

AdaPE: Adaptive Editing Strength for Batch Knowledge Editing in Large Language Models

Jinjin Qian^{1*} Yunqi Mi^{1*} Zeyu Song¹ Guoshuai Zhao^{1†} Xueming Qian²

¹School of Software Engineering, Xi'an Jiaotong University

²School of Information and Communication Engineering, Xi'an Jiaotong University
guoshuai.zhao@xjtu.edu.cn

Abstract

Large language models (LLMs) encode vast amounts of knowledge in their parameters, but these facts often require precise post-training rectification as they become outdated or incorrect over time. Traditional batch editing methods typically apply a uniform update strength across all instances. However, this uniform approach ignores the inherent variation in editing difficulty, leading to a fundamental trade-off between correction success and model stability. To overcome this limitation, we introduce AdaPE (Adaptive Pressure Editing), a parameter-free adaptive editing framework that dynamically adjusts the update behavior for each specific instance. Specifically, we derive a closed-form adaptive update rate based on the ratio of residual energy to spatial sensitivity, which strictly quantifies the intrinsic injection difficulty. This formulation adaptively limits the update strength for difficult facts to preserve model locality, while ensuring sufficient knowledge injection for easier ones. Extensive experiments on three benchmarks and two prominent LLMs demonstrate the superiority of our approach in consistently circumventing the fixed-strength bottleneck, achieving a new state-of-the-art balance between editing efficacy and model integrity. Our code is available at <https://github.com/skyandkibo/adape>.

1 Introduction

Large Language Models (LLMs) encode vast amounts of knowledge and are widely used in knowledge-intensive NLP tasks (Touvron et al., 2023; Kamaloo et al., 2023; Lai and Nissim, 2024; Chen, 2023). However, as real-world knowledge evolves, fully retraining a model to reflect new facts is often prohibitively expensive. Knowledge editing (KE) offers a more efficient alternative by updating targeted knowledge in a trained model

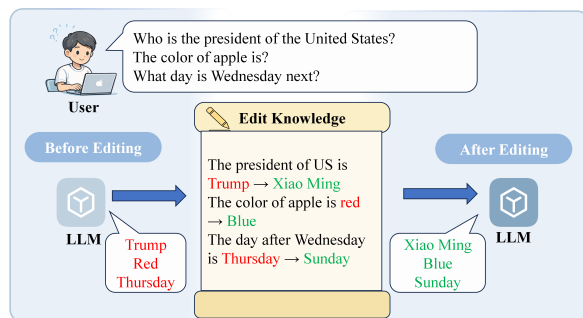


Figure 1: Illustration of batch knowledge editing in LLMs.

without full retraining, while preserving unrelated behaviors (Wang et al., 2024c; Li et al., 2024; Tan et al., 2024). Existing KE methods can be broadly grouped into two categories: parametric editing methods, which directly modify a small subset of model parameters, and parameter-preserving methods, which integrate external modules without altering the original backbone parameters. Among parametric editing methods, locate-and-edit has emerged as a representative paradigm, including methods such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), due to its efficiency in direct model updating. In practice, knowledge updates often arrive in batches rather than as isolated facts, as illustrated in Fig. 1. However, existing locate-and-edit methods typically use a single fixed editing strength α for all requests in a batch (Meng et al., 2023; Park et al., 2025; Fang et al., 2025), which conflicts with the heterogeneous nature of real-world knowledge updates.

To characterize this heterogeneity, we introduce the notion of editing pressure P_i , a representation-derived metric that will later be formalized as a normalized ratio between residual energy and key-space sensitivity. For the empirical analysis in Fig. 2(a), we partition edit requests into ten equal-frequency bins according to P_i , ensuring comparable sample sizes across pressure regimes. Similar

*Equal contribution.

†Corresponding author.

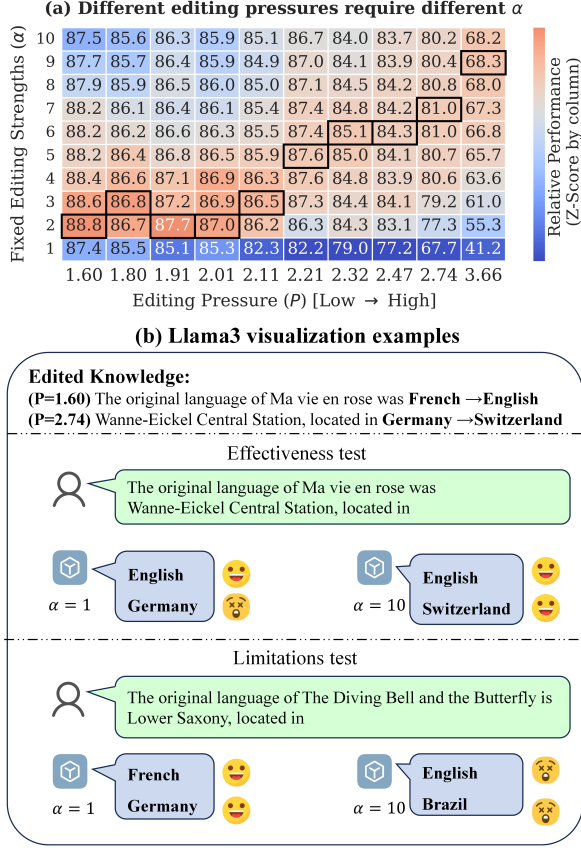


Figure 2: Editing performance under different editing pressures on Llama3-8B-Instruct over CounterFact (MLP layers 4–8). **(a)** Grid search over fixed α across pressure bins. Colors show column-wise Z-score normalization, while numbers denote absolute scores; black boxes mark the best-performing fixed α in each bin. **(b)** Qualitative examples under the fixed- α setting.

to standard batch knowledge editing benchmarks MEMIT, we evaluate update quality using a composite F1-score that harmonically balances editing efficacy and locality preservation. As shown in Fig. 2(a), the best-performing fixed editing strength varies across pressure bins rather than remaining constant. This observation highlights the central motivation of our work. Different edit requests require different α values, and therefore no single global editing strength can adequately serve a heterogeneous batch. Fig. 2(b) provides qualitative examples of this mismatch under the fixed- α setting. Together, these observations suggest that the appropriate editing strength should be determined at the instance level, rather than being imposed uniformly across the entire batch.

Motivated by this observation, we propose AdaPE (Adaptive Pressure Editing), a dynamic framework for batch knowledge editing that assigns instance-specific editing strengths according to the

relative pressure of each request. Instead of enforcing a single uniform intervention across a heterogeneous batch, AdaPE derives a pressure-aware update allocation mechanism and integrates it seamlessly into the batch editing solver. In this way, AdaPE assigns more appropriate update strengths to heterogeneous edit requests, thereby achieving a better balance between editing efficacy and locality preservation. Through this adaptive editing strategy, AdaPE enables batch knowledge updates to be performed in a more precise and robust manner. Our contributions as follows:

- We analyze a key limitation of batch knowledge editing. Heterogeneous edit requests require different editing strengths, making a fixed α inherently suboptimal for a batch.
- We introduce the editing pressure P_i as a representation-derived measure for each edit request, and propose AdaPE, an adaptive batch editing framework that assigns instance-specific editing strengths according to P_i for more fine-grained model updates.
- Extensive experiments across multiple benchmarks and LLMs show that AdaPE consistently outperforms fixed-strength baselines, achieving better overall editing performance while preserving strong locality and fluency.

2 Preliminaries

2.1 Knowledge Editing Objective

Following the locate-and-edit paradigm, we formulate batch knowledge editing as a weighted constrained least-squares problem. For a batch of N_t target edit requests $\{(k_t^i, m_t^i)\}_{i=1}^{N_t}$, the goal is to find an updated weight matrix $\hat{W}$ that minimizes

$$\min_{\hat{W}} \left(\sum_{i=1}^{N_t} \alpha_i \|\hat{W} k_t^i - m_t^i\|^2 + \sum_{j=1}^{N_p} \|\hat{W} k_p^j - m_p^j\|^2 \right), \quad (1)$$

where α_i denotes the editing strength assigned to the i -th edit request. Standard locate-and-edit methods such as MEMIT correspond to the uniform case $\alpha_i = 1$ for all $i \in \{1, \dots, N_t\}$. The second term represents the stability constraint on N_p preserved facts, ensuring that the updated weight matrix $\hat{W}$ (typically the W_{out} of an MLP) retains pre-existing knowledge and prevents catastrophic forgetting.

