, resulting in hallucinations or inaccuracies in their outputs (Balachandran et al., 2022; Ji et al., 2023). For models with billions of parameters, the computational cost of full retraining is prohibitively high (Mitchell et al., 2022b). Accordingly, knowledge editing has emerged as a promising solution (Meng et al., 2022; Hartvigsen et al., 2023), enabling targeted updates to specific facts in LLMs without necessitating full retraining.

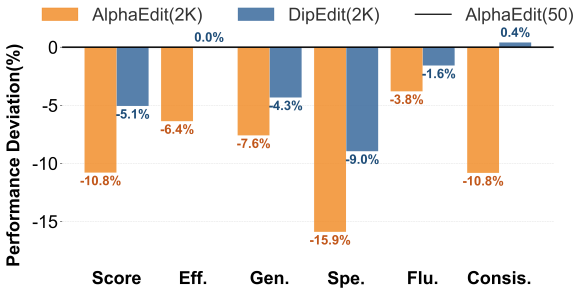

Figure 1: Comparison of performance deviation for the first 50 facts between AlphaEdit and DipEdit after 2,000 sequential edits, relative to their initial post-edit performance.

Although existing research has transitioned from single or batch editing to sequential editing to address real-world demands for frequent knowledge updates (Fang et al., 2024; Jiang et al., 2025b). For instance, Fang et al. (Fang et al., 2024) leverage null space to safeguard original knowledge, substantially mitigating interference with the model's prior facts and performance during sequential edits, but little attention has been paid to knowledge coupling (Xu et al., 2025), wherein new edits interfere with previously edited knowledge. As illustrated in Figure 1, during a sequence of 2000 edits, performance on the initial 50 facts degrades significantly when reevaluated after all edits compared to immediately post their incorporation. The few existing approaches addressing knowledge coupling

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 38308–38333
July 2-7, 2026

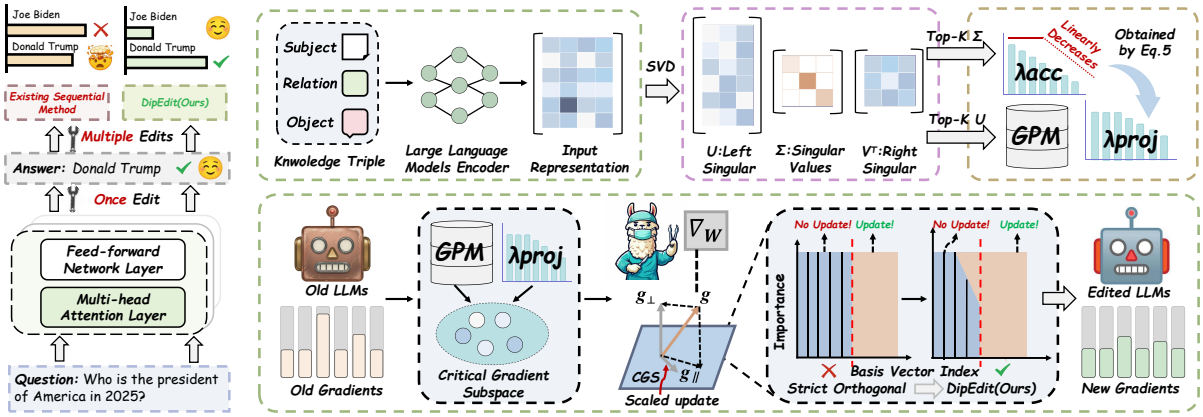

Figure 2: Overview of sequential model editing with DipEdit. The framework identifies the Critical Gradient Subspace (CGS) via SVD to construct the Gradient Projection Memory (GPM). Unlike strict orthogonal projection, DipEdit introduces a dual-importance mechanism (λ_{acc} and λ_{proj}) to dynamically scale gradient components parallel to the CGS based on their significance.

predominantly rely on orthogonal projection (Xu et al., 2025; Cai and Cao, 2024), constraining gradients of new knowledge to directions orthogonal to previously edited knowledge subspaces, thereby mitigating inter-task interference and achieving knowledge decoupling. While these methods partially alleviate coupling, the enforced orthogonality restricts gradient expressiveness, ultimately impairing generalization and editing efficacy in long-sequence editing and causing overall performance degradation.

Inspired by continual learning (Saha and Roy, 2023; Saha et al., 2021), we propose Dual-Importance Projection Editing (DipEdit), a novel knowledge-decoupling editing method that maximizes preservation of previously edited knowledge while enhancing model generalization. We perform singular value decomposition (SVD) on input activations to identify bases spanning the critical gradient subspace (CGS) of edited knowledge and store them in memory. Unlike strict orthogonal projection, we assign two distinct importance measures to these bases: **(1) Accumulated importance,** computed and aggregated from singular values across edits, directly reflecting each basis's significance; **(2) Projection importance,** obtained by assigning 1 to the top-R bases ranked by accumulated importance and linearly decaying values for the remainder. For a new edit, we decompose the gradient into components parallel and orthogonal to the critical subspace. The orthogonal component remains unchanged, while the parallel component is scaled by the corresponding projection importance. The final gradient, combining both, is used to update the model. Furthermore, we introduce an Improved Template Context Mechanism as an auxiliary measure to further enhance model generalization. By leveraging Maximum Marginal Relevance (MMR), this mechanism diversifies instruction semantics.

Our contributions are as follows: (a)We introduce a novel scaled projection mechanism for sequential knowledge editing that resolves interference with previously edited knowledge via projection while alleviating over-constraint through a dual-importance design. (b)We introduce an Improved Template Context Mechanism to mitigate model overfitting to specific phrasings. (c)We conduct extensive experiments on ZsRE and CounterFact using representative models including Phi-1.5, GPT2-XL, GPT-J, Qwen2.5, and LLaMA3, demonstrating the superior effectiveness of DipEdit.

2 Preliminaries

In this section, we introduce the fundamental concepts of model editing and sequential model editing.

Model editing, also known as knowledge editing, refers to a targeted update technique for pre-trained language models. Its primary objective is to adjust model parameter (specifically in parameter-modifying approaches) to alter the output for specific inputs as desired, while preserving the original performance on unrelated inputs as much as possible. Formally, given a base model f_θ and an edit instance (x_e, y_e) where the origin output $f_\theta(x_e) \neq y_e$, model editing aims to produce a new model $f_{\theta'}$ such that $f_{\theta'}(x_e) = y_e$.

Current research primarily focus on editing

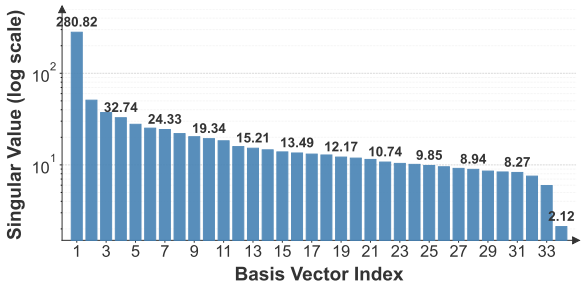

Figure 3: Distribution of singular values obtained from the SVD of the input representation matrix R_1.

knowledge tuples $t = (s, r, o)$, replacing the original object o with a new object o^* to form the updated tuple (s, r, o^*)(Meng et al., 2022; Fang et al., 2024; Xu et al., 2025; Meng et al., 2023). A single edit operation is denoted as $e = (s, r, o, o^*)$. Given a set of knowledge to update $D^* = \{t_i^* = (s_i, r_i, o_i^*)\}_{i=1}^n$ and the original model f_θ, the edited model f_{θ_e} obtained via the edit set E must satisfy:

$$f_{\theta_e}(s_j, r_j) = \begin{cases} o_j^* & \text{if } (s_j, r_j, o_j^*) \in D^*, \\ o_j & \text{otherwise.} \end{cases} \quad (1)$$

Real-world scenarios often involve sequential model editing, when the model undergoes multiple consecutive edits (hundreds or even thousands). Let the initial model be f_{θ_0} and the dynamically growing edit dataset $D_{\text{edit}} = \{(s_t, r_t, o_t^*)\}_{t=1}^T$. At each timestep T, the model editor ME uses the current target data (s_T, r_T, o_T^*) to update the previous model $f_{\theta_{T-1}}$:

$$f_{\theta_T} = \text{ME}(f_{\theta_{T-1}}, s_T, r_T, o_T^*). \quad (2)$$

The edited model f_{θ_T} must adhere to following strict conditions: **Current Edit Reliability**: For the current target knowledge (s_T, r_T), the model must output the new object o_T^*, i.e., $f_{\theta_T}(s_T, r_T) = o_T^*$. **Historical Edit Retention**: For all historical target data $(s_{<T}, r_{<T}, o_{<T}^*) \in D_{\text{edit}}$, even without direct access by ME, the model must output the corresponding new objects $o_{<T}^*$. **General Performance Preservation**: For any unedited input x , the model's output should remain consistent with the initial model f_{θ_0}, i.e., $f_{\theta_T}(x) = f_{\theta_0}(x)$. Specific evaluation metrics are provided in Appendix A.2.

3 Method

In this section, we introduce our Dual-Importance Projection Editing (DipEdit) method. An overview of the proposed DipEdit is illustrated in Figure 2. We address the sequential knowledge editing setting where a model undergoes a sequence of edits over facts indexed by $t \in \{1, 2, ..., \mathcal{T}\}$. Our method comprises three key steps: (1) modify the model parameters, (2) update the important gradient space based on the input representations, and (3) adjust the importance of the gradient basis.

3.1 Editing Initial Fact ($t = 1$)

Modify Parameters. To compute the parameter update Δ for the initial edit, we employ a null-space constrained optimization objective (Fang et al., 2024). This formulation ensures minimal interference with pre-existing knowledge while updating the weights from W_0 to $W_1 = W_0 + \Delta$.

Gradient Subspace Identification. To preserve the knowledge from the first edit, we identify its critical gradient subspace, which initializes the Gradient Projection Memory (GPM). We collect the input representations of the edited fact into a matrix R_1 and perform SVD: $R_1 = U_1 \Sigma_1 V_1^\top$, where U_1 and V_1 are orthogonal matrices, and Σ_1 is a diagonal matrix of singular values. The singular values in Σ_1 quantify the explanatory power of the corresponding left singular vectors in U_1. We select the top-k_1 columns of U_1 as the basis vectors to form the initial memory $\mathcal{M} = [u_{1,1}, \ldots, u_{k_1,1}]$. These bases span the most important space for the input representations of fact 1. As demonstrated by Saha et al. (Saha et al., 2021), these bases also span the most critical gradient space for task 1, with a detailed proof provided in Appendix B.1. The rank k_1 is determined by a Frobenius norm energy threshold $\gamma_{th} \in (0, 1)$:

$$\frac{\|(R_1)_{k_1}\|_F^2}{\|R_1\|_F^2} \geq \gamma_{\text{th}}. \quad (3)$$

Here, $\|\cdot\|_F^2$ denotes the Frobenius norm of a matrix, this thresholding effectively filters out noise, ensuring $\mathcal{M}$ spans the most significant gradient directions.