2.2 Closed-Form Solution for Batch Editing

Let W_0 denote the original weight matrix and define the parameter perturbation as $\Delta W \triangleq \hat{W} - W_0$. By stacking the target keys and memory vectors into matrices K_t and M_t , the corresponding residual matrix can be written as $R = M_t - W_0 K_t$. Defining the diagonal editing-strength matrix $A = \text{diag}(\alpha_1, \dots, \alpha_{N_t})$, the weighted optimization problem in Equation 1 admits the following closed-form solution:

$$\Delta W = R A K_t^T (C_p + K_t A K_t^T)^{-1}, \quad (2)$$

where $R = [r_t^1 | \dots | r_t^{N_t}]$ is the batch residual matrix. The covariance matrix C_p is typically formulated as $C_p = \lambda \Sigma$, where $\Sigma = \mathbb{E}_{k_p}[k_p k_p^T]$ represents the unscaled covariance of pre-existing knowledge and λ is a global stability coefficient that scales the preservation covariance. Standard methods such as MEMIT can be viewed as the special case with uniform editing strength, i.e., $A = I$, together with a fixed global coefficient λ (e.g., $\lambda = 15000$ for Llama models). A detailed derivation of this solution is provided in Appendix A.

3 Methodology

An overview of AdaPE is shown in Fig. 3. Starting from the limitations of fixed-strength editing, our method derives editing pressure from edit density and uses it to determine an adaptive update rule and the final parameter-free batch solver.

3.1 Motivation

While Equation 2 provides a general framework, existing solvers suffer from the rigidity of the uniform editing-strength assumption. To analyze the update behavior at the instance level, we examine the case of a single edit request ($N_t = 1$) to isolate the intrinsic interaction between the residual and the parameter space. Under this condition, we define the Effective Update Rate $\rho_i \in [0, 1]$ as the fraction of the initial residual r_t^i absorbed by the update, which quantifies the injection efficiency for a specific fact. This rate can be written as

$$\rho_i \triangleq \frac{\|\Delta W_i k_t^i\|}{\|r_t^i\|} = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}, \quad (3)$$

where $\tilde{S}_i \triangleq (k_t^i)^T \Sigma^{-1} k_t^i$ denotes the spatial sensitivity of the key direction (detailed derivation in Appendix B).

To understand how the fixed-strength formulation affects different facts, we must mathematically evaluate two critical metrics determined by ρ_i . The first metric is the remaining absolute error E_{rem}^i , which governs editing efficacy. The second metric is the required parameter perturbation measured under the knowledge covariance C_p , which governs model locality. Based on the analytical solution, these two metrics are strictly derived as (proof in Appendix C):

$$E_{rem}^i = (1 - \rho_i) \|r_t^i\|, \quad (4)$$

$$\|\Delta W_i\|_{C_p} = \rho_i \sqrt{\lambda} \frac{\|r_t^i\|}{\sqrt{\tilde{S}_i}}. \quad (5)$$

Intuitively, a smaller E_{rem}^i indicates better target injection, whereas a smaller $\|\Delta W_i\|_{C_p}$ indicates better preservation of model locality.

Equations 3 through 5 provide the mathematical evidence for the fundamental dilemma of the uniform setting ($\alpha_i = 1$). Under the uniform setting together with a fixed global λ , ρ_i is determined solely by $\tilde{S}_i$ and remains completely blind to the residual magnitude $\|r_t^i\|$. Consider a challenging fact with a massive initial residual $\|r_t^i\|$. Under the fixed $\alpha_i = 1$, Equation 4 shows that the constrained update rate ρ_i leaves behind a substantial absolute error E_{rem}^i , leading directly to an injection failure. Conversely, if one attempts to rectify this failure by globally increasing the editing strength (i.e., forcing a higher ρ_i via α_i), Equation 5 demonstrates that the perturbation norm $\|\Delta W_i\|_{C_p}$ will scale linearly with the massive $\|r_t^i\|$. Existing solvers fail to balance these two extremes because locking $\alpha_i = 1$ deprives the model of the degree of freedom required to adaptively modulate ρ_i according to the coupled interaction between residual magnitude and spatial sensitivity.

3.2 Scale-Invariant Adaptive Objective

The standard batch formulation relies on a uniform editing strength together with a fixed global stability coefficient, which prevents the solver from adapting the update allocation to the pressure of each edit request. As shown in Section 3.1, this leads to a fundamental dilemma between efficacy and locality at the instance level. Directly optimizing a dynamic diagonal matrix A within the batch formulation is mathematically intractable due to the non-linear coupling in $(C_p + K_t A K_t^T)^{-1}$. We therefore adopt a decoupling strategy: we first analyze a single edit request (k_t^i, r_t^i) to derive an intrinsic

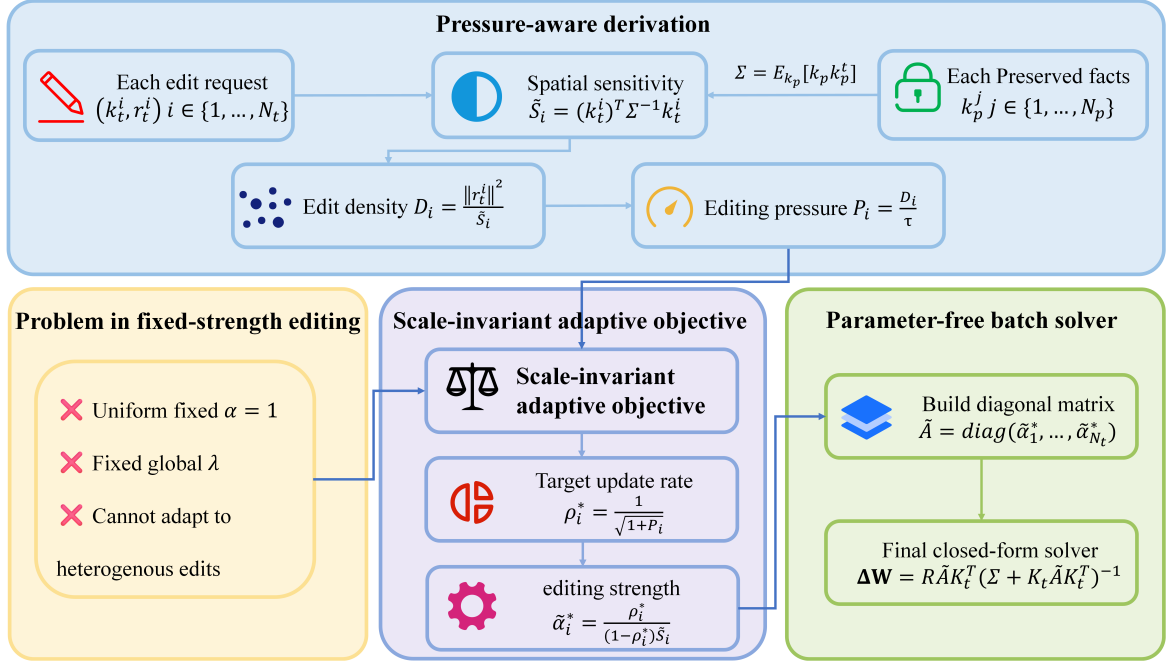


Figure 3: Overview of AdaPE.

sic adaptive rule, and then integrate the resulting instance-specific allocation into the batch solver.

To address this limitation, we propose the following scale-invariant adaptive objective:

$$\min_{\Delta W_i} \mathcal{L}_i = \underbrace{\left(1 - \frac{\|\Delta W_i k_t^i\|^2}{\|r_t^i\|^2}\right)^2}_{\text{Residual assimilation penalty}} + \underbrace{\gamma \|\Delta W_i\|_{C_p}^2 \frac{\|\Delta W_i k_t^i\|^2}{\|r_t^i\|^2}}_{\text{Effort-aware locality penalty}}, \quad (6)$$

where γ is a global regularization coefficient. Recalling the Effective Update Rate

$$\rho_i \triangleq \frac{\|\Delta W_i k_t^i\|}{\|r_t^i\|},$$

the objective simplifies to

$$\mathcal{L}_i = (1 - \rho_i^2)^2 + \gamma \|\Delta W_i\|_{C_p}^2 \rho_i^2. \quad (7)$$

This formulation overcomes the main limitation of standard quadratic objectives. Conventional solvers impose a static locality penalty, which forces a rigid trade-off between editing efficacy and locality. As a result, the update may either fail to inject the target fact or introduce excessive perturbation to the model. In contrast, our objective is built on two complementary principles. The first term measures the fraction of residual energy that

remains unassimilated, thereby directly encouraging factual injection. The second term scales the locality penalty by the actual assimilation effort, so that more aggressive updates are penalized more heavily. In addition, normalizing both terms by the initial residual energy $\|r_t^i\|^2$ makes the objective scale-invariant across layers.