Compute Importance of Gradient Basis. Existing methods such as KDE (Xu et al., 2025) strictly constrain gradient updates to the orthogonal space of the GPM, prohibiting any updates parallel to the gradient space. However, such rigid constraints result in excessive gradient loss, limiting gradient expressiveness and degrading generalization. To mitigate this, we propose scaling updates based on basis importance, inspired by the observation

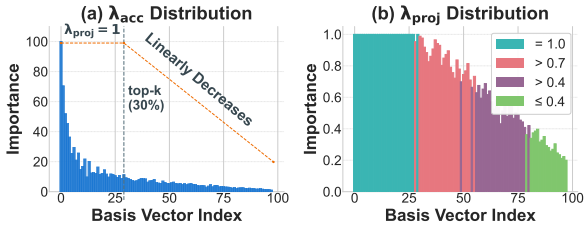

Figure 4: Distribution of accumulated importance (λ_{acc}) and projection importance (λ_{proj}).

that singular values of R_1 exhibit a non-uniform distribution (Figure 3). We first extract the singular value vector $\boldsymbol{\sigma_1} = diag(\Sigma_1, k_1)$ corresponding to the top-k_1 bases, where $\boldsymbol{\sigma_1} \in \mathbb{R}^{k_1 \times 1}$. To quantify the significance of each basis, we compute the accumulated importance $\lambda_{acc_{i,1}} \in [0, 1]$ for the i-th basis:

$$\lambda_{acc_{i,1}} = \frac{(\alpha + 1)\sigma_{i,1}}{\alpha\sigma_{i,1} + \max(\sigma_1)}. \tag{4}$$

Here, α is a non-negative scaling hyperparameter and the value of $\lambda_{acc_{i,1}}$ ranges from 0 to 1. We construct importance vector for this edit by $\boldsymbol{\lambda}_{acc_1} = [\lambda_{acc_{i,1}}, ..., \lambda_{acc_{k_1,1}}]^\top$. Based on this, we derive the projection importance $\lambda_{proj_{i,1}}$ to directly modulate the gradient constraints:

$$\lambda_{proj_{i,1}} = \begin{cases} 1 & L \leq R, \\ \tau + \dfrac{(1-\tau)\cdot(k_1 - L)}{k_1 - R} & \text{otherwise}, \end{cases} \tag{5}$$

where L is the rank of the basis sorted by $\boldsymbol{\lambda}_{acc}$, and $R = \lfloor r \cdot k_1 \rfloor$ denotes the number of fully preserved bases controlled by the retention ratio $r \in (0, 1)$. The hyperparameter $\tau \in (0, 1)$ sets the minimum scaling floor. We construct importance vector for this edit by $\boldsymbol{\lambda}_{proj_1} = [\lambda_{proj_{i,1}}, ..., \lambda_{proj_{k_1,1}}]^\top$. Equation 5 ensures that the most significant bases, namely those with $L \leq R$, are fully preserved with an importance of 1. For subsequent bases, the importance decays linearly from 1 down to τ(as shown in Figure 4), thereby allowing controlled flexibility in gradient updates to mitigate over-constraint while protecting critical knowledge, and enabling knowledge edited with longer time intervals to receive greater protection. We adopt linear decay strategy because it offers a simpler and more intuitive formulation while achieving comparable performance gains (details are provided in Appendix C.7). Note that setting $\tau = 1$ or $r = 1$ recovers the strict orthogonality of KDE. We transfer the $\mathcal{M}$, $\boldsymbol{\lambda}_{\text{acc}}$ and $\boldsymbol{\lambda}_{\text{proj}}$ to the next edit.

Here, $\boldsymbol{\lambda}_{\text{acc}} = \boldsymbol{\lambda}_{\text{acc}_1}$, $\boldsymbol{\lambda}_{\text{acc}} \in \mathbb{R}^{k \times 1}$, $\boldsymbol{\lambda}_{\text{proj}} = \boldsymbol{\lambda}_{\text{proj}_1}$, $\boldsymbol{\lambda}_{\text{proj}} \in \mathbb{R}^{k \times 1}$ with $k = k_1$ is the number of bases after editing fact 1.

3.2 Sequential Editing ($t \in [2, \mathcal{T}]$)

Modify Parameters. To prevent interference with previously edited tasks, which could lead to forgetting of the edited knowledge, and to ensure post-editing accuracy and generalization, we project the new gradient $\nabla_{W_t}\mathcal{L}_t$ during the editing process, yielding the updated gradient:

$$\nabla_{W_t}\mathcal{L}_t = \nabla_{W_t}\mathcal{L}_t - (\mathcal{M}\Lambda\mathcal{M}^\top)(\nabla_{W_t}\mathcal{L}_t). \tag{6}$$

where $\Lambda = \text{diag}(\boldsymbol{\lambda}_{proj})$ encapsulates the constraint strength. This operation leaves gradient components orthogonal to $\mathcal{M}$ unchanged, while scaling components parallel to $\mathcal{M}$ by $(1 - \lambda_{proj_i})$, thereby achieving a balance between stability and plasticity.

Update Important Gradient Space. Upon completing the t^{th} knowledge edit, we update $\mathcal{M}$ by adding the important gradient space of this fact. For that, we first collect the input representations to form the representation matrix R_t, To avoid redundant bases that may overlap or be linearly dependent with those in $\mathcal{M}$, we perform SVD on R_t only after removing such redundancies with the following step:

$$\hat{R}_t = R_t - (\mathcal{M}\mathcal{M}^\top)R_t = R_t - R_{t,\mathcal{M}}. \tag{7}$$

where $R_{t,\mathcal{M}} = (\mathcal{M}\mathcal{M}^\top)R_t$ denotes the projection of R_t onto the space spanned by $\mathcal{M}$. Next, SVD is performed on $\hat{R}_t = \hat{U}_t\hat{\Sigma}_t\hat{V}_t^\top$ and new k_t bases are chosen for minimum k_t satisfying the criteria:

$$||R_{t,\mathcal{M}}||_F^2 + ||(\hat{R}_t)_{k_t}||_F^2 \geq \gamma_{th}||R_t||_F^2. \tag{8}$$

Gradient space in GPM is updated (after $\boldsymbol{\lambda}_{\text{proj}}$ and $\boldsymbol{\lambda}_{\text{acc}}$ update) by adding these new bases to $\mathcal{M}$ as $\mathcal{M} = [\mathcal{M}, \hat{u}_{1,t}, ..., \hat{u}_{k_t,t}]$.

Update Importance of Gradient Basis. We next assign importance to the new bases and refresh that of the existing ones. This process faces two primary obstacles: (1) Distribution mismatch: Since $\hat{R}_t$ is merely a residual subset of R_t, its singular values ($\hat{\Sigma}_t$) fail to accurately reflect the true distribution of R_t, making them unsuitable for direct importance computation via Equation 4. (2) Contribution entanglement: While the existing k bases in $\mathcal{M}$ capture the redundant component of R_t, the global nature of projection prevents decomposing individual basis contributions for the current task.

Table 1: Comparison of DipEdit with existing methods on the sequential model editing task. *Score*, *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Score, Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact						ZsRE		
		Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		12.87	7.85	10.58	89.48	635.23	24.14	36.99	36.34	31.89
MEMIT	Llama3	59.89	65.65	64.65	51.56	437.43	6.58	34.62	31.28	18.49
PRUNE		59.80	68.25	64.75	49.82	418.03	5.90	24.77	23.87	20.69
RECT		63.64	66.05	63.62	61.41	526.62	20.54	86.05	80.54	31.67
AlphaEdit		84.61	98.90	**94.22**	67.88	**622.49**	**32.40**	94.47	91.13	32.55
DipEdit (ours)		**87.35**	**98.95**	93.05	**74.12**	621.35	30.59	**94.80**	**91.65**	**32.76**
Pre-edited		30.36	22.23	24.34	78.53	626.64	31.88	22.19	31.30	24.15
MEMIT	GPT2-XL	77.44	94.70	85.82	60.50	477.26	22.72	79.17	71.44	26.12
PRUNE		68.53	82.05	78.55	53.02	530.47	15.93	21.62	19.27	13.19
RECT		77.86	92.15	81.15	65.13	480.83	21.05	81.02	73.08	24.85
AlphaEdit		83.90	99.50	93.95	66.39	597.88	39.38	94.81	86.11	25.88
DipEdit (ours)		**85.82**	**99.70**	**94.38**	**69.78**	**605.32**	**39.49**	**95.80**	**88.06**	**27.06**
Pre-edited		23.52	16.22	18.56	83.11	621.81	29.74	26.32	25.79	27.42
MEMIT	GPT-J	82.57	98.55	95.50	63.64	546.28	34.89	94.91	90.22	27.56
PRUNE		71.97	86.15	86.85	53.87	427.14	14.78	0.15	0.15	0.00
RECT		84.46	98.80	86.58	72.22	617.31	41.39	96.38	91.21	27.79
AlphaEdit		89.16	99.75	95.80	75.48	618.50	**42.08**	**99.79**	96.00	28.29
DipEdit (ours)		**90.43**	**99.85**	**96.05**	**78.43**	**619.77**	41.11	99.68	**96.44**	**28.35**

To overcome these, we use a method to update importance without requiring historical data:

First, since the redundant bases are not incorporated into $\mathcal{M}$, to capture the importance of these redundant bases for the current task, we perform SVD on $R_{t,\mathcal{M}} = U_{t,\mathcal{M}}\Sigma_{t,\mathcal{M}}V_{t,\mathcal{M}}^{\top}$. Then, we use the coefficient matrix $C = \mathcal{M}^{\top}U_{t,\mathcal{M}}$ to transfer their importance (singular values) to the stored bases in $\mathcal{M}$, where $c_{i,j}$ quantifies the directional similarity. The "surrogate singular values" for $\mathcal{M}$ are computed as:

$$\boldsymbol{\sigma}'_{t,\mathcal{M}} = \sqrt{(C \odot C)(\boldsymbol{\sigma}_{t,\mathcal{M}})^2}, \tag{9}$$

where $\odot$ denotes element-wise multiplication. Subsequently, We combine these with the singular values $\hat{\boldsymbol{\sigma}}_t$ obtained from the new bases (via SVD on $\hat{R}_t$) to construct the full singular value vector for the current task t:

$$\boldsymbol{\sigma}_t = \begin{bmatrix} \boldsymbol{\sigma}'_{t,\mathcal{M}} \\ \hat{\boldsymbol{\sigma}}_t \end{bmatrix} \in \mathbb{R}^{(k+k_t)\times 1}. \tag{10}$$

Thus, for the t^{th} task, we leverage $\boldsymbol{\sigma}_t$ to derive the importance vector for the bases as $\boldsymbol{\lambda}_{acc_t} = f(\boldsymbol{\sigma}_t, \alpha)$ via Equation 4. We then compute and update the cross-task accumulated importance vector $\boldsymbol{\lambda}_{\text{acc}}$. For the existing k bases, the update rule is:

$$\lambda_{acc_i} = \lambda_{acc_i} + \lambda_{acc_{i,t}}, \tag{11}$$

where $i \in [1, k]$ and $\lambda_{acc_{i,t}}$ is the i^{th} element of the previously computed $\boldsymbol{\lambda}_{acc_t}$. Subsequently, the importance of the newly added k_t bases is appended, yielding the updated $\boldsymbol{\lambda}_{\text{acc}} = [\boldsymbol{\lambda}_{acc}^{\top}, \lambda_{\text{acc}_{k+1,t}}, ..., \lambda_{\text{acc}_{k+k_t,t}}]^{\top}$.

As evident from the Equation 11, in long-sequence editing tasks, most importance values exceed 1 due to repeated accumulations (as shown in Figure 4). Directly truncating them to 1 degenerates the process to updates solely in directions orthogonal to the basis space. To mitigate this, we introduce projection importance based on accumulated importance. This design preserves prior edited knowledge while allocating sufficient space for subsequent updates. The updated matrices and vectors are then passed to the next task, repeating the process. Pseudocode for the algorithm is provided in Algorithm 1 of Appendix A.4.

3.3 Improved Template Context Mechanism

To further enhance the model's generalization performance, we propose an Improved Template Context Mechanism to replace the reliance of previous methods on a limited set of fixed prefixes (e.g.,'The','I'). This mechanism incorporates two key optimizations: first, diversified generation, by introducing a prompt library covering various tones such as declarative, interrogative, and hypothetical to enrich the semantic space, and strictly filtering out low-quality samples that are too short or contain degenerate patterns; second, MMR deredundancy, utilizing the Maximum Marginal Relevance (MMR) algorithm to suppress highly similar samples, thereby maximizing the diversity cover-

age of the context distribution. Details are provided in Appendix A.5. In Appendix C.6, we demonstrate that this plays only an auxiliary role in the performance improvement.

4 Experiments

In this section, we empirically address the following research questions to validate the effectiveness of DipEdit: **RQ1:** How does DipEdit perform on sequential editing tasks compared to baseline methods? **RQ2:** What is the comprehensive performance of DipEdit in the more challenging scenario of sequential editing? **RQ3:** How does DipEdit-edited LLM perform on general ability evaluations?

Due to space limitations, we only report the key experimental results in this section. More comprehensive results are provided in Appendix C.