3.3 Closed-Form Derivation of the Adaptive Update Rate

To find the exact theoretical optimum, we substitute Equation 5 into Equation 7. This gives

$$\mathcal{L}_i(\rho_i^2) = (1 - \rho_i^2)^2 + \gamma \lambda \frac{\|r_t^i\|^2}{\tilde{S}_i} \rho_i^4. \quad (8)$$

The objective is now written entirely as a function of the energy variable ρ_i^2 . Since α_i influences the objective only through the ρ_i , it is more convenient to optimize $\mathcal{L}_i$ directly with respect to ρ_i^2 rather than α_i . Setting

$$\frac{\partial \mathcal{L}_i}{\partial (\rho_i^2)} = -2(1 - \rho_i^2) + 2\gamma \lambda \frac{\|r_t^i\|^2}{\tilde{S}_i} \rho_i^2 = 0$$

yields the exact optimal update rate:

$$\rho_i^* = \frac{1}{\sqrt{1 + \gamma \lambda \frac{\|r_t^i\|^2}{\tilde{S}_i}}}. \quad (9)$$

This result shows that the optimal update behavior is governed by the coupled quantity $\|r_t^i\|^2 / \tilde{S}_i$. We

therefore define the Effective Edit Density as

$$\mathcal{D}_i \triangleq \frac{\|r_t^i\|^2}{\tilde{S}_i}. \quad (10)$$

Equation 9 thus provides an instance-aware control law for the target update rate. For difficult facts with high density $\mathcal{D}_i$, it suppresses ρ_i^* to protect model locality. Conversely, for simple facts with low density, ρ_i^* approaches 1, promoting successful factual injection.

3.4 From Edit Density to Editing Pressure

While the closed-form solution resolves the instance-level trade-off, the coupled hyperparameter $\gamma\lambda$ does not generalize well across models and layers. This is because the absolute magnitude of the edit density $\mathcal{D}_i$ can vary by several orders of magnitude across different LLM architectures and layer representations. To obtain a parameter-free adaptive rule, we normalize the penalty term relative to the typical edit density of the target layer. Specifically, we anchor the expected penalty over the data distribution to unit scale:

$$\mathbb{E}[\gamma\lambda\mathcal{D}_i] = 1. \quad (11)$$

This condition ensures that, for an average edit request in a given layer, the efficacy and locality terms are balanced at the same order of magnitude. Solving this expectation gives $\gamma\lambda = 1/\tau$, where τ denotes the layer-wise expected edit density:

$$\tau \triangleq \mathbb{E}[\mathcal{D}_i]. \quad (12)$$

Using τ , we formally define the editing pressure of each edit request as the following:

$$\mathcal{P}_i \triangleq \gamma\lambda\mathcal{D}_i = \frac{\mathcal{D}_i}{\tau}. \quad (13)$$

The adaptive update rate therefore becomes

$$\rho_i^* = \frac{1}{\sqrt{1 + \mathcal{P}_i}}. \quad (14)$$

This mechanism admits a simple interpretation: the editing pressure of an edit is determined not by its absolute density, but by its density relative to the typical edit density of the layer. As $\mathcal{P}_i$ increases, the target update rate decreases smoothly according to the inverse-square-root law in Equation 14, leading to a more conservative update behavior. In this way, the adaptive rule adjusts the target update behavior according to the relative editing pressure of each request while maintaining a balanced trade-off between efficacy and locality.

3.5 Parameter-Free Batch Solver Implementation

Given the target update rate ρ_i^* , the corresponding scaled editing strength can be obtained by equating $\rho_i^* = 1/\sqrt{1 + \mathcal{P}_i}$ with its structural relation in Equation 3. We then construct the diagonal scaled editing-strength matrix

$$\tilde{A} = \text{diag}(\tilde{\alpha}_1^*, \dots, \tilde{\alpha}_{N_t}^*), \tilde{\alpha}_i^* \triangleq \alpha_i^*/\lambda,$$

where the scaled editing strength is

$$\tilde{\alpha}_i^* = \frac{1}{\tilde{S}_i} \left(\frac{1}{\sqrt{1 + \mathcal{P}_i} - 1} \right). \quad (15)$$

The algebraic derivation is provided in Appendix D. This expresses the scaled editing strength as the inverse spatial geometry $1/\tilde{S}_i$, modulated by the editing pressure $\mathcal{P}_i$.

Substituting $\tilde{A}$ into Equation 2, together with $C_p = \lambda\Sigma$, yields

$$\Delta W = R\tilde{A}K_t^T(\lambda\Sigma + K_t\tilde{A}K_t^T)^{-1}.$$

The global stability coefficient λ then cancels out from both the numerator and the inverse term, and the final parameter-free update rule becomes

$$\Delta W = R\tilde{A}K_t^T(\Sigma + K_t\tilde{A}K_t^T)^{-1}. \quad (16)$$

In practice, the procedure is straightforward. For a batch of edit requests, we first compute the edit density $\mathcal{D}_i$ for each instance and then estimate the layer-wise average τ . Using the editing pressure $\mathcal{P}_i = \mathcal{D}_i/\tau$, we populate the matrix $\tilde{A}$ via Equation 15, and finally compute the weight update with Equation 16. The resulting solver requires no manual tuning of a global stability coefficient.

4 Experiments

4.1 Experimental Settings

Models, Datasets, and Baselines. We conduct experiments on two widely used LLMs, Llama3-8B (Grattafiori et al., 2024) and Qwen2.5-7B (Hui et al., 2024). We evaluate all methods on three standard knowledge editing benchmarks: ZsRE (Meng et al., 2022), CounterFact (Meng et al., 2022), and Wiki-CounterFact (Wang et al., 2024b). As baselines, we consider a range of representative KE methods, including FT (Rawat et al., 2021), ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), CORE (Park et al., 2025), and AlphaEdit (Fang et al., 2025). For locate-and-edit

Table 1: Performance of batch-editing algorithms across various models and datasets.

Algorithm	Wiki-counterfact				Counterfact				ZsRE		
	Eff ↑	Por ↑	Loc ↑	Flu ↑	Eff ↑	Gen ↑	Spe ↑	Flu ↑	Eff ↑	Gen ↑	Spe ↑
Model: Llama3-8b-instruct											
Pre-edit	0.00	21.86	57.63	6.18	0.34	0.43	89.19	6.29	1.16	1.06	38.44
FT (Rawat et al., 2021)	0.11	21.63	44.27	2.66	45.93	9.7	37.32	3.59	0.44	0.49	16.33
ROME (Meng et al., 2022)	0.95	21.64	44.26	5.55	4.5	0.93	47.40	6.06	0.00	0.00	0.32
MEMIT (Meng et al., 2023)	64.24	29.71	51.34	6.16	86.47	42.36	84.07	6.25	77.25	69.54	43.43
CORE (Park et al., 2025)	63.05	41.94	45.22	6.00	84.46	53.96	63.59	5.93	57.07	53.34	37.00
AlphaEdit (Fang et al., 2025)	53.03	27.95	51.99	6.15	78.21	37.84	83.67	6.25	72.03	64.65	44.88
AdaPE (Ours)	74.85	32.16	50.32	6.14	92.27	47.20	83.20	6.24	81.56	73.67	42.97
Model: Qwen2.5-7b-instruct											
Pre-edit	0.00	21.77	47.52	6.22	0.39	0.46	85.62	6.25	0.95	0.42	38.29
FT (Rawat et al., 2021)	0.00	21.61	44.26	3.04	67.02	14.02	3.30	4.16	0.02	0.04	5.41
ROME (Meng et al., 2022)	15.85	25.25	44.36	5.45	0.16	0.11	50.34	6.03	0.00	0.00	1.16
MEMIT (Meng et al., 2023)	67.93	43.66	44.49	6.17	85.58	59.33	59.65	6.18	60.74	55.35	34.76
CORE (Park et al., 2025)	28.00	32.38	44.41	6.19	58.36	39.24	57.27	6.19	20.62	18.73	20.32
AlphaEdit (Fang et al., 2025)	60.19	42.93	44.52	6.19	24.61	14.58	55.70	6.12	7.30	6.40	11.44
AdaPE (Ours)	70.56	44.27	44.49	6.17	86.78	59.75	59.64	6.18	62.44	56.70	35.97

baselines such as MEMIT, CORE, and AlphaEdit, the editing hyperparameters are fixed globally across edit requests, following their standard batch-editing setup. In contrast, AdaPE assigns instance-specific editing strengths according to the editing pressure of each request. Full details of the datasets and baselines are provided in Appendix E.

Evaluation Metrics. We follow the standard evaluation protocols used for each benchmark. For CounterFact and ZsRE, we report efficacy (Eff.), generalization (Gen.), and specificity (Spe.), together with fluency (Flu.) when applicable (Meng et al., 2022). For Wiki-CounterFact, we adopt its benchmark-specific evaluation setting and report efficacy (Eff.), portability (Por.), locality (Loc.), and fluency (Flu.) (Wang et al., 2024b). Our experiments focus primarily on the batch editing scenario, where multiple edit requests are processed simultaneously to evaluate the scalability and efficiency of each method. Full details of the evaluation metrics are provided in Appendix G.

All experiments are conducted on two 48GB vGPU machines. Baseline methods are implemented using the widely adopted EasyEdit toolkit. We follow the hyperparameter settings reported in the original papers whenever available; otherwise, we use the default configurations recommended by EasyEdit. Minor implementation adjustments are made when necessary to ensure compatibility with our batch editing evaluation framework.