4.1 Experimental Setup

Models. We conduct experiments on five mainstream large language models: Phi-1.5 (1.3B) (Li et al., 2023), GPT2-XL (1.5B) (Radford et al., 2019), GPT-J (6B) (Wang and Komatsuzaki, 2021), Qwen2.5 (7B) (Qwen et al., 2025), and LLaMA3 (8B) (Grattafiori et al., 2024).

Datasets. To evaluate DipEdit's effectiveness, we use two common datasets, ZsRE (Levy et al., 2017) and Counterfact (Meng et al., 2022).

Other Settings. More experimental setup details are provided in Appendix A.

4.2 Overall Performance (RQ1)

To comprehensively evaluate the effectiveness of DipEdit, we conducted systematic experiments across various model architectures and datasets. These experiments involved editing 2K facts in total, divided into 20 rounds with 100 facts per round.

4.2.1 Comparison with SOTA Baselines

We first benchmark DipEdit against established sequential editing baselines. The main results are summarized in Table 1. More experimental results are provided in Appendix C.2 and Appendix C.3.

Obs 1: DipEdit demonstrates superior comprehensive performance in sequential model editing tasks. As shown in Table 1, DipEdit achieves optimal or near-optimal results across the majority of evaluation metrics on the Counterfact and ZsRE datasets. Notably, on the core Score metric, DipEdit outperforms all baselines. It also maintains top-tier performance in key metrics such as editing efficacy and generalization, while preserving high-quality language fluency and consistency.

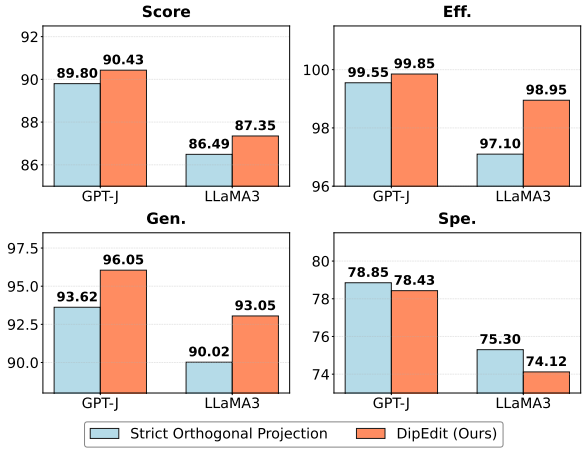

Figure 5: Performance comparison between Strict Orthogonal Projection and DipEdit on sequential editing tasks using LLaMA3 and GPT-J under an identical framework.

4.2.2 Mechanistic Analysis: Scaling vs. Strict Projection

To further investigate the effectiveness of the dual-importance projection, we conduct a focused ablation study by comparing DipEdit against a Strict Orthogonal Projection mechanism.

Implementation Note: Strict orthogonal projection is the theoretical core of recent works such as KDE (Xu et al., 2025). Since KDE's code is not publicly available, we implemented its core mechanism within our own unified framework (e.g., incorporating our Improved Template Context) to ensure a fair, mechanism-level comparison. This baseline is referred to as "Strict Orthogonal Projection.

Obs 2: DipEdit's "soft" gradient scaling optimizes the balance between stability and plasticity. As illustrated in Figure 5, compared to the strict projection (KDE core), DipEdit achieves superior overall Scores and generalization on both GPT-J and LLaMA3. For instance, on LLaMA3, efficacy and generalization improve significantly while specificity decreases only minimally (from 75.30 to 74.12). This suggests that moderate gradient scaling provides the necessary flexibility for long-sequence editing that hard constraints lack.

While this section focuses on the primary projection mechanism, a more exhaustive ablation study is provided in Appendix C.6 to further substantiate our findings.

Table 2: Extended to 5000 and 10000 edits on Counterfact and ZsRE (GPT2-XL)

Method	K = 5000									K = 10000								
	Counterfact						ZsRE			Counterfact						ZsRE		
	Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Con.↑	Eff.↑	Gen.↑	Spe.↑	Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Con.↑	Eff.↑	Gen.↑	Spe.↑
MEMIT	64.87	74.22	66.71	56.23	570.3	15.50	33.26	29.71	13.44	61.49	68.23	61.21	56.19	529.2	7.95	5.98	5.32	2.50
RECT	72.52	90.58	73.79	59.61	500.3	15.86	66.79	60.06	20.87	67.80	86.05	67.61	56.07	538.8	11.29	13.24	12.26	2.00
AlphaEdit	79.04	97.72	88.15	61.06	567.2	34.89	82.90	72.50	21.36	71.09	89.96	73.83	57.02	**586.4**	32.72	62.82	54.31	13.76
DipEdit	**82.73**	**98.40**	**89.98**	**66.72**	**587.4**	**37.81**	**86.86**	**77.11**	**23.54**	**77.01**	**93.41**	**82.00**	**62.28**	585.3	**35.61**	**71.30**	**61.03**	**19.67**

Table 3: Performance evaluation of AlphaEdit and DipEdit on homogeneous sequential editing tasks using GPT2-XL and GPT-J across the constructed "City" and "Language" datasets.

Data	Model	Method	Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Con.↑
Language	GPT-J	AlphaEdit	86.49	99.75	**96.82**	69.77	619.0	**38.42**
		DipEdit	**87.65**	**99.90**	96.52	**72.17**	**620.1**	37.75
	GPT2-XL	AlphaEdit	82.15	99.25	93.20	63.64	571.1	31.32
		DipEdit	**85.22**	**99.60**	**94.10**	**68.80**	**591.9**	**34.05**
City	GPT-J	AlphaEdit	87.07	99.75	94.88	72.00	605.8	**36.05**
		DipEdit	**88.82**	**99.80**	**95.55**	**75.42**	**609.3**	35.68
	GPT2-XL	AlphaEdit	80.94	98.75	93.45	61.58	585.6	36.92
		DipEdit	**84.40**	**99.10**	**94.48**	**67.24**	**594.7**	**37.60**

4.3 Robustness of Sequential Editing (RQ2)

To comprehensively evaluate the robustness of DipEdit under extreme conditions, we designed two challenging experimental scenarios: **Homogeneous Fact Sequential Editing** and **Ultra-Long Sequence Sequential Editing**. The former assesses knowledge retention under high-interference conditions, while the latter probes the performance boundaries of the model during extended editing sequences.

4.3.1 Homogeneous Fact Sequential Editing

Li and Yin et al. (Li et al., 2024b; Yin et al., 2023) showed that sequential edits within the same knowledge category affect previously edited facts more than edits across categories. To test DipEdit's robustness in this setting, we created a dataset of 2,000 language-related and 2,000 city-related facts from Counterfact (details are provided in Appendix D.3) and performed sequential editing on GPT2-XL and GPT-J, with results illustrated in Table 3. Based on the experimental results, the following conclusions are drawn:

Obs 3: DipEdit maintains SOTA performance in the highly challenging scenario of homogeneous fact sequential editing. In detail, on the language dataset, DipEdit achieves comprehensive scores of 87.65 and 85.22 on GPT-J and GPT2-XL, respectively, consistently outperforming AlphaEdit. Notably, while maintaining superior editing efficacy and generalization, DipEdit also excels in core capabilities such as fluency and consistency. These results confirm that DipEdit possesses exceptional stability in dense homogeneous knowledge editing scenarios, effectively balancing knowledge updates with the preservation of model performance.

4.3.2 Ultra-Long Sequence Sequential Editing

To further explore performance boundaries under ultra-long sequences, we extended the experimental scope from 2,000 to 5,000, and subsequently to 10,000 edits. The results are presented in Table 2.

Obs 4: DipEdit consistently maintains SOTA performance in long-sequence editing, with its advantage amplifying as the sequence grows. Specifically, DipEdit achieves superior results in both 5k and 10k settings. Notably, the performance gap between DipEdit and AlphaEdit widens as the number of edits increases. On Counterfact, the lead in comprehensive score expands from 3.69% at 5k to 5.92% at 10k. A similar trend is observed on ZsRE, where the disparity in editing efficacy increases from 3.96% (5k) to 8.48% (10k). This divergence stems from AlphaEdit's susceptibility to cumulative interference, which degrades retained knowledge. In contrast, DipEdit effectively isolates and protects edited knowledge, significantly mitigating performance degradation while preserving high generalization and efficacy.

4.4 General Capability Tests (RQ3)

To assess whether DipEdit compromises the general capabilities of LLMs under large-scale sequential editing, we conducted evaluations on the GLUE benchmark and the MMLU dataset (details are provided in Appendix A.1.3). The results are summarized in Figure 6, with more experiments provided in Appendix C.4.

Obs 5: DipEdit sustains the general capability

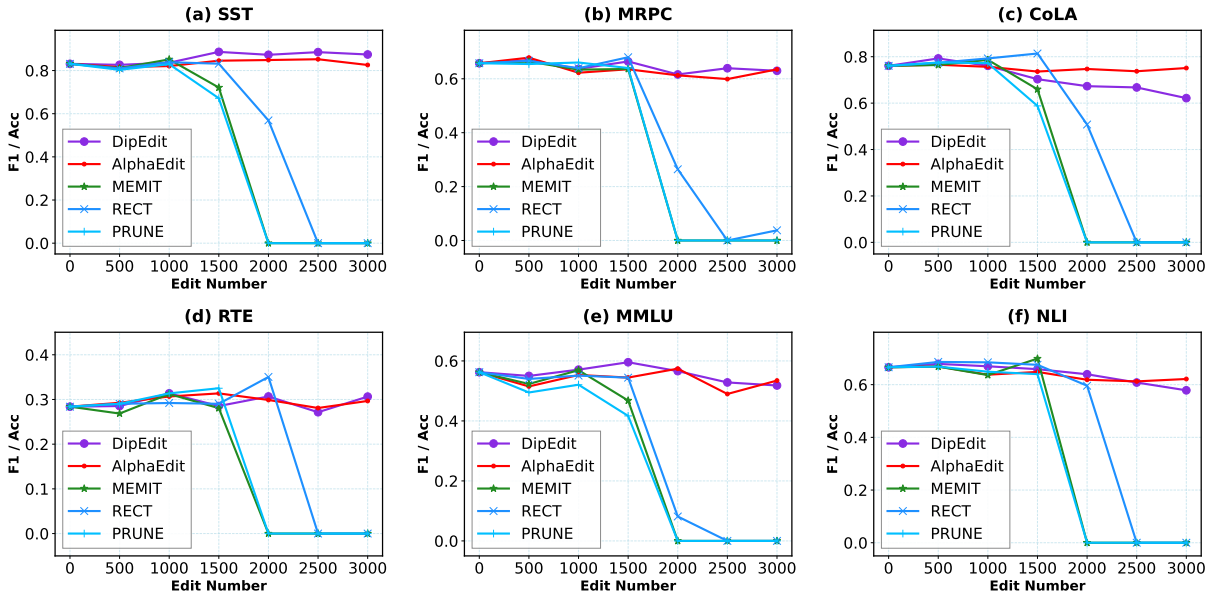

Figure 6: F1 scores of the LLaMA3 (8B) model after editing across six general capability benchmarks (i.e., SST, MRPC, CoLA, RTE, MMLU, and NLI).

of post-edited LLMs even after extensive editing. Mainstream model editing methods show severe instability, with performance dropping notably after 1,500 edits and catastrophic failure occurring by 2,000 edits. In contrast, DipEdit remains robust, maintaining core capabilities even after 3,000 sequential edits. Among baselines, only AlphaEdit also withstands 3,000 edits, but DipEdit excels in semantic understanding—achieving 87.44% accuracy on SST versus AlphaEdit's 82.58%. These results confirm that DipEdit effectively avoids catastrophic forgetting while preserving both editing efficacy and general reasoning.

5 Related work

Current model editing approaches are primarily categorized into parameter-preserving and parameter-modifying methods (Yao et al., 2023; Zhang et al., 2024b). Parameter-preserving techniques maintain original model weights by introducing external mechanisms: SERAC (Mitchell et al., 2022b) employs a separate counterfactual model; IKE (Zheng et al., 2023) utilizes in-context demonstrations; while T-Patcher (Huang et al., 2023), CaliNet (Dong et al., 2022), GRACE (Hartvigsen et al., 2023), and WISE (Wang et al., 2025) incorporate additional trainable neurons or dynamic memory codebooks to enable updates. Conversely, parameter-modifying methods directly update internal weights: FT-W (Zhu et al., 2020) and KN (Dai et al., 2022) fine-tune specific layers or neurons; meta-learning approaches like MEND (Mitchell et al., 2022a), MALMEM (Tan et al., 2024), InstructEdit (Zhang et al., 2024a), and DAFNET (Zhang et al., 2024c) predict weight adjustments via hypernetworks (Li et al., 2025); and "Locate-Then-Edit" methods such as ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), PMET (Li et al., 2024a), RECT (Gu et al., 2024), AlphaEdit (Fang et al., 2024), and AnyEdit (Jiang et al., 2025a) directly modify knowledge-storing layers for large-scale updates. This study focuses on the latter category of parameter-modifying editing.

6 Conclusion

In this paper, we address the stability-plasticity dilemma in sequential knowledge editing by proposing DipEdit. Unlike strict orthogonal projection methods that over-constrain gradient updates, DipEdit introduces a novel dual-importance mechanism to dynamically scale gradients, achieving a balance between preserving historical knowledge and adapting to new information. Extensive experiments on five mainstream Large Language Models demonstrate that DipEdit significantly outperforms state-of-the-art baselines, effectively handling up to 10,000 sequential edits while maintaining the models' general capabilities. DipEdit also exhibits robustness in complex scenarios, such as homogeneous and ultra-long sequence editing. Our work provides a robust and scalable solution for the lifelong maintenance of Large Language Models.

Limitations

Despite the superior performance of DipEdit in sequential editing, several limitations remain. First, the reliance on SVD for identifying gradient subspaces introduces computational overhead that scales with the model's hidden dimension, potentially constraining efficiency on extremely large-scale models. Second, while our extensive sensitivity analysis confirms the method's robust plug-and-play capability with generic parameters, pinpointing the absolute optimal configuration for specific architectures remains a non-trivial task. Finally, as our evaluation primarily targets factual knowledge updates, the applicability of DipEdit to complex logical or procedural editing warrants further investigation.

References

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, and 109 others. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

Vidhisha Balachandran, Hannaneh Hajishirzi, William Cohen, and Yulia Tsvetkov. 2022. Correcting diverse factual errors in abstractive summarization via post-editing and language model infilling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9818–9830, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The fifth PASCAL recognizing textual entailment challenge. In *TAC*. NIST.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *ACL (1)*, pages 8493–8502. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*. Asian Federation of Natural Language Processing.

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

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: lifelong model editing with discrete key-value adaptors. In *NeurIPS*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*. OpenReview.net.

Zeyu Huang, Yikang Shen, Xiaofeng Zhang, Jie Zhou, Wenge Rong, and Zhang Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *ICLR*. OpenReview.net.

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 Computing Surveys*, 55(12):1–38.

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

Houcheng Jiang, Junfeng Fang, Tianyu Zhang, Baolong Bi, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. 2025b. Neuron-level sequential editing for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16678–16702, Vienna, Austria. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *CoNLL*, pages 333–342. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024a. PMET: precise model editing in a transformer. In *AAAI*, pages 18564–18572. AAAI Press.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. 2025. Reinforced lifelong editing for language models. *arXiv preprint arXiv:2502.05759*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024b. Unveiling the pitfalls of knowledge editing for large language models. *Preprint*, arXiv:2310.02129.

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.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *NeurIPS*.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *ICLR*. OpenReview.net.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *ICLR*. OpenReview.net.

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

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Gobinda Saha, Isha Garg, and Kaushik Roy. 2021. Gradient projection memory for continual learning. In *ICLR*. OpenReview.net.

Gobinda Saha and Kaushik Roy. 2023. Continual learning with scaled gradient projection. *Preprint*, arXiv:2302.01386.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. ACL.

Chenmien Tan, Ge Zhang, and Jie Fu. 2024. Massive editing for large language models via meta learning. *Preprint*, arXiv:2311.04661.

Ben Wang and Aran Komatsuzaki. 2021. Gpt-j-6b: A 6 billion parameter autoregressive language model.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL-HLT*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

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

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. In *EMNLP*, pages 10222–10240. Association for Computational Linguistics.

Xunjian Yin, Jin Jiang, Liming Yang, and Xiaojun Wan. 2023. History matters: Temporal knowledge editing in large language model. *Preprint*, arXiv:2312.05497.

Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. MELO: enhancing model editing with neuron-indexed dynamic lora. In *AAAI*, pages 19449–19457. AAAI Press.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024a. Instructedit: Instruction-based knowledge editing for large language models. In *IJCAI*, pages 6633–6641. ijcai.org.

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, and 3 others. 2024b. A comprehensive study of knowledge editing for large language models. *CoRR*, abs/2401.01286.

Taolin Zhang, Qizhou Chen, Dongyang Li, Chengyu Wang, Xiaofeng He, Longtao Huang, Hui Xue', and Jun Huang. 2024c. DAFNet: Dynamic auxiliary fusion for sequential model editing in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1588–1602, Bangkok, Thailand. Association for Computational Linguistics.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2025. A survey of large language models. *Preprint*, arXiv:2303.18223.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *CoRR*, abs/2305.12740.

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

Appendix

In the appendix, we detail the experimental setup, theoretical analysis, more experimental results, and dataset visualizations.

- Appendix A: Experimental Setup.
- Appendix B: Theoretical Analysis.
- Appendix C: More Experimental Results.
- Appendix D: Dataset Visualizations.

A Experimental Setup

A.1 Dataset

A.1.1 ZsRE

ZsRE (Levy et al., 2017) is a Wikipedia-based question-answering (QA) benchmark designed to evaluate zero-shot generalization on specific relations. Utilizing back-translation, ZsRE constructs a rich semantically equivalent questions to rigorously test the robustness of edited models against diverse formulations. Following prior research, we employ Natural Questions as out-of-distribution (OOD) data to assess locality. Each sample comprises a subject entity and a target answer for measuring editing efficacy, a rephrased question for generalization, and a distinct locality question for specificity evaluation.

A.1.2 CounterFact

CounterFact (Meng et al., 2022) represents a more challenging benchmark designed to assess knowledge editing under counter-intuitive scenarios. It strictly evaluates the model's ability to update specific knowledge while preserving the stability of related factual knowledge by introducing targets that contradict established facts. Out-of-scope data is constructed by substituting the subject entity with a similar entity sharing the same predicate. Each sample includes a prompt template and a corresponding counterfactual target for efficacy evaluation, alongside multiple semantically consistent paraphrases for generalization. Crucially, it features "Neighborhood Prompts"—prompts with similar attributes but distinct factual independence—serving as a rigorous baseline for locality to ensure the update does not trigger catastrophic forgetting or erroneous modifications to semantically adjacent entities. Additionally, generation prompts are included to evaluate text generation quality, focusing on fluency and consistency.

A.1.3 General Capabilities Evaluation

SST (Socher et al., 2013) is a binary sentiment analysis dataset based on movie reviews, assessing the model's understanding of affective semantics.
CoLA (Warstadt et al., 2019) is a corpus of sentences from linguistic literature used to evaluate the model's perception of grammatical acceptability.
RTE (Bentivogli et al., 2009) is a compilation of textual entailment datasets designed to assess logical reasoning capabilities in low-resource settings.
MRPC (Dolan and Brockett, 2005) is a dataset of sentence pairs automatically extracted from online news, requiring the model to detect semantic equivalence (paraphrasing).
NLI (Williams et al., 2018) requires the analysis of logical relationships between sentences, serving as a core metric for deep semantic understanding and complex inference.
MMLU (Hendrycks et al., 2021) is a massive multitask benchmark covering diverse disciplines (e.g., STEM, humanities, social sciences) to evaluate multidisciplinary knowledge and reasoning in zero-shot and few-shot settings.

A.2 Evaluation Metrics

In this section, we outline the evaluation metrics employed in our experiments. Given the similarity of our tasks to prior work, we directly adopt the metric definitions established by Fang et al. (2024)

A.2.1 ZsRE Metrics

In this section, we define the evaluation metrics for the ZsRE dataset. Let f_θ denote the large language model, (s_i, r_i) the input prompt containing factual knowledge, o_i^* the post-edit target output, and o_i the model's original output.

Efficacy. This metric evaluates whether the new knowledge has been successfully injected. Specifically, it calculates the average accuracy where the Top-1 output exactly matches the target o_i^* given the edit prompt (s_i, r_i):

$$\mathbb{E}_i \left\{ o_i^* = \arg\max_o \mathbb{P}_{f_\theta}(o \mid (s_i, r_i) \right\}. \quad (12)$$

Generalization. This metric measures the model's robustness to inputs that are semantically equivalent but distinct in formulation. Using a set of paraphrases $\mathcal{N}((s_i, r_i))$, we calculate the average Top-1 accuracy of generating the correct target o_i^*:

$$\mathbb{E}_i \left\{ o_i^* = \arg\max_o \mathbb{P}_{f_\theta}(o \mid \mathcal{N}((s_i, r_i)) \right\}. \quad (13)$$

Specificity. This metric verifies the locality of the edit, ensuring that the model's memory of non-target knowledge remains undisturbed. We select a set of unrelated samples $O((s_i, r_i))$ and evaluate the average Top-1 accuracy of the model retaining the original ground truth o_i:

$$\mathbb{E}_i \left\{ o_i = \arg\max_o \mathbb{P}_{f_\theta}(o \mid O((s_i, r_i)) \right\}. \quad (14)$$

A.2.2 CounterFact Metrics.

In this section, we define the evaluation framework for the CounterFact dataset. The variables f_θ, (s_i, r_i), o_i, and o_i^* are defined as above.

Efficacy. Measures the extent to which the model accepts counterfactual knowledge. It calculates the proportion of samples where the prediction probability of the target edit word o_i^* exceeds that of the original word o_i given the original prompt (s_i, r_i):

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i^* \mid (s_i, r_i)] > \mathbb{P}_{f_\theta}[o_i \mid (s_i, r_i)] \right]. \quad (15)$$

Generalization. Evaluates the transferability of the edit under semantic rephrasing. This metric tracks the proportion of samples where the probability of predicting o_i^* is higher than o_i given the paraphrased prompts $\mathcal{N}((s_i, r_i))$.

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i^* \mid \mathcal{N}((s_i, r_i))] > \mathbb{P}_{f_\theta}[o_i \mid \mathcal{N}((s_i, r_i))] \right]. \quad (16)$$

Specificity. Examines the precise boundary of the edit. Using neighborhood prompts $O((s_i, r_i))$—prompts with distinct subjects but similar semantic properties—this metric calculates the proportion of samples where the model correctly assigns a higher probability to the original fact, ensuring the update does not erroneously spill over to semantically adjacent entities.

$$\mathbb{E}_i \left[\mathbb{P}_{f_\theta}[o_i^* \mid O((s_i, r_i))] > \mathbb{P}_{f_\theta}[o_i \mid O((s_i, r_i))] \right]. \quad (17)$$

Fluency. Quantifies the naturalness and repetitiveness of the generated text. It is measured by a weighted combination of the entropy of the n-gram distribution:

$$-\frac{2}{3} \sum_k g_2(k) \log_2 g_2(k) + \frac{4}{3} \sum_k g_3(k) \log_2 g_3(k), \quad (18)$$

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

Consistency. Assesses factual consistency in generation. We utilize the cosine similarity of TF-IDF vectors between the text generated by the model regarding subject s and the Wikipedia reference text for o as the consistency score.

Score. To provide a holistic evaluation, we calculate the harmonic mean of the three core metrics, including Efficacy (E), Generalization (G), and Specificity (L), to serve as the final score:

$$\text{Score} = \frac{3}{\frac{1}{E} + \frac{1}{G} + \frac{1}{L}}. \quad (19)$$

A.3 Baseline Methods

In this section, we detail the baseline methods employed in our comparative analysis.

MEMIT (Meng et al., 2023) extends the ROME paradigm to overcome scalability limitations. By employing least-squares optimization, it distributes updates for thousands of new facts smoothly across multiple layers. This approach enables the simultaneous update of massive knowledge entries while preserving the model's general reasoning abilities and stability.