4.2 Comparison of Editing Effects

Table 1 reports the main batch editing results on CounterFact (10,000 samples), ZsRE (10,000 samples), and Wiki-CounterFact (839 samples) under Llama3-8B-Instruct and Qwen2.5-7B-Instruct. Overall, AdaPE consistently improves editing efficacy across most settings, while maintaining competitive locality-related performance. This indicates that adaptive update allocation is more effective than fixed-strength editing in the batch setting.

The trade-off is particularly clear on Wiki-CounterFact. On Llama3-8B-Instruct, AdaPE achieves the best efficacy (**74.85**), substantially outperforming MEMIT (64.24) and AlphaEdit (53.03), while also improving portability to 32.16 over MEMIT (29.71) and AlphaEdit (27.95). Although its locality score (50.32) is slightly lower than AlphaEdit (51.99) and MEMIT (51.34), the gain in efficacy and portability is much larger than the locality loss. Compared with CORE, AdaPE improves both efficacy (74.85 vs. 63.05) and locality (50.32 vs. 45.22), while sacrificing part of the portability advantage (32.16 vs. 41.94). This suggests that AdaPE reaches a more favorable overall trade-off by accepting a small loss on one metric for a much larger gain on another.

A similar pattern appears on CounterFact and ZsRE. On Llama3-8B-Instruct, AdaPE improves CounterFact efficacy from 86.47 to **92.27** over MEMIT while keeping specificity comparable (83.20 vs. 84.07). On ZsRE, it increases efficacy

from 77.25 to **81.56** and generalization from 69.54 to **73.67**, with only a modest decrease in specificity. Similar trends hold on Qwen2.5-7B-Instruct, where AdaPE achieves the best efficacy on all three benchmarks and also improves generalization on CounterFact and ZsRE over MEMIT, while keeping locality-related metrics close to the strongest baselines. AdaPE’s fluency remains stable across methods, around 6.1 on all reported settings.

These results support the main motivation of AdaPE. Fixed-strength methods apply the same update magnitude to edit requests with highly different editing pressure, which often leads either to insufficient editing or to unnecessary collateral perturbation. In contrast, AdaPE adapts the update allocation according to the editing pressure of each request, producing a more favorable trade-off between efficacy and locality in batch editing.

4.3 Fixed-Strength Trade-off Analysis

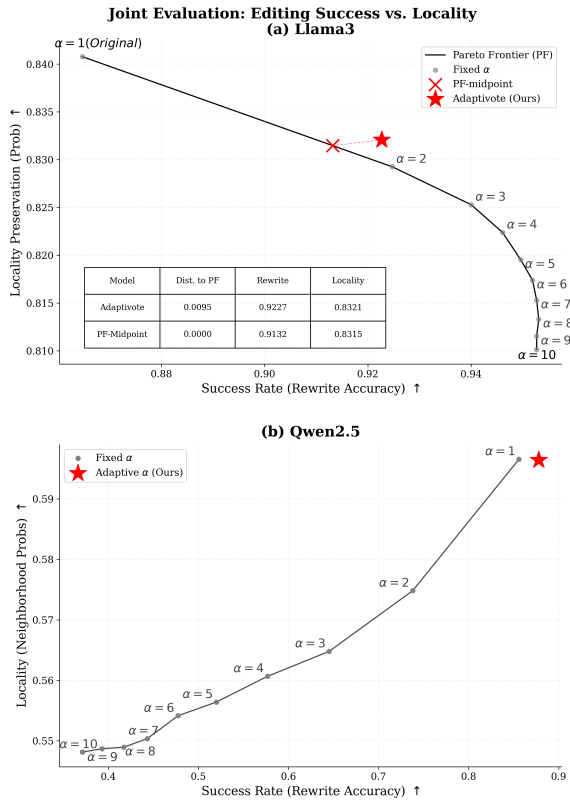


Figure 4: Joint evaluation on Llama3-8B-Instruct and Qwen2.5-7B-Instruct under different fixed α values.

To further understand the effect of fixed editing strength, we compare different fixed values of α with the adaptive strategy used in AdaPE. Figures 4 (a) and 4 (b) show the joint relationship between rewrite accuracy and locality on Llama3-

8B-Instruct and Qwen2.5-7B-Instruct, respectively.

On Llama3-8B-Instruct, the fixed- α configurations form a clear Pareto frontier: higher rewrite accuracy can only be achieved at the expense of lower locality. Following prior work on Pareto-based joint evaluation, we additionally consider the midpoint on the Pareto frontier (PF-midpoint) as a representative balanced operating point (Rampisela et al., 2025). The PF-midpoint achieves rewrite accuracy 0.9132 and locality 0.8315, whereas AdaPE reaches 0.9227 and 0.8321, respectively. Thus, AdaPE not only lies beyond the fixed- α frontier, but also outperforms the PF-midpoint on both dimensions simultaneously.

The behavior on Qwen2.5-7B-Instruct is different. Instead of tracing a clean frontier over the full range of α , excessively large fixed strengths lead to simultaneous degradation in both editing success and locality. This indicates that, under a single shared strength, the update can move beyond a useful trade-off regime and enter an over-editing region where stronger interventions no longer improve rewriting but instead damage overall behavior. In contrast, AdaPE avoids this fixed-strength failure pattern and reaches a more favorable joint operating point.

Taken together, these results show that a single shared α is not a globally reliable operating rule. Depending on the model and the edit distribution, fixed-strength editing may either induce a rigid Pareto frontier or enter a joint degradation regime. By assigning instance-specific update allocations, AdaPE avoids both limitations and achieves a better balance between rewrite accuracy and locality.

4.4 Impact of Editing Pressure on Performance

To better understand performance under different difficulty regimes, we stratify the evaluation set by editing pressure (P). Each edit request is assigned a pressure score, and the dataset is divided into four equal quartiles from Q1 (low pressure) to Q4 (high pressure). Compared with the finer-grained pressure-bin analysis in the background section, this quartile-based view provides a more compact comparison across methods. Table 2 reports the performance of AdaPE together with two representative fixed-strength baselines, $\alpha = 1$ and $\alpha = 10$. In addition to Efficacy and Locality, we also report their joint F1-score.

As editing pressure increases from Q1 to Q4, the performance of all methods declines. The two

Table 2: Performance stratified by editing pressure (P). The dataset is divided into four pressure quartiles (Q1–Q4). Best results in each quartile are highlighted in **bold**. AdaPE consistently achieves the highest F1-score, indicating a better balance between efficacy and locality.

Pressure Interval	α	Efficacy (%)	Locality (%)	F1-score (%)
Q1 (1.7380)	$\alpha = 1$	96.5	84.1	86.3
	$\alpha = 10$	98.8	81.4	86.6
	AdaPE	98.7	83.4	88.0
Q2 (2.0362)	$\alpha = 1$	94.0	83.9	83.9
	$\alpha = 10$	98.1	80.7	85.5
	AdaPE	97.8	83.1	86.8
Q3 (2.2972)	$\alpha = 1$	89.6	84.4	80.3
	$\alpha = 10$	96.7	81.4	83.7
	AdaPE	95.0	83.6	84.7
Q4 (3.0620)	$\alpha = 1$	65.7	83.9	58.6
	$\alpha = 10$	80.4	80.5	68.3
	AdaPE	77.6	82.7	69.0

fixed-strength baselines show a clear trade-off between efficacy and locality. The $\alpha = 1$ setting better preserves locality, keeping it around 84% across quartiles, but its efficacy drops sharply under high pressure, reaching 65.7% in Q4. By contrast, $\alpha = 10$ achieves the highest efficacy in every quartile, but does so at the cost of lower locality, which stays around 81%. AdaPE achieves a better overall balance than both fixed-strength baselines: its efficacy remains close to that of $\alpha = 10$, while its locality stays much closer to that of $\alpha = 1$. As a result, AdaPE obtains the best F1-score in all pressure quartiles. This advantage is most evident in Q4, where AdaPE reaches 69.0%, outperforming $\alpha = 1$ (58.6%) and $\alpha = 10$ (68.3%). The same pattern also appears in the lower-pressure regimes, where AdaPE maintains efficacy comparable to $\alpha = 10$ while preserving better locality.

These results further support the motivation of AdaPE. Fixed-strength updates cannot adapt to heterogeneous editing pressure and therefore tend to favor either efficacy or locality depending on the chosen α . In contrast, AdaPE adjusts the update allocation for each request, producing a more stable trade-off between factual injection and neighborhood preservation across pressure regimes.

5 Related Work

Knowledge editing methods can be broadly divided into two categories: parameter-preserving methods and parametric editing methods.

Parameter-preserving methods. These methods avoid directly modifying the backbone parameters,

and instead store updated knowledge in auxiliary memories or external modules. SERAC (Mitchell et al., 2022b) maintains an explicit edit memory with separate retrieval and counterfactual models. GRACE (Hartvigsen et al., 2023) builds a codebook-like memory at intermediate layers to cache edited representations. IKE (Zheng et al., 2023) performs edits through retrieved demonstration prompts, while MELO (Yu et al., 2024) uses a neuron-indexed dynamic LoRA (Hu et al., 2022) module as an auxiliary memory. WISE (Wang et al., 2024a) further introduces a dual-memory architecture to coordinate parametric and side memories.

Parametric editing methods. These methods directly modify model parameters to inject new knowledge, often through meta-learning or locate-and-edit strategies. Meta-learning-based methods learn to generate editing updates via auxiliary networks. KE (De Cao et al., 2021) formulates editing as a meta-learning problem, while MEND (Mitchell et al., 2022a) predicts low-rank parameter updates from gradients. InstructEdit (Zhang et al., 2024) further extends this line with instruction-based training across diverse editing tasks. Another common strategy is locate-and-edit, which identifies and updates a small set of parameters associated with the target fact. ROME (Meng et al., 2022) performs localized single-fact updates, and MEMIT (Meng et al., 2023) extends this idea to batch editing. CORE (Park et al., 2025) edits facts by extending key representations in parameter space. AlphaEdit (Fang et al., 2025) improves robustness by constraining perturbations to reduce interference.

6 Conclusion

In this work, we identify a key limitation of batch knowledge editing: fixed-strength updates cannot accommodate the heterogeneous difficulty of different edits, resulting in an inherent trade-off between editing success and locality preservation. To address this issue, we propose AdaPE, a parameter-free adaptive editing framework that assigns instance-specific update strengths according to the intrinsic difficulty of each fact. Extensive experiments across multiple models and benchmarks show that AdaPE consistently achieves stronger editing performance while maintaining model stability, effectively surpassing the fixed-strength trade-off and establishing a more robust solution for batch knowledge editing.

Acknowledgments

This work is in part funded by the National Natural Science Foundation of China (Grant No. 62372364) and Shaanxi Province Natural Science Foundation (Grant No. 2026JC-YXQN-213).

Limitations

Despite the strong theoretical foundation and empirical performance of AdaPE, there are still several directions worth further exploration. First, like most locate-and-edit methods, AdaPE is currently evaluated mainly on structured factual associations (e.g., subject-relation-object triplets) that are typically mediated by feed-forward layers. Extending the adaptive mechanism to broader forms of model editing, such as logical reasoning patterns, multi-hop knowledge, or implicit procedural skills, would be an interesting direction for future work. Second, AdaPE focuses on adaptively controlling the magnitude of parameter updates through editing pressure, while following the update direction induced by the underlying regularized least-squares formulation. Although this design already yields a strong balance between efficacy and locality, it may be less flexible in cases where the representations of different target facts are highly entangled in the model space. Exploring how to jointly adapt both update magnitude and direction could further improve factual disentanglement and editing precision in such settings.

Ethics Statement

While knowledge editing techniques provide an effective approach for correcting inaccurate or outdated knowledge in large language models, it is important to recognize the potential risks associated with their misuse. In particular, such techniques could be exploited to deliberately inject misleading, biased, or harmful information into a model by directly modifying its internal knowledge representations. This capability raises concerns about the integrity and trustworthiness of language model systems, as manipulated knowledge may influence model outputs in unintended ways. In applications involving information dissemination or decision support, such risks could undermine the reliability, fairness, and safety of the system. Therefore, the development and deployment of knowledge editing methods should be accompanied by careful oversight, transparent evaluation procedures, and appropriate safeguards to mitigate potential misuse.

References

- Huajun Chen. 2023. Large knowledge model: Perspectives and challenges. *arXiv preprint arXiv:2312.02706*.
- Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6491–6506.
- Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. 2025. [Alphaedit: Null-space constrained knowledge editing for language models](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- 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*.
- Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. *Advances in Neural Information Processing Systems*, 36:47934–47959.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Binyuan Hui, Jian Yang, Zeyu Cui, Jiayi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, and 1 others. 2024. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.
- Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, pages 5591–5606.
- Huiyuan Lai and Malvina Nissim. 2024. A survey on automatic generation of figurative language: From rule-based systems to large language models. *ACM Computing Surveys*, 56(10):1–34.
- Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

- Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. [Mass-editing memory in a transformer](#). In *ICLR*.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. [Fast model editing at scale](#). In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.
- Haewon Park, Gyubin Choi, Minjun Kim, and Yohan Jo. 2025. Context-robust knowledge editing for language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10360–10385.
- Theresia Veronika Rampisela, Tuukka Ruotsalo, Maria Maistro, and Christina Lioma. 2025. Joint evaluation of fairness and relevance in recommender systems with pareto frontier. In *Proceedings of the ACM on Web Conference 2025*, pages 1548–1566.
- Ankit Singh Rawat, Chen Zhu, Daliang Li, Felix Yu, Manzil Zaheer, Sanjiv Kumar, and Srinadh Bhojanapalli. 2021. Modifying memories in transformer models (ICML), volume 2020.
- Chenmien Tan, Ge Zhang, and Jie Fu. 2024. [Massive editing for large language models via meta learning](#). In *The Twelfth International Conference on Learning Representations*.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Hua-jun Chen. 2024a. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797.
- Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, and 1 others. 2024b. Easyedit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 82–93.
- Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024c. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.
- Lang Yu, Qin Chen, Jie Zhou, and Liang He. 2024. Melo: Enhancing model editing with neuron-indexed dynamic lora. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19449–19457.
- Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. 2024. [Instructedit: Instruction-based knowledge editing for large language models](#). In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 6633–6641. ijcai.org.
- 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? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4862–4876.

A Derivation of the Closed-Form Solution for Weighted Batch Editing

This section derives the closed-form solution of the weighted batch editing objective in Equation 1. For a batch of target edit requests $\{(k_t^i, m_t^i)\}_{i=1}^{N_t}$ and preserved facts $\{(k_p^j, m_p^j)\}_{j=1}^{N_p}$, the objective is

$$\min_{\hat{W}} \left(\sum_{i=1}^{N_t} \alpha_i \left\| \hat{W} k_t^i - m_t^i \right\|^2 + \sum_{j=1}^{N_p} \left\| \hat{W} k_p^j - m_p^j \right\|^2 \right). \quad (17)$$

Let W_0 denote the original weight matrix and define

$$\Delta W \triangleq \hat{W} - W_0, \hat{W} = W_0 + \Delta W. \quad (18)$$

For the target batch, define

$$K_t = [k_t^1 | k_t^2 | \dots | k_t^{N_t}], \quad (19)$$

$$M_t = [m_t^1 | m_t^2 | \dots | m_t^{N_t}],$$

and the residual matrix

$$R = M_t - W_0 K_t = [r_t^1 | r_t^2 | \dots | r_t^{N_t}], \quad (20)$$

where

$$r_t^i = m_t^i - W_0 k_t^i. \quad (21)$$

Let the diagonal editing-strength matrix be

$$A = \text{diag}(\alpha_1, \dots, \alpha_{N_t}). \quad (22)$$

Substituting $\hat{W} = W_0 + \Delta W$ into the target term gives

$$\begin{aligned} \hat{W} k_t^i - m_t^i &= (W_0 + \Delta W) k_t^i - m_t^i \\ &= \Delta W k_t^i - (m_t^i - W_0 k_t^i) \\ &= \Delta W k_t^i - r_t^i. \end{aligned} \quad (23)$$

Therefore,

$$\sum_{i=1}^{N_t} \alpha_i \left\| \hat{W} k_t^i - m_t^i \right\|^2 = \sum_{i=1}^{N_t} \alpha_i \left\| \Delta W k_t^i - r_t^i \right\|^2. \quad (24)$$

In matrix form, this term can be written as

$$\sum_{i=1}^{N_t} \alpha_i \left\| \Delta W k_t^i - r_t^i \right\|^2 = \text{Tr} \left((\Delta W K_t - R) A (\Delta W K_t - R)^\top \right). \quad (25)$$

For the preservation term, substituting $\hat{W} = W_0 + \Delta W$ yields

$$\hat{W} k_p^j - m_p^j = (W_0 k_p^j - m_p^j) + \Delta W k_p^j. \quad (26)$$

Following the standard covariance-based approximation used in locate-and-edit methods, we replace the preservation loss with a quadratic surrogate induced by the key covariance:

$$\sum_{j=1}^{N_p} \left\| \hat{W} k_p^j - m_p^j \right\|^2 \approx \text{Tr} \left(\Delta W C_p \Delta W^\top \right) + \text{const}, \quad (27)$$

where

$$C_p = \lambda \Sigma, \Sigma = \mathbb{E}_{k_p} [k_p k_p^\top]. \quad (28)$$

After replacing the preservation term with this surrogate and dropping terms independent of ΔW , the objective becomes

$$\mathcal{L}(\Delta W) = \text{Tr} \left((\Delta W K_t - R) A (\Delta W K_t - R)^\top \right) + \text{Tr} \left(\Delta W C_p \Delta W^\top \right). \quad (29)$$