PRUNE (Ma et al., 2024) focuses on preserving general capabilities during sequential editing. To mitigate performance degradation caused by accumulated edits, it applies constraints on the condition number of the update matrix. By filtering out updates that heighten numerical sensitivity, PRUNE ensures knowledge injection does not compromise the structural integrity of the model.

RECT (Gu et al., 2024) addresses side effects arising from excessive weight modifications. It proposes a regularization-based solution that constrains the relative change of parameter weights. This filters out redundant updates that contribute minimally to new knowledge but significantly disrupt the original model structure, thereby balancing editing performance with general capability retention.

AlphaEdit (Fang et al., 2024) adopts a geometric perspective to mitigate catastrophic forgetting. It employs null-space projection to map parameter updates into the null space of the pre-existing knowledge covariance matrix. This method enhances techniques like MEMIT by ensuring unrelated query outputs remain unchanged, significantly improving reliability and generalization.

SERAC (Mitchell et al., 2022b) represents a memory-based approach that leaves the LLM parameters frozen. It utilizes a "Semi-Parametric Editing" framework where edits are stored in an explicit memory and processed by a retrieval-augmented counterfactual model. While this circumvents catastrophic forgetting entirely, it introduces increased computational complexity during inference.

GRACE (Hartvigsen et al., 2023) performs lifelong knowledge editing by storing edits as a discrete key-value codebook within the latent space. It handles thousands of sequential modifications without altering model weights, effectively addressing performance degradation in streaming error correction settings while minimizing impact on unrelated knowledge.

MELO (Yu et al., 2024) is a plug-in knowledge editing method that enhances LLMs with dynamic LoRA blocks indexed by neurons. It modifies behavior efficiently through activation via an internal vector database. MELO ensures independence between distinct edits and significantly reduces storage and computational costs.

A.4 DipEdit Algorithm

For the pseudocode of Dual Importance Scaled Gradient Projection Editing (DipEdit), please refer to Algorithm 1.

A.5 Improved Template Context Mechanism

To maximize the distributional coverage of the context templates without incurring the computational overhead of external semantic encoders (e.g., BERT), we introduce a lightweight Hash-based Pseudo-Embedding strategy coupled with the Maximal Marginal Relevance (MMR) criterion. The implementation details are as follows:

Deterministic Pseudo-Vector Mapping: Instead of relying on heavy pre-trained language models, we devise a zero-dependency vectorization approach. For each generated candidate template t, we compute its MD5 hash to serve as a deterministic seed. This seed is then used to sample a feature vector $v_t \in \mathbb{R}^{32}$ from a standard normal distribution $\mathcal{N}(0, 1)$.

MMR-based Selection: We employ the MMR algorithm to iteratively select a subset of K templates. At each step, the algorithm selects the candidate t_i that maximizes:

$$\text{MMR} = \lambda \cdot \text{Sim}(v_{t_i}, Q) - (1-\lambda) \cdot \max_{t_j \in S} \text{Sim}(v_{t_i}, v_{t_j})$$

where S is the set of already selected templates, Q is the centroid of the candidate pool, and Sim denotes cosine similarity.With $\lambda = 0.5$, this mechanism functions as a deterministic stratified sampler. It rigorously penalizes duplicates (where Sim ≈ 1.0) and selects candidates that are uniformly distributed across the randomized projection space.

A.6 Implementation Details

We detail the specific configurations used in our experiments below. All evaluations were conducted on a single NVIDIA A800 (80GB) GPU, and large language models were instantiated using the HuggingFace Transformers library (Wolf et al., 2020). For a detailed ablation study on the choice of the proposed hyperparameters ($\gamma_{th}, \alpha, r, t$), please refer to Appendix C.1.

- **GPT-J:** We designate layers 3–8 as the critical layers for editing. We set the hyperparameter $\lambda = 15000$ and perform 40 optimization steps for the hidden representation calculation with a learning rate of 0.5. The specific parameters are set to $\gamma_{th} = 0.7$, $\alpha = 0.5$, $r = 0.4$, and $t = 0.7$.
- **GPT2-XL:** Layers 13–17 are selected as critical layers. We set $\lambda = 20000$ and execute 20 optimization steps with a learning rate of 0.5. The parameters are configured as $\gamma_{th} = 0.7$, $\alpha = 0.5$, $r = 0.5$, and $t = 0.5$.
- **Llama3-8b:** We target layers 4–8. We set $\lambda = 15000$ and perform 25 optimization steps with a learning rate of 0.5. The parameters are set to $\gamma_{th} = 0.7$, $\alpha = 0.5$, $r = 0.2$, and $t = 0.4$.

B Theoretical Analysis

In this section, we provide a rigorous theoretical examination of our proposed method.

B.1 Relationship between Input Representations and Gradient Space

Our method relies on constraining gradient updates within the subspace spanned by input representations stored in the gradient projection memory. Here, we formally prove that the gradient

Algorithm 1 Algorithm for DipEdit

Input: The initial LLM model f_{θ_0}, the edit dataset $D_{\text{edit}} = \{(s_t, r_t, o_t^*)\}_{t=1}^{T}$, the loss function $\mathcal{L}$, the hyperparameters $\gamma_{th}, \alpha, \eta$, the initial weight W_0.
Output: The final LLM model after f_{θ_T} after T edits.
Initialize: Gradient Projection Memory $\mathcal{M} \leftarrow \emptyset$, Accumulated Importance $\boldsymbol{\lambda}_{acc} \leftarrow \emptyset$, Projection Importance $\boldsymbol{\lambda}_{proj} \leftarrow \emptyset$.
for each edit $(s_t, r_t, o_t^*) \in D_{\text{edit}}$, where $t \in [1, T]$ **do**
    $\nabla_{W_t}\mathcal{L}_t \leftarrow SGD((s_t, r_t, o_t^*), \mathcal{L})$
    if $\mathcal{M}$ is not empty **then**
        $\nabla_{W_t}\mathcal{L}_t \leftarrow project(\nabla_{W_t}\mathcal{L}_t, \mathcal{M})$
    end if
    $\Delta_{target} \leftarrow \Delta_{target} - \eta\nabla_{W_t}\mathcal{L}_t$ ▷ η denotes the learning rate.
    $W_1 = W_0 + \Delta_{target}$
    $R_t \leftarrow forward((s_t, r_t), W_1)$
    if $\mathcal{M}$ is not empty **then**
        $\hat{R}_t \leftarrow R - R_{t,M}$ ▷ see Equation 6
        $\hat{U}_t, \hat{\Sigma}_t \leftarrow SVD(\hat{R}_t)$
        $k_t \leftarrow criteria(\hat{R}_t, R_t, \gamma_{th})$ ▷ see Inequality 8
        $k = M.shape[1]$ ▷ get the number of (old) basis in $\mathcal{M}$ after task $t-1$
        $U_{t,M}, \Sigma_{t,M} \leftarrow SVD(R_{t,M})$
        $C \leftarrow M^\top U_{t,M}$
        $\boldsymbol{\sigma}_t \leftarrow SingularValueVector(C, \Sigma_{t,M}, \hat{\Sigma}_t)$ ▷ see Equation 10
        $\boldsymbol{\lambda}_{acc_t} \leftarrow f(\boldsymbol{\sigma}_t), \alpha$ ▷ Accumulated importance vector for t^{th} task, see Equation 4
        $\boldsymbol{\lambda}_{acc} \leftarrow AccumulateImportance(\boldsymbol{\lambda}_{acc}, \boldsymbol{\lambda}_{acc_t}, k)$ ▷ importance accumulation for the old k basis, see Equation 11
        $\boldsymbol{\lambda}_{acc} \leftarrow [\boldsymbol{\lambda}_{acc}^\top, \boldsymbol{\lambda}_{acc_t}[k : k + k_t]^\top]^\top$ ▷ Accumulated importance vector update
        $\boldsymbol{\lambda}_{proj} \leftarrow getProjectionImportance(\boldsymbol{\lambda}_{acc})$ ▷ get Projection Importance via Accumulated Importance, see Equation 5
    else
        $k_1 \leftarrow criteria(R_t, \gamma_{th})$ ▷ see Inequality 3
        $U_1, \Sigma_1 \leftarrow SVD(R_t)$
        $\boldsymbol{\sigma}_1 \leftarrow Top - k_1(\Sigma_1)$
        $\boldsymbol{\lambda}_{acc_1} \leftarrow f(\boldsymbol{\sigma}_1, \alpha)$
        $\boldsymbol{\lambda}_{acc} \leftarrow \boldsymbol{\lambda}_{acc_1}$ ▷ Accumulated importance vector update
        $\boldsymbol{\lambda}_{proj} \leftarrow getProjectionImportance(\boldsymbol{\lambda}_{acc})$
    end if
    $\mathcal{M} \leftarrow [\mathcal{M}, U_t[0 : k_t]]$ ▷ Basis matrix (GPM) update
end for
return the post-edit LLM model f_{θ_T}

space is indeed a subset of the input representation space.Consider a linear layer where the forward propagation is defined as follows:

$$h_l = \sigma_l(W_l^\top h_{l-1} + b_l), \tag{20}$$

where $h_{l-1} \in \mathbb{R}^{d_{in}}$ denotes the input vector, $W_l \in \mathbb{R}^{d_{in} \times d_{out}}$ is the weight matrix, b_l is the bias vector, and σ_l represents the activation function. Let $\mathcal{L}$ be the loss function. Applying the chain rule, the gradient with respect to W_l is derived from the outer product of the input and the back-propagation error:

$$\frac{\partial \mathcal{L}}{\partial W_l} = h_{l-1}(\delta_l)^\top, \tag{21}$$

where $\delta_l = \frac{\partial \mathcal{L}}{\partial h_l} \odot \sigma_l' \in \mathbb{R}^{d_{out}}$ is the error vector.

Equation 21 explicitly indicates that every column of the gradient matrix $\frac{\partial \mathcal{L}}{\partial W_l}$ is a scalar multiple of the input vector h_{l-1}. For a batch of input data with a representation matrix R, the total gradient is a linear combination of the columns of R. There-

fore, the following inclusion holds:

$$\text{span}(\nabla W_l) \subseteq \text{span}(R), \tag{22}$$

This justifies the rationale for performing Singular Value Decomposition (SVD) on R. By identifying the principal basis M of the input representations, we fundamentally capture the subspace in which the critical gradients reside.

B.2 Optimality of Basis Selection via the Eckart-Yang Theorem

To effectively store historical knowledge, we must compress the input representation matrix R into a low-rank basis matrix M while minimizing information loss. The theoretical justification for our selection strategy rests on the Eckart-Young-Mirsky theorem.

For any matrix $A \in \mathbb{R}^{m\times n}$ with rank r^*, its Singular Value Decomposition (SVD) is expressed as:

$$A = U\Sigma V^\top, \tag{23}$$

where U and V are orthogonal matrices containing the left and right singular vectors, respectively, and Σ is a diagonal matrix containing the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{r^*} > 0$.

Let A_k denote the rank-k approximation of A (where $k < r^*$), constructed by retaining the top-k singular values and their corresponding vectors:

$$A_k = U_k\Sigma_k V_k^\top = \sum_{i=1}^{k} \sigma_i u_i v_i^\top. \tag{24}$$

The Eckart-Young theorem asserts that under the Frobenius norm, A_k is the optimal rank-k approximation of A. Specifically, it solves the following minimization problem:

$$\min_{\text{rank}(B)=k} \|A - B\|_F$$

Furthermore, the minimum approximation error is explicitly determined by the tail energy of the singular values:

$$\|A - A_k\|_F = \sqrt{\sum_{i=k+1}^{r^*} \sigma_i^2}. \tag{25}$$

Summary: Our selection criterion Equation 8 ensures that for a defined retained energy threshold γ_{th}, the selected basis M captures the maximum possible information regarding historical inputs. This theoretically minimizes the "information loss" induced by memory compression, thereby ensuring that the projection constraints remain as accurate as possible.