Expanding the first term in Equation 29, we have

$$\begin{aligned} \mathcal{L}(\Delta W) &= \text{Tr}(\Delta W K_t A K_t^\top \Delta W^\top) \\ &\quad - 2 \text{Tr}(\Delta W K_t A R^\top) + \text{Tr}(R A R^\top) \\ &\quad + \text{Tr}(\Delta W C_p \Delta W^\top). \end{aligned} \quad (30)$$

Using the matrix derivative identities

$$\frac{\partial}{\partial X} \text{Tr}(X B X^\top) = 2X B, \text{ for symmetric } B, \quad (31)$$

and

$$\frac{\partial}{\partial X} \text{Tr}(X M) = M^\top, \quad (32)$$

the derivative of $\mathcal{L}(\Delta W)$ with respect to ΔW is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \Delta W} &= 2 \Delta W K_t A K_t^\top - 2 R A K_t^\top \\ &\quad + 2 \Delta W C_p. \end{aligned} \quad (33)$$

Setting the derivative to zero gives

$$2 \Delta W K_t A K_t^\top - 2 R A K_t^\top + 2 \Delta W C_p = 0. \quad (34)$$

which is equivalent to

$$\Delta W K_t A K_t^\top + \Delta W C_p = R A K_t^\top. \quad (35)$$

Factoring out ΔW on the left-hand side gives

$$\Delta W (C_p + K_t A K_t^\top) = R A K_t^\top. \quad (36)$$

Assuming $C_p + K_t A K_t^\top$ is invertible, we obtain

$$\begin{aligned} \Delta W &= R A K_t^\top \\ &\quad \cdot (C_p + K_t A K_t^\top)^{-1}. \end{aligned} \quad (37)$$

Hence, the updated weight matrix is

$$\begin{aligned} \hat{W} &= W_0 + \Delta W \\ &= W_0 + R A K_t^\top (C_p + K_t A K_t^\top)^{-1}. \end{aligned} \quad (38)$$

B Derivation of the Effective Update Rate

This section derives the closed-form expression of the Effective Update Rate in Equation 3 under the single-edit setting. For a single edit request, Equation 2 reduces to

$$\Delta W_i = r_t^i \alpha_i (k_t^i)^T (C_p + \alpha_i k_t^i (k_t^i)^T)^{-1}, \quad (39)$$

where $r_t^i = m_t^i - W_0 k_t^i$. Using $C_p = \lambda \Sigma$, we rewrite Equation 39 as

$$\Delta W_i = r_t^i \alpha_i (k_t^i)^T (\lambda \Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1}. \quad (40)$$

We are interested in the actual injected update along the target key direction:

$$\Delta W_i k_t^i = r_t^i \alpha_i (k_t^i)^T (\lambda \Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} k_t^i. \quad (41)$$

Since the term multiplying r_t^i is scalar, define

$$\beta_i \triangleq \alpha_i (k_t^i)^T (\lambda \Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} k_t^i, \quad (42)$$

so that

$$\Delta W_i k_t^i = \beta_i r_t^i. \quad (43)$$

Therefore, the effective update rate is

$$\rho_i \triangleq \frac{\|\Delta W_i k_t^i\|}{\|r_t^i\|} = \frac{\|\beta_i r_t^i\|}{\|r_t^i\|} = |\beta_i|. \quad (44)$$

Under the standard assumptions $\alpha_i > 0$, $\lambda > 0$, and $\Sigma \succ 0$, we will show below that $\beta_i > 0$, hence

$\rho_i = \beta_i$. It therefore remains to compute β_i . Applying the Sherman–Morrison formula,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}, \quad (45)$$

with

$$A = \lambda\Sigma, u = \alpha_i k_t^i, v = k_t^i, \quad (46)$$

we obtain

$$\begin{aligned} & (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} \\ &= (\lambda\Sigma)^{-1} - \frac{(\lambda\Sigma)^{-1}(\alpha_i k_t^i)(k_t^i)^T(\lambda\Sigma)^{-1}}{1 + (k_t^i)^T(\lambda\Sigma)^{-1}(\alpha_i k_t^i)} \\ &= \frac{1}{\lambda}\Sigma^{-1} - \frac{\frac{\alpha_i}{\lambda^2}\Sigma^{-1}k_t^i(k_t^i)^T\Sigma^{-1}}{1 + \frac{\alpha_i}{\lambda}(k_t^i)^T\Sigma^{-1}k_t^i}. \end{aligned} \quad (47)$$

Substituting Equation 47 into β_i , and defining

$$\tilde{S}_i \triangleq (k_t^i)^T \Sigma^{-1} k_t^i, \quad (48)$$

we obtain

$$\beta_i = \frac{\alpha_i}{\lambda} \tilde{S}_i - \frac{\left(\frac{\alpha_i}{\lambda} \tilde{S}_i\right)^2}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}. \quad (49)$$

Let

$$x \triangleq \frac{\alpha_i}{\lambda} \tilde{S}_i. \quad (50)$$

Then

$$\beta_i = x - \frac{x^2}{1+x} = \frac{x(1+x) - x^2}{1+x} = \frac{x}{1+x}. \quad (51)$$

Substituting back gives

$$\beta_i = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}. \quad (52)$$

Because $\alpha_i > 0$, $\lambda > 0$, and $\tilde{S}_i > 0$, it follows that $\beta_i > 0$. Therefore, from Equation 44,

$$\rho_i = \frac{\|\Delta W_i k_t^i\|}{\|r_t^i\|} = \beta_i = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}, \quad (53)$$

which completes the derivation.

C Derivation of the Error and Perturbation Metrics

This section derives the two metrics in Equations 4 and 5 under the single-edit setting. For a single edit request, the closed-form update is

$$\Delta W_i = r_t^i \alpha_i (k_t^i)^T (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1}. \quad (54)$$

From the derivation of the Effective Update Rate, we have

$$\Delta W_i k_t^i = \rho_i r_t^i, \quad (55)$$

where

$$\rho_i = \frac{\|\Delta W_i k_t^i\|}{\|r_t^i\|} = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}, \tilde{S}_i \triangleq (k_t^i)^T \Sigma^{-1} k_t^i. \quad (56)$$

Under the assumptions $\alpha_i > 0$, $\lambda > 0$, and $\Sigma \succ 0$, we have $\tilde{S}_i > 0$, and therefore

$$0 < \rho_i < 1. \quad (57)$$

The residual after editing along the target fact is

$$r_{rem}^i \triangleq r_t^i - \Delta W_i k_t^i. \quad (58)$$

Substituting Equation 55 gives

$$r_{rem}^i = r_t^i - \rho_i r_t^i = (1 - \rho_i) r_t^i. \quad (59)$$

Therefore, the remaining absolute error is

$$E_{rem}^i \triangleq \|r_{rem}^i\| = \|(1 - \rho_i) r_t^i\| = |1 - \rho_i| \|r_t^i\|. \quad (60)$$

Since $0 < \rho_i < 1$, we obtain

$$E_{rem}^i = (1 - \rho_i) \|r_t^i\|. \quad (61)$$

Next, we derive the perturbation norm under the knowledge covariance. Define the covariance-weighted norm as

$$\|\Delta W_i\|_{C_p}^2 \triangleq \text{Tr}(\Delta W_i C_p \Delta W_i^T). \quad (62)$$

Using $C_p = \lambda\Sigma$ and Equation 54, we obtain

$$\begin{aligned} & \|\Delta W_i\|_{C_p}^2 \\ &= \text{Tr}(\Delta W_i \lambda\Sigma \Delta W_i^T) \\ &= \text{Tr} \left[r_t^i \alpha_i (k_t^i)^T (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} \right. \\ & \quad \cdot \lambda\Sigma \cdot (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} k_t^i \alpha_i (r_t^i)^T \left. \right]. \end{aligned} \quad (63)$$

Since all terms except r_t^i and $(r_t^i)^T$ are scalars, this becomes

$$\begin{aligned} \|\Delta W_i\|_{C_p}^2 &= \alpha_i^2 \|r_t^i\|^2 (k_t^i)^T (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} \\ & \quad \cdot \lambda\Sigma \cdot (\lambda\Sigma + \alpha_i k_t^i (k_t^i)^T)^{-1} k_t^i. \end{aligned} \quad (64)$$

Let

$$M_i \triangleq \lambda\Sigma + \alpha_i k_t^i (k_t^i)^T. \quad (65)$$

Then Equation 64 becomes

$$\|\Delta W_i\|_{C_p}^2 = \alpha_i^2 \|r_t^i\|^2 (k_t^i)^T M_i^{-1} \lambda \Sigma M_i^{-1} k_t^i. \quad (66)$$

Using

$$\lambda \Sigma = M_i - \alpha_i k_t^i (k_t^i)^T, \quad (67)$$

we have

$$\begin{aligned} & (k_t^i)^T M_i^{-1} \lambda \Sigma M_i^{-1} k_t^i \\ &= (k_t^i)^T M_i^{-1} (M_i - \alpha_i k_t^i (k_t^i)^T) M_i^{-1} k_t^i \\ &= (k_t^i)^T M_i^{-1} k_t^i - \alpha_i [(k_t^i)^T M_i^{-1} k_t^i]^2. \end{aligned} \quad (68)$$

From the definition of ρ_i ,

$$\rho_i = \alpha_i (k_t^i)^T M_i^{-1} k_t^i. \quad (69)$$

Hence

$$(k_t^i)^T M_i^{-1} k_t^i = \frac{\rho_i}{\alpha_i}. \quad (70)$$

Substituting Equation 70 into Equation 68, we obtain

$$\begin{aligned} (k_t^i)^T M_i^{-1} \lambda \Sigma M_i^{-1} k_t^i &= \frac{\rho_i}{\alpha_i} - \alpha_i \left(\frac{\rho_i}{\alpha_i} \right)^2 \\ &= \frac{\rho_i(1 - \rho_i)}{\alpha_i}. \end{aligned} \quad (71)$$

Therefore,

$$\|\Delta W_i\|_{C_p}^2 = \alpha_i^2 \|r_t^i\|^2 \cdot \frac{\rho_i(1 - \rho_i)}{\alpha_i} = \alpha_i \rho_i (1 - \rho_i) \|r_t^i\|^2. \quad (72)$$

Using the closed-form expression of ρ_i ,

$$\rho_i = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}, \quad (73)$$

we obtain

$$1 - \rho_i = \frac{1}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}. \quad (74)$$

Thus,

$$\frac{\rho_i}{1 - \rho_i} = \frac{\alpha_i}{\lambda} \tilde{S}_i, \quad (75)$$

which implies

$$\alpha_i (1 - \rho_i) = \frac{\lambda \rho_i}{\tilde{S}_i}. \quad (76)$$

Substituting Equation 76 into Equation 72 yields

$$\|\Delta W_i\|_{C_p}^2 = \rho_i \left(\frac{\lambda \rho_i}{\tilde{S}_i} \right) \|r_t^i\|^2 = \frac{\lambda \rho_i^2 \|r_t^i\|^2}{\tilde{S}_i}. \quad (77)$$

Since $\tilde{S}_i > 0$, taking the square root gives

$$\|\Delta W_i\|_{C_p} = \rho_i \sqrt{\lambda} \frac{\|r_t^i\|}{\sqrt{\tilde{S}_i}}. \quad (78)$$

Combining Equations 61 and 78, we obtain

$$E_{rem}^i = (1 - \rho_i) \|r_t^i\|, \quad (79)$$

$$\|\Delta W_i\|_{C_p} = \rho_i \sqrt{\lambda} \frac{\|r_t^i\|}{\sqrt{\tilde{S}_i}}. \quad (80)$$

D Derivation of the Scaled Editing Strength

This section derives the closed-form expression of the scaled editing strength in Equation 15. For a single edit request, the Effective Update Rate is given by

$$\rho_i = \frac{\frac{\alpha_i}{\lambda} \tilde{S}_i}{1 + \frac{\alpha_i}{\lambda} \tilde{S}_i}. \quad (81)$$

Define the scaled editing strength as

$$\tilde{\alpha}_i \triangleq \frac{\alpha_i}{\lambda}. \quad (82)$$

Substituting Equation 82 into Equation 81, we obtain

$$\rho_i = \frac{\tilde{\alpha}_i \tilde{S}_i}{1 + \tilde{\alpha}_i \tilde{S}_i}. \quad (83)$$

The target adaptive update rate is defined as

$$\rho_i^* = \frac{1}{\sqrt{1 + \mathcal{P}_i}}. \quad (84)$$

By matching ρ_i with ρ_i^* , Equation 83 becomes

$$\frac{1}{\sqrt{1 + \mathcal{P}_i}} = \frac{\tilde{\alpha}_i^* \tilde{S}_i}{1 + \tilde{\alpha}_i^* \tilde{S}_i}. \quad (85)$$

Let

$$x_i \triangleq \tilde{\alpha}_i^* \tilde{S}_i. \quad (86)$$

Then Equation 85 can be rewritten as

$$\frac{1}{\sqrt{1 + \mathcal{P}_i}} = \frac{x_i}{1 + x_i}. \quad (87)$$

Multiplying both sides of Equation 87 by $1 + x_i$ gives

$$\frac{1 + x_i}{\sqrt{1 + \mathcal{P}_i}} = x_i. \quad (88)$$

Rearranging terms yields

$$\frac{1}{\sqrt{1 + \mathcal{P}_i}} = x_i \left(1 - \frac{1}{\sqrt{1 + \mathcal{P}_i}} \right). \quad (89)$$

Hence,

$$x_i = \frac{\frac{1}{\sqrt{1+\mathcal{P}_i}}}{1 - \frac{1}{\sqrt{1+\mathcal{P}_i}}}. \quad (90)$$

Multiplying the numerator and denominator by $\sqrt{1+\mathcal{P}_i}$, we obtain

$$x_i = \frac{1}{\sqrt{1+\mathcal{P}_i} - 1}. \quad (91)$$

Substituting Equation 86 into Equation 91, we obtain

$$\tilde{\alpha}_i^* \tilde{S}_i = \frac{1}{\sqrt{1+\mathcal{P}_i} - 1}, \quad (92)$$

which yields the closed-form scaled editing strength

$$\tilde{\alpha}_i^* = \frac{1}{\tilde{S}_i} \left(\frac{1}{\sqrt{1+\mathcal{P}_i} - 1} \right). \quad (93)$$

E Datasets and Baselines

We evaluate AdaPE on three widely used knowledge editing benchmarks: **Wiki-CounterFact**, **CounterFact**, and **ZsRE**. These benchmarks cover complementary aspects of factual model editing, including counterfactual factual rewriting, robustness to paraphrased queries, transfer of edited knowledge to related queries, and preservation of unrelated knowledge.

Wiki-CounterFact. Wiki-CounterFact evaluates knowledge editing under more realistic knowledge-update scenarios. In addition to verifying whether the target knowledge is successfully injected, the benchmark evaluates whether the updated fact can be transferred to semantically related queries while preserving unrelated model behavior. We therefore report efficacy, portability, locality, and fluency following the benchmark protocol. We use all 839 samples in our experiments.

CounterFact. CounterFact is a standard benchmark for factual model editing. Each instance contains a subject–relation–object fact together with a counterfactual target object. Besides the original rewrite prompt, the benchmark provides paraphrase prompts and neighborhood prompts, enabling evaluation of edit success, generalization to semantically equivalent queries, and undesired interference with neighboring facts. We use 10,000 edit requests in the main batch-editing experiments and report efficacy, generalization, specificity, and fluency.

ZsRE. ZsRE is derived from the Zero-Shot Relation Extraction dataset. Each knowledge item can be represented as a subject–relation–object relation and is expressed through natural-language questions. The dataset provides rephrased queries for evaluating whether the edited knowledge generalizes across different linguistic formulations, together with unrelated queries for evaluating preservation. We use 10,000 samples and report efficacy, generalization, and specificity.

Baselines. We compare AdaPE with five representative knowledge editing methods: **FT**, **ROME**, **MEMIT**, **CORE**, and **AlphaEdit**. Together, these baselines cover optimization-based editing, single-fact direct editing, large-scale batch editing, context-robust editing, and projection-based stability preservation.

FT. FT is a direct optimization-based baseline that updates model parameters by fine-tuning on the target edit requests. The model is optimized toward the desired target facts without retraining from scratch. Although FT can achieve strong fitting on the edited examples, unrestricted parameter updates may overfit the target requests and interfere with unrelated knowledge. It therefore serves as a simple but important reference for the efficacy–locality trade-off.

ROME. ROME is a representative locate-and-edit method for single-fact editing. It is motivated by the observation that factual associations in autoregressive Transformers can be mediated by middle-layer MLP modules when processing the subject representation. ROME interprets the selected MLP as an associative memory and rewrites a target association through a closed-form rank-one update. This design enables precise modification of individual factual associations while maintaining relatively strong generalization and specificity in single-edit settings.

MEMIT. MEMIT extends direct model editing from individual edits to batch and mass editing. Instead of independently applying a sequence of rank-one updates, MEMIT distributes the desired memory modifications across a set of critical MLP layers and jointly solves for multiple edits. Its multi-layer closed-form update makes it a particularly relevant fixed-strength baseline for the batch-editing setting considered in this work.

CORE. CORE is a locate-and-edit method designed to improve robustness under contextual variation. It is motivated by the observation that an edit that succeeds under an isolated prompt may become ineffective when semantically related prefix contexts are introduced. CORE incorporates context-aware representations and cross-prefix regularization during editing, encouraging the injected fact to remain effective across different contextual formulations.