B.3 Analysis of the Gradient Projection Mechanism

Let $\mathbf{g} = \nabla_W \mathcal{L}$ denote the original gradient and $\tilde{\mathbf{g}}$ denote the updated gradient obtained via Equation 6. Let $\mathcal{S} = \text{span}(M)$ represent the subspace spanned by the orthogonal basis vectors in M. By decomposing the gradient into $\mathbf{g} = \mathbf{g}_\parallel + \mathbf{g}_\perp$, where $\mathbf{g}_\parallel \in \mathcal{S}$ and $\mathbf{g}_\perp \perp \mathcal{S}$, we demonstrate that the projection mechanism guarantees the following properties:

1. The orthogonal component remains invariant: $\tilde{\mathbf{g}}_\perp = \mathbf{g}_\perp$.
2. The parallel component is scaled element-wise by $(1 - \lambda_i)$: $\tilde{\mathbf{g}}_\parallel = M(I - \Lambda)M^\top \mathbf{g}_\parallel$.

Proof: Based on Equation 6, the update rule is defined as:

$$\tilde{\mathbf{g}} = \mathbf{g} - M\Lambda M^\top \mathbf{g} = (I - M\Lambda M^\top)\mathbf{g}, \tag{26}$$

where $M \in \mathbb{R}^{d\times k}$ satisfies $M^\top M = I_k$ (orthogonality), and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k)$. We analyze the impact of this projection on the two components of $\mathbf{g}$ separately.

(a) For the orthogonal component $\mathbf{g}_\perp$:

By definition, $\mathbf{g}_\perp$ lies in the orthogonal complement of $\mathcal{S}$, implying it is orthogonal to every column vector in M. Therefore:

$$M^\top \mathbf{g}_\perp = \mathbf{0}$$

Substituting $\mathbf{g}_\perp$ into the update rule yields:

$$\begin{aligned}\tilde{\mathbf{g}}_\perp &= \mathbf{g}_\perp - M\Lambda(M^\top \mathbf{g}_\perp) \\ &= \mathbf{g}_\perp - M\Lambda(\mathbf{0}) \\ &= \mathbf{g}_\perp\end{aligned}$$

Consequently, gradient components orthogonal to the GPM basis remain unchanged.

(b) For the parallel component $\mathbf{g}_\parallel$:

Since $\mathbf{g}_\parallel \in \mathcal{S}$, $\mathbf{g}_\parallel$ lies strictly within the subspace spanned by M. By the property of orthogonal projection operators, projecting $\mathbf{g}_\parallel$ back onto this subspace results in the vector itself:

$$MM^\top \mathbf{g}_\parallel = \mathbf{g}_\parallel$$

Substituting $\mathbf{g}_\parallel$ into the update rule defined in Equation 6:

$$\tilde{\mathbf{g}}_\parallel = \mathbf{g}_\parallel - M\Lambda M^\top \mathbf{g}_\parallel$$

Table 4: Performance of LLaMA3, GPT2-XL, and GPT-J under selected hyperparameter settings.

Model	Hyperparameters				Performance Metrics					
	α	γ_{th}	r	τ	Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑
LLaMA3	0.5	0.7	0.3	0.7	88.05	98.75	91.28	76.98	623.59	30.35
	0.5	0.7	0.4	0.7	87.85	98.75	91.15	76.61	623.10	30.36
	0.5	0.7	0.2	0.6	87.82	98.85	91.52	76.24	620.81	30.38
	0.5	0.7	0.2	0.5	87.36	99.20	92.28	74.49	618.99	30.12
	0.5	0.7	0.2	0.4	87.35	98.95	93.02	74.12	621.35	30.59
GPT2-XL	0.5	0.7	0.5	0.5	85.82	99.70	94.38	69.78	605.32	39.49
	0.5	0.7	0.2	0.5	85.48	99.50	94.50	69.14	602.34	39.47
	0.5	0.7	0.5	0.7	85.79	99.60	94.12	69.90	610.82	40.19
GPT-J	0.5	0.7	0.5	0.5	90.39	99.80	96.18	78.30	618.59	40.72
	0.5	0.7	0.5	0.7	90.37	99.80	95.98	78.38	619.32	40.99
	0.5	0.7	0.3	0.7	90.36	99.80	96.25	78.17	618.93	41.00
	0.5	0.7	0.4	0.7	90.43	99.85	96.05	78.43	619.77	41.11
	0.5	0.7	0.4	0.6	90.35	99.85	96.15	78.20	619.34	40.84

Utilizing the property $MM^\top \mathbf{g}_\parallel = \mathbf{g}_\parallel$ to replace the first term, we obtain:

$$\begin{aligned}\tilde{\mathbf{g}}_\parallel &= MM^\top \mathbf{g}_\parallel - M\Lambda M^\top \mathbf{g}_\parallel \\ &= (M - M\Lambda)M^\top \mathbf{g}_\parallel \\ &= M(I_k - \Lambda)M^\top \mathbf{g}_\parallel\end{aligned}$$

This formulation confirms that the gradient component parallel to the memory subspace is scaled by the factor $(I_k - \Lambda)$.

C More Experimental Results

C.1 Hyperparameter Sensitivity Analysis and Robustness Verification

Our proposed method relies on four key hyperparameters: the energy threshold γ_{th}, the scaling factor α, the core basis ratio r, and the minimum scaling lower bound τ. Given the vast combinatorial space and the dependency of optimal settings on model architecture and task characteristics, we adopted a heuristic tuning strategy rather than an exhaustive search. The performance of partial parameters for GPT-J, LLaMA3, and GPT2-XL is shown in Table 4. In this section, we provide qualitative selection guidelines based on the functional role of each parameter, followed by a rigorous assessment of the method's "Plug-and-Play" capability to demonstrate its robustness against parameter variations.

C.1.1 Qualitative Analysis and Selection Guidelines

Threshold γ_{th}: This parameter determines the energy retention rate of the SVD, directly dictating the size of the gradient projection memory M. A higher γ_{th} incorporates more basis vectors, offering stronger protection against catastrophic forgetting; however, it may also introduce noise and overconstrain the model parameters. If the edited model exhibits low efficacy or generalization, we recommend lowering γ_{th} to discard less informative basis vectors. Conversely, if specificity is insufficient, increasing γ_{th} can enforce stricter constraints.

Ratio r: This parameter controls the proportion of core basis vectors that receive full protection. A high r value causes the method to approximate hard orthogonal projection. A decline in generalization suggests the feasible parameter space is too restricted; in such cases, reducing r is the most effective remedy, as it permits scaled updates along a broader range of directions.

Minimum Scaling Lower Bound τ: This parameter sets the floor for projection weights applied to non-core basis vectors. A lower τ allows for larger gradient updates along these basis directions. If the model suffers from significant forgetting, increasing τ ensures that even lower-ranked historical knowledge is adequately preserved.

Scaling Factor α: This parameter adjusts the curvature of the cumulative importance distribution. Empirically, our method is relatively robust to α,

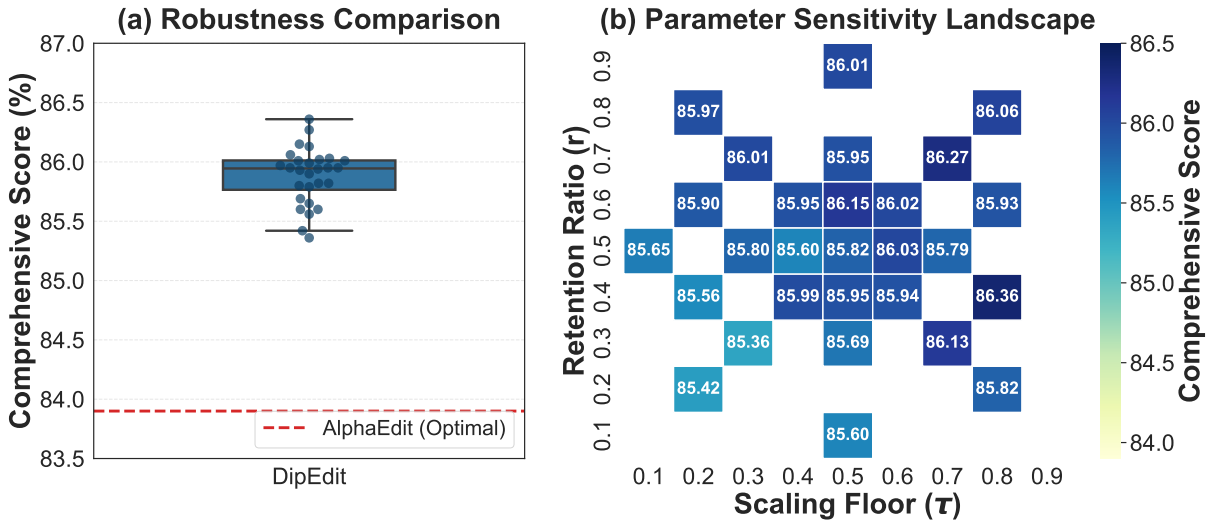

Figure 7: (a) The box plot illustrates the score distribution of DipEdit across 28 random hyperparameter configurations ($r, \tau \in [0.1, 0.9]$). The red dashed line denotes the optimal score of the strongest baseline, AlphaEdit. Notably, even the worst-performing configuration of DipEdit significantly outperforms the optimally tuned baseline. The gray shaded area (0.2%) denotes the concentration of the performance distribution. (b) Parameter sensitivity heatmap for Retention Ratio (r) and Scaling Floor (τ). White cells represent untested configurations; all colored cells (tested points) achieve a Comprehensive Score surpassing 85.0, consistently exceeding the baseline score (83.90).

which primarily serves to normalize the scale of singular values.

Recommended Configuration. In our experiments, we prioritized generalization while maintaining competitive retention capabilities. We observed that setting $\alpha = 0.5$ and $\gamma_{th} = 0.7$ consistently yields robust performance. However, we suggest fine-tuning r and τ according to the specific model architecture to achieve optimal results.

C.1.2 Plug-and-Play Capability and Stability Assessment

To address potential concerns regarding the model's sensitivity to hyperparameter fine-tuning in continuous knowledge editing tasks, we rigorously evaluated the "Plug-and-Play" capability of DipEdit. Given that γ_{th} and α have demonstrated strong robustness across different architectures, we fixed these parameters for this experiment and focused our sensitivity analysis on the two critical hyperparameters: the core basis ratio r and the scaling lower bound τ. We performed a grid search on the GPT2-XL model, using stratified sampling to select 28 representative parameter combinations covering the $[0.1, 0.9]$ interval. The experimental results, illustrated in Figure 7, yield the following conclusions:

Wide Safe Operating Zone. As shown in Figure 7 (a), the composite scores of all parameter configurations tested in our grid search significantly outperform the optimal baseline of AlphaEdit, which underwent full parameter tuning. Even in the worst-case configuration, DipEdit achieved a high score of 85.36, realizing a 1.46% performance advantage over the baseline. This indicates that DipEdit possesses an exceptionally high performance floor and is resilient to parameter fluctuations.

Superiority without Fine-tuning. As depicted in Figure 7 (b), the performance surface of DipEdit exhibits smooth and continuous characteristics, void of steep performance cliffs. This implies that while theoretical optimal configurations may vary slightly across architectures, a generic balanced setting can achieve performance far exceeding existing methods. Unless pursuing extreme marginal gains, DipEdit requires no complex heuristic search. This result strongly validates DipEdit's reliable plug-and-play capability in practical applications, greatly reducing deployment complexity.

C.2 Comparison with Parameter-Preserving Methods

Although DipEdit is fundamentally a parameter-modifying approach, we benchmark it against leading memory-based editing methods to provide a comprehensive performance assessment. Specifically, we selected SERAC (Mitchell et al., 2022b),

Table 5: Comparison of DipEdit with existing methods on the sequential model editing task. *Score*, *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Score, Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact						ZsRE		
		Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
Pre-edited		12.87	7.85	10.58	89.48	635.23	24.14	36.99	36.34	31.89
SERAC	Llama3	66.12	71.21	61.05	66.90	615.72	20.77	67.75	33.96	22.17
GRACE	Llama3	67.98	96.72	50.14	72.23	620.43	23.79	93.58	1.03	31.86
MELO	Llama3	62.28	65.29	58.58	63.36	608.98	22.18	25.18	24.14	30.36
DipEdit	Llama3	**87.35**	**98.95**	**93.05**	**74.12**	**621.35**	**30.59**	**94.80**	**91.65**	**32.76**
Pre-edited		30.36	22.23	24.34	78.53	626.64	31.88	22.19	31.30	24.15
SERAC	GPT2-XL	64.32	72.25	58.18	64.06	595.35	27.35	92.17	36.57	20.67
GRACE	GPT2-XL	68.23	98.88	50.05	**72.07**	**620.21**	28.53	94.33	1.59	**27.63**
MELO	GPT2-XL	62.21	72.62	53.63	63.25	588.57	23.58	93.54	45.25	23.45
DipEdit	GPT2-XL	**85.82**	**99.70**	**94.38**	69.78	605.32	**39.49**	**95.80**	**88.06**	27.06
Pre-edited		23.52	16.22	18.56	83.11	621.81	29.74	26.32	25.79	27.42
SERAC	GPT-J	68.49	82.28	58.31	68.98	615.92	28.65	92.37	38.21	25.17
GRACE	GPT-J	68.56	96.50	50.10	74.42	**620.56**	31.55	96.54	0.40	24.78
RELO	GPT-J	67.77	78.29	60.52	66.80	610.82	24.31	82.24	32.88	26.65
DipEdit	GPT-J	**90.43**	**99.85**	**96.05**	**78.43**	619.77	**41.11**	**99.68**	**96.44**	**28.35**

Table 6: Comparison of DipEdit with existing methods on the sequential model editing task. *Score*, *Eff.*, *Gen.*, *Spe.*, *Flu.* and *Consis.* denote Score, Efficacy, Generalization, Specificity, Fluency and Consistency, respectively. The best results are highlighted in bold, while the second-best results are underlined.

Method	Model	Counterfact						ZsRE		
		Score↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Consis.↑	Eff.↑	Gen.↑	Spe.↑
MEMIT	phi-1.5	46.97	55.71	56.58	35.41	368.57	19.79	54.41	52.47	20.98
RECT	phi-1.5	49.87	58.19	58.92	38.46	362.94	19.88	55.15	53.64	18.58
AlphaEdit	phi-1.5	60.11	70.79	65.12	48.96	399.47	25.98	70.02	63.19	20.69
DipEdit	phi-1.5	**81.93**	**97.90**	**86.05**	**67.66**	**636.23**	**34.80**	**95.76**	**85.76**	**22.93**
MEMIT	qwen2.5	61.71	68.60	66.18	52.84	193.12	7.78	24.75	23.80	11.91
RECT	qwen2.5	67.86	77.05	73.75	56.60	502.28	8.81	58.43	56.14	27.73
AlphaEdit	qwen2.5	76.07	90.87	84.33	60.33	570.76	31.36	87.08	82.89	33.58
DipEdit	qwen2.5	**82.57**	**97.61**	**94.84**	**64.34**	**606.73**	**36.38**	**90.86**	**86.84**	**34.84**

GRACE (Hartvigsen et al., 2023), and MELO (Yu et al., 2024) as representative baselines. The comparative results are presented in Table 5, from which we derive the following conclusions:

First, DipEdit exhibits superior generalization capabilities compared to memory-based alternatives. While baselines like GRACE achieve high efficacy, they struggle with semantic variations—dropping to 1.03% generalization on Llama3 (ZsRE)—whereas DipEdit maintains robust performance (>91%), indicating effective semantic learning rather than rote memorization. Second, regarding generation quality, DipEdit achieves the highest consistency scores across all architectures. However, we acknowledge that memory-based methods occasionally retain a marginal advantage in Specificity and Fluency. This is attributed to their non-destructive nature, where original parameters remain frozen to strictly preserve pre-existing behaviors. In contrast, DipEdit modifies model weights to intrinsically integrate knowledge; while this incurs a slight trade-off in strict locality, it yields a substantially more favorable balance between plasticity and generalization.

C.3 Evaluation on Qwen2.5 and Phi-1.5

To broaden the scope of our evaluation and verify architectural generalization, we conducted supple-

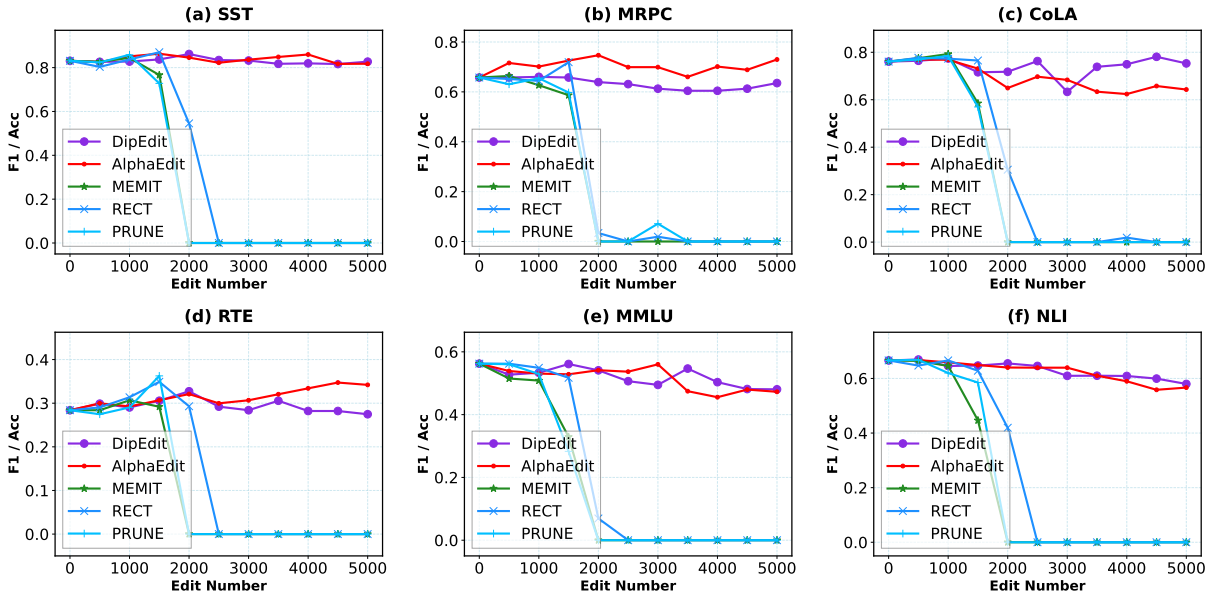

Figure 8: F1 scores of the LLaMA3 (8B) model after editing across six general capability benchmarks (i.e., SST, MRPC, CoLA, RTE, MMLU, and NLI).

Edit Count	# Bases	Mem. (KB)	Time (s)
1	1	6.26	5.40
10	10	62.58	53.96
100	98	613.27	539.97
500	366	2290.36	3543.05
1000	494	3091.36	6460.09
1500	565	3535.66	10652.61
2000	625	3911.13	15062.02

Table 7: Efficiency analysis of DipEdit on GPT2-XL. We report the number of stored bases (**# Bases**), the memory overhead (**Mem.**), and the total editing duration (**Time**) across different edit counts.

mentary experiments on the Qwen2.5 (Qwen et al., 2025) and Phi-1.5 (Li et al., 2023) models. The detailed empirical results are summarized in Table 6.

These results demonstrate that DipEdit consistently outperforms baseline methods across all metrics on both the CounterFact and ZsRE datasets. We attribute this superior performance to the proposed scalable gradient projection mechanism. By effectively mitigating the trade-off between stability and plasticity, this mechanism not only rigorously preserves previously edited knowledge but also retains sufficient parameter space for subsequent updates, thereby simultaneously enhancing model specificity and generalization. Notably, the most substantial performance gain is observed on the Phi-1.5 model, which achieves an improvement of approximately 36%. This highlights the adaptability of our approach to smaller, more parameter-efficient architectures. Collectively, these findings confirm the robustness of DipEdit across diverse LLM architectures and its capability to deliver high-quality continuous knowledge editing.

C.4 General Capability Tests

To further explore the upper limits of model stability and verify the long-term impact of continuous editing, we extended the evaluation scope from the initial 3,000 steps to a more challenging 5,000 sequential edits, with the results illustrated in Figure 8.

Overall, DipEdit and AlphaEdit are the only methods capable of withstanding this extended threshold without suffering from model collapse. Notably, DipEdit exhibits a significant advantage in linguistic stability and semantic preservation. On the CoLA dataset, for instance, DipEdit maintains a high accuracy of 75.31% after 5,000 edits, substantially outperforming AlphaEdit (64.30%). This indicates that DipEdit better preserves fundamental grammatical knowledge and linguistic "naturalness" during extensive parameter updates.

However, we observe a performance trade-off in specific reasoning-centric subtasks. While DipEdit maintains performance levels consistent with the original model baseline on MRPC and RTE, AlphaEdit achieves higher absolute scores in the later stages of the experiment. This result suggests

Table 8: Ablation study on GPT2-XL modules. "Multi", "Dual", and "Proj" denote multi-template input, dual importance, and the projection mechanism, respectively.

	Multi	Dual	Proj	Counterfact				ZsRE		
				Score↑	Eff.↑	Gen.↑	Spe.↑	Eff.↑	Gen.↑	Spe.↑
①	✗	✗	✗	83.90	99.50	93.95	66.39	94.81	86.11	25.88
②	✓	✗	✗	83.92	99.25	**95.18**	65.93	93.49	86.76	25.86
③	✗	✗	✓	85.40	99.65	92.45	**70.04**	95.32	83.51	**27.06**
④	✓	✗	✓	85.51	99.55	93.04	69.97	95.47	86.48	26.99
⑤	✗	✓	✓	85.59	**99.75**	93.58	69.74	94.79	86.96	27.01
⑥	✓	✓	✓	**85.82**	99.70	94.38	69.78	**95.80**	**88.06**	**27.06**

that the parameter shift induced by AlphaEdit may inadvertently enhance performance on certain classification tasks, while DipEdit prioritizes fidelity to the original data distribution.

C.5 Efficiency Analysis

Memory Overhead. To protect edited knowledge, our method caches basis vectors during the editing process, incurring additional memory overhead computed as (number of basis vectors $\times$ hidden dimension $\times$ 4)/1024 KB. As shown in Table 7, editing 2000 facts in GPT2-XL (hidden dimension 1600) requires only 3911.13 KB $\approx$ 3.82 MB of extra memory. According to Equation 8 and Figure 3, editing a single fact adds at most one basis—or even none. Consequently, even for the largest models, such as GPT-J and LLaMA3, the maximum overhead for 2000 basis vectors is only $4096 \times 2000 \times 4 \div 1024 = 32000$ KB ≈ 31.25 MB, which is negligible in typical tens-of-GB GPU memory environments.

Editing Efficiency. As shown in Table 7, the average time to edit one fact in GPT2-XL is only 7.5 seconds.Although larger hidden dimensions increase gradient computation complexity and thus per-edit time in bigger models. Nonetheless, the approach remains practical for real-world applications.

C.6 Module Ablation Study

To rigorously evaluate the contribution of each component and verify that the performance improvement primarily stems from the proposed Dual Importance Scaled Gradient Projection mechanism rather than data augmentation (i.e., the Improved Template Context Mechanism), we conducted a series of component-level ablation studies on the GPT2-XL model. The results are presented in Table 8.

Dominant Role of the Core Algorithm: One of the primary objectives of this section is to decouple the effects of the dual importance projection mechanism from prompt engineering. Comparing Row ② (Improved Template Context only) with Row ⑤ (Dual Importance mechanism only), we observe that while multi-template inputs significantly enhance generalization relative to Baseline ① by providing diverse inputs to improve robustness for edited knowledge, they offer no protection for previously edited facts. Consequently, the improvement in comprehensive performance is minimal. In contrast, employing the dual importance mechanism to protect edited knowledge results in a significant boost in comprehensive performance. This confirms that the substantial performance leap is driven by the algorithmic innovation of DipEdit, with prompt augmentation serving only an auxiliary role.

Dual Importance Mechanism Independently Restores Generalization: While the projection mechanism is crucial for task specificity, strict orthogonal projection (as shown in Row ③) severely constrains gradient expressiveness, causing generalization to drop from the baseline's 93.95 to 92.45. However, even without the multi-template strategy, merely adding the dual importance mechanism (as shown in Row ⑤) restores generalization to 93.58, with only a minor decrease in specificity. This validates our theoretical hypothesis that this "soft" scaled constraint successfully balances model plasticity and stability. Although it does not fully surpass the baseline of 93.95 because even scaled projection imposes some gradient constraints, further integrating the multi-template strategy (as shown in Row ⑥) elevates generalization to optimal levels. Nevertheless, the fundamental performance

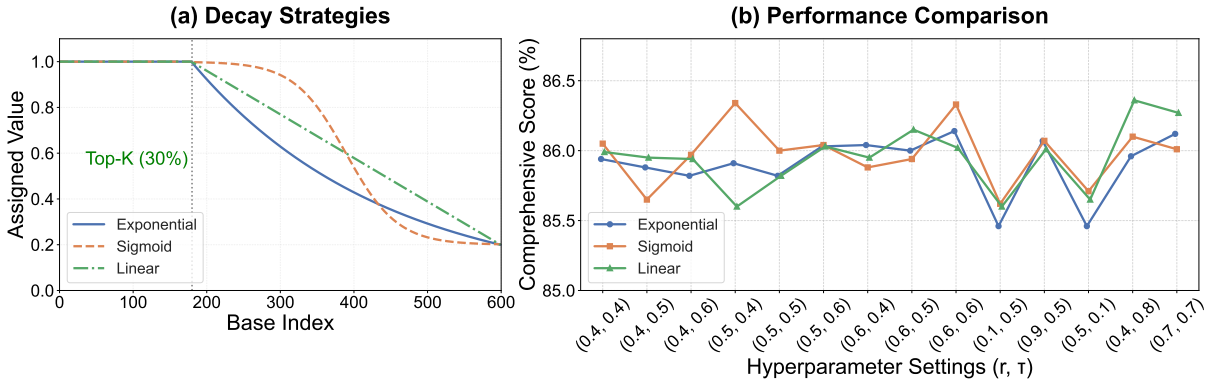

Figure 9: (a)Schematic illustration of the three importance decay strategies: Linear, Exponential, and Sigmoid. (b) Performance comparison of Linear, Exponential, and Sigmoid decay functions across 14 diverse hyperparameter configurations. The x-axis represents different combinations of (Retention Ratio r, Scaling Floor τ).

capability is determined by the dual importance mechanism itself.

Summary: The multi-template mechanism provides a robust enhancement to model performance, while the dual importance mechanism successfully balances plasticity and stability. However, DipEdit achieves SOTA performance primarily due to its innovative gradient scaling strategy, with template augmentation acting merely as an auxiliary optimization measure.

C.7 Robustness to Decay Function Shapes

While singular value distributions typically exhibit heavy-tailed characteristics (as shown in Figure 3), we posit that the efficacy of DipEdit stems primarily from the dual-importance mechanism itself rather than fitting a specific spectral shape. To validate this and address concerns regarding the heuristic nature of the linear decay strategy, we compared three distinct decay functions for projection importance (λ_{proj}): (1) Linear Decay (Ours); (2) Exponential Decay, mimicking the theoretical distribution of singular values; and (3) Sigmoid Decay.We conducted experiments across 14 representative hyperparameter configurations of (r, τ). As illustrated in Figure 9, the results demonstrate:

Performance Consistency: The three decay functions show no statistically significant difference across all tested configurations. The average comprehensive score for the linear strategy (85.95%) differs from Sigmoid (85.98%) and Exponential (85.90%) strategies by less than 0.1%.

Robustness across Hyperparameters: Whether under conservative (e.g., $r = 0.9$) or aggressive (e.g., $\tau = 0.1$) settings, the linear decay consistently tracks the performance of the theoretically more complex exponential decay.These findings confirm the high robustness of DipEdit. Following the Principle of Parsimony, we select the linear decay strategy to avoid introducing additional shape-controlling parameters (e.g., decay rates or inflection points), thereby minimizing tuning complexity and overfitting risks while maintaining optimal performance.

D Dataset Visualizations

In this section, we present comprehensive visualizations of the datasets utilized in our experiments to elucidate the paradigm of knowledge editing for readers unfamiliar with the task. We first provide illustrative examples sampled from the standard CounterFact and ZsRE benchmarks. Subsequently, we detail the construction criteria for the City and Language datasets, which were curated specifically to evaluate model robustness in homogeneous sequential editing scenarios. Corresponding visualizations of these domain-specific subsets are also provided to demonstrate their structural consistency.

D.1 CounterFact Visualization

Representative samples from the CounterFact dataset are shown in Figure 10. These examples illustrate the standard edit request format alongside paraphrase, neighborhood, attribute, and generation prompts used for comprehensive evaluation.

D.2 ZsRE Visualization

Figure 11 presents illustrative examples from the ZsRE dataset. This visualization highlights the question-answering format employed for zero-shot

```
{
    "case_id": 0,
    "pararel_idx": 2796,
    "requested_rewrite": {
        "prompt": "The mother tongue of {} is",
        "relation_id": "P103",
        "target_new": {
            "str": "English",
            "id": "Q1860"
        },
        "target_true": {
            "str": "French",
            "id": "Q150"
        },
        "subject": "Danielle Darrieux"
    },
    "paraphrase_prompts": [
        "Shayna does this and Yossel goes still and dies. Danielle Darrieux, a native",
        "An album was recorded for Capitol Nashville but never released. Danielle Darrieux spoke the language"
    ],
    "neighborhood_prompts": [
        "The mother tongue of Léon Blum is",
        "The native language of Montesquieu is",
        "François Bayrou, a native",
        "The native language of Raymond Barre is",
        "Michel Rocard is a native speaker of",
        "Jacques Chaban-Delmas is a native speaker of",
        "The native language of François Bayrou is",
        "Maurice Genevoix, speaker of",
        "The mother tongue of François Bayrou is",
        "Melchior de Vogüé, speaker of"
    ],
    "attribute_prompts": [
        "J. R. R. Tolkien is a native speaker of",
        "The mother tongue of Douglas Adams is",
        "The mother tongue of Paul McCartney is",
        "Elvis Presley is a native speaker of",
        "Barack Obama, speaker of",
        "Douglas Adams, speaker of",
        "Meryl Streep, a native",
        "George Orwell spoke the language",
        "George Washington, a native",
        "Michael Jackson, a native"
    ],
    "generation_prompts": [
        "Danielle Darrieux's mother tongue is",
        "Where Danielle Darrieux is from, people speak the language of",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux was born in",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux's mother tongue is",
        "Danielle Darrieux was born in",
        "Where Danielle Darrieux is from, people speak the language of",
        "Danielle Darrieux was born in",
        "Danielle Darrieux was born in"
    ]
}
```


Figure 10: A Samples of the CounterFact dataset.

relation extraction tasks, including the subject, prompt, and target answers.

D.3 Homogeneous Fact Datasets: Construction and Visualization

The City and Language datasets were derived from the CounterFact benchmark. To construct robust homogeneous datasets for sequential editing evaluation, we aimed for a scale of approximately 2,000 edits per category. However, we observed that the sample volume for individual relation IDs within CounterFact was insufficient to meet this threshold.

To address this limitation, we aggregated samples from semantically aligned relations that share similar factual attributes. Specifically:

- **City Dataset:** Aggregates facts involving location-associated relations, including P140 (religion/location), P17 (country), and P495

```
[
    {
        "subject": "Watts Humphrey",
        "src": "What university did Watts Humphrey attend?",
        "pred": "Trinity College",
        "rephrase": "What university did Watts Humphrey take part in?",
        "alt": "University of Michigan",
        "answers": [
            "Illinois Institute of Technology"
        ],
        "loc": "nq question: who played desmond doss father in hacksaw ridge",
        "loc_ans": "Hugo Weaving",
        "cond": "Trinity College >> University of Michigan || What university did Watts Humphrey attend?"
    },
    {
        "subject": "Ramalinaceae",
        "src": "Which family does Ramalinaceae belong to?",
        "pred": "Ramalinales",
        "rephrase": "What family are Ramalinaceae?",
        "alt": "Lamiinae",
        "answers": [
            "Lecanorales"
        ],
        "loc": "nq question: types of skiing in the winter olympics 2018",
        "loc_ans": "Downhill",
        "cond": "Ramalinales >> Lamiinae || Which family does Ramalinaceae belong to?"
    },
    {
        "subject": "Denny Herzig",
        "src": "What role does Denny Herzig play in football?",
        "pred": "midfielder",
        "rephrase": "What's Denny Herzig's role in football?",
        "alt": "winger",
        "answers": [
            "defender"
        ],
        "loc": "nq question: where does aarp fall on the political spectrum",
        "loc_ans": "non-partisan",
        "cond": "midfielder >> winger || What role does Denny Herzig play in football?"
    },
    {
        "subject": "Call the Doctor",
        "src": "What artist created Call the Doctor?",
        "pred": "Riders in the Sky",
        "rephrase": "Which artist created Call the Doctor?",
        "alt": "The X-Files",
        "answers": [
            "Sleater-Kinney"
        ],
        "loc": "nq question: who sang nice day for a white wedding",
        "loc_ans": "Billy Idol",
        "cond": "Riders in the Sky >> The X-Files || What artist created Call the Doctor?"
    }
]
```


Figure 11: A Samples of the ZsRE dataset.

(country of origin).

- **Language Dataset:** Combines linguistic-related relations, including P103 (native language), P364 (original language of work), and P1412 (languages spoken, written, or signed).

Data instances were extracted sequentially from the original CounterFact corpus based on these relation identifiers. This process yielded a final compilation of 2,584 entries for the Language dataset and 2,195 entries for the City dataset. Examples are shown in Figures 12 and 13.

```
[
    {
        "case_id": 1,
        "pararel_idx": 19501,
        "requested_rewrite": {
            "prompt": "The official religion of {} is",
            "relation_id": "P140",
            "target_new": {
                "str": "Islam",
                "id": "Q432"
            },
            "target_true": {
                "str": "Christianity",
                "id": "Q5043"
            },
            "subject": "Edwin of Northumbria"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    },
    {
        "case_id": 2,
        "pararel_idx": 6791,
        "requested_rewrite": {
            "prompt": "{}, which is located in",
            "relation_id": "P17",
            "target_new": {
                "str": "Sweden",
                "id": "Q34"
            },
            "target_true": {
                "str": "Spain",
                "id": "Q29"
            },
            "subject": "Autonomous University of Madrid"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    },
    {
        "case_id": 3,
        "pararel_idx": 10743,
        "requested_rewrite": {
            "prompt": "{}, created in",
            "relation_id": "P495",
            "target_new": {
                "str": "Sweden",
                "id": "Q34"
            },
            "target_true": {
                "str": "India",
                "id": "Q668"
            },
            "subject": "Shree Pundalik"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    }
]
```


Figure 12: A Samples of the City dataset.

```
[
    {
        "case_id": 1,
        "pararel_idx": 2796,
        "requested_rewrite": {
            "prompt": "The mother tongue of {} is",
            "relation_id": "P103",
            "target_new": {
                "str": "English",
                "id": "Q1860"
            },
            "target_true": {
                "str": "French",
                "id": "Q150"
            },
            "subject": "Danielle Darrieux"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    },
    {
        "case_id": 5,
        "pararel_idx": 9913,
        "requested_rewrite": {
            "prompt": "The original language of {} was",
            "relation_id": "P364",
            "target_new": {
                "str": "Tamil",
                "id": "Q5885"
            },
            "target_true": {
                "str": "Icelandic",
                "id": "Q294"
            },
            "subject": "The Icelandic Dream"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    },
    {
        "case_id": 8,
        "pararel_idx": 18404,
        "requested_rewrite": {
            "prompt": "The language used by {} is",
            "relation_id": "P1412",
            "target_new": {
                "str": "Italian",
                "id": "Q652"
            },
            "target_true": {
                "str": "Hebrew",
                "id": "Q9288"
            },
            "subject": "Gilad Atzmon"
        },
        "...": "< ... Paraphrase, Neighborhood, Attribute prompts omitted ... >"
    }
]
```


Figure 13: A Samples of the Language dataset.