AlphaEdit. AlphaEdit improves editing stability by constraining parameter perturbations to the null space associated with preserved knowledge. Rather than relying only on the magnitude of the update, AlphaEdit explicitly restricts the update direction to reduce interference with existing factual associations. It therefore provides a strong projection-based baseline for evaluating the balance between successful knowledge injection and preservation.

For locate-and-edit baselines such as MEMIT, CORE, and AlphaEdit, the editing configuration and editing strength are fixed globally across requests according to their standard batch-editing setup. In contrast, AdaPE assigns an instance-specific editing strength according to the editing pressure of each request. Thus, different requests in the same batch can absorb different fractions of their target residual according to their correction demand and local geometric sensitivity.

F Experimental Setup and Reproducibility

All experiments are implemented using **EasyEdit v2.0**. The software environment consists of Python 3.10, PyTorch 2.1.2, HuggingFace Transformers 4.44.2, and CUDA 12.1. Experiments are conducted on two machines equipped with 48GB vGPUs.

We evaluate two instruction-tuned decoder-only language models: **Llama3-8B-Instruct** and **Qwen2.5-7B-Instruct**. For each model, we evaluate all methods under the same EasyEdit-based experimental framework. The main batch-editing experiments contain 10,000 edit requests for CounterFact, 10,000 edit requests for ZsRE, and all 839 instances of Wiki-CounterFact.

The fixed-strength trade-off experiment in Section 4.3 is conducted on CounterFact. The pressure-stratified experiment in Section 4.4 is conducted on CounterFact using Llama3-8B-Instruct.

AdaPE follows the standard ROME/MEMIT

locate-and-edit paradigm and operates on localized MLP editing sites rather than arbitrary Transformer layers. For example, the Llama3-8B-Instruct experiments use early-to-middle MLP layers 4–8. Interactions among different edit requests are handled through the shared MEMIT-style batch solver instead of independently summing single-edit updates.

We describe AdaPE as *tuning-free* in the sense that it does not require manually tuned global editing-strength or stability coefficients. The method nevertheless uses layer-wise empirical density statistics as automatically computed normalization quantities. Therefore, “tuning-free” specifically refers to the absence of manually selected global editing-strength or stability hyperparameters rather than independence from data-dependent statistics.

G Evaluation Metrics

We follow the benchmark-specific evaluation protocols implemented in our EasyEdit-based evaluation pipeline. Since CounterFact, ZsRE, and Wiki-CounterFact adopt different forms of evaluation prompts, we define their metrics separately.

Let f_θ denote the original model and $f_{\theta'}$ the edited model. For edit request i , let q_i denote the original edit prompt, o_i the original object, and $\hat{o}_i$ the desired edited object. We use $\mathbb{I}[\cdot]$ to denote the indicator function.

G.1 CounterFact Metrics

For CounterFact, edit success is determined by comparing the probability of the desired edited object with that of the original object.

Efficacy (Eff.). Efficacy measures whether the desired edited object becomes more probable than the original object under the original rewrite prompt. For each edit, we define

$$e_i^{\text{CF}} = \mathbb{I}[P_{\theta'}(\hat{o}_i | q_i) > P_{\theta'}(o_i | q_i)]. \quad (94)$$

The dataset-level efficacy is

$$\text{Eff}_{\text{CF}} = \frac{100}{N} \sum_{i=1}^N e_i^{\text{CF}}. \quad (95)$$

Generalization (Gen.). Let $\mathcal{R}_i$ denote the set of paraphrased prompts associated with edit i . We first compute the generalization score for each edit:

$$g_i^{\text{CF}} = \frac{1}{|\mathcal{R}_i|} \sum_{q \in \mathcal{R}_i} \mathbb{I}[P_{\theta'}(\hat{o}_i | q) > P_{\theta'}(o_i | q)]. \quad (96)$$

The final generalization score is

$$\text{Gen}_{\text{CF}} = \frac{100}{N} \sum_{i=1}^N g_i^{\text{CF}}. \quad (97)$$

Specificity (Spe.). Let $\mathcal{N}_i$ denote the neighborhood prompts associated with edit i . For each edit, we define

$$s_i^{\text{CF}} = \frac{1}{|\mathcal{N}_i|} \sum_{q \in \mathcal{N}_i} \mathbb{I}[P_{\theta'}(o_i | q) > P_{\theta'}(\hat{o}_i | q)]. \quad (98)$$

The dataset-level specificity is

$$\text{Spe}_{\text{CF}} = \frac{100}{N} \sum_{i=1}^N s_i^{\text{CF}}. \quad (99)$$

G.2 ZsRE Metrics

For ZsRE, efficacy and generalization are evaluated as answer accuracy on the original and rephrased questions, respectively.

Efficacy (Eff.). For each edit request, we define

$$e_i^{\text{ZsRE}} = \mathbb{I}[\hat{o}_i = \arg \max_o P_{\theta'}(o | q_i)]. \quad (100)$$

The efficacy score is

$$\text{Eff}_{\text{ZsRE}} = \frac{100}{N} \sum_{i=1}^N e_i^{\text{ZsRE}}. \quad (101)$$

Generalization (Gen.). For the rephrased prompt set $\mathcal{R}_i$, we define

$$g_i^{\text{ZsRE}} = \frac{1}{|\mathcal{R}_i|} \sum_{q \in \mathcal{R}_i} \mathbb{I}[\hat{o}_i = \arg \max_o P_{\theta'}(o | q)]. \quad (102)$$

The dataset-level generalization score is

$$\text{Gen}_{\text{ZsRE}} = \frac{100}{N} \sum_{i=1}^N g_i^{\text{ZsRE}}. \quad (103)$$

Specificity (Spe.). Let $\mathcal{U}_i$ denote the unrelated prompts associated with edit i , and let $y(q)$ denote the corresponding ground-truth answer. We define the instance-level specificity as

$$s_i^{\text{ZsRE}} = \frac{1}{|\mathcal{U}_i|} \sum_{q \in \mathcal{U}_i} \mathbb{I}[y(q) = \arg \max_o P_{\theta'}(o | q)]. \quad (104)$$

The final specificity score is

$$\text{Spe}_{\text{ZsRE}} = \frac{100}{N} \sum_{i=1}^N s_i^{\text{ZsRE}}. \quad (105)$$

G.3 Wiki-CounterFact Metrics

Wiki-CounterFact additionally evaluates whether the edited knowledge can be used in semantically related or downstream queries.

Efficacy (Eff.). For each edit request,

$$e_i^{\text{Wiki}} = \mathbb{I}[\hat{o}_i = \arg \max_o P_{\theta'}(o | q_i)]. \quad (106)$$

The efficacy score is

$$\text{Eff}_{\text{Wiki}} = \frac{100}{N} \sum_{i=1}^N e_i^{\text{Wiki}}. \quad (107)$$

Portability (Por.). Let $\mathcal{P}_i$ denote the portability prompts associated with edit i , and let $a(q)$ denote the correct answer to portability query q . For each edit, we compute

$$p_i = \frac{1}{|\mathcal{P}_i|} \sum_{q \in \mathcal{P}_i} \mathbb{I}[a(q) = \arg \max_o P_{\theta'}(o | q)]. \quad (108)$$

The portability score is

$$\text{Por} = \frac{100}{N} \sum_{i=1}^N p_i. \quad (109)$$

Locality (Loc.). Let $\mathcal{L}_i$ denote neighboring or unrelated prompts whose predictions should remain unchanged after editing. For each edit, we define

$$\ell_i = \frac{1}{|\mathcal{L}_i|} \sum_{q \in \mathcal{L}_i} \mathbb{I}[\hat{y}_{\theta'}(q) = \hat{y}_{\theta}(q)], \quad (110)$$

where

$$\begin{aligned} \hat{y}_{\theta}(q) &= \arg \max_o P_{\theta}(o | q), \\ \hat{y}_{\theta'}(q) &= \arg \max_o P_{\theta'}(o | q). \end{aligned} \quad (111)$$

The dataset-level locality is then

$$\text{Loc} = \frac{100}{N} \sum_{i=1}^N \ell_i. \quad (112)$$

A higher locality score indicates that editing causes less collateral change to knowledge outside the intended editing scope.

G.4 Fluency

Fluency is reported for CounterFact and Wiki-CounterFact and is evaluated using generation entropy. The metric is designed to detect degenerate or repetitive generation after editing.

For generated text x , let $g_n(k; x)$ denote the empirical probability of the n -gram k . The n -gram entropy is

$$H_n(x) = - \sum_k g_n(k; x) \log_2 g_n(k; x). \quad (113)$$

Following the standard CounterFact evaluation, bigram and trigram entropy are combined as

$$\text{Flu}(x) = \frac{1}{2} \left(\frac{2}{3} H_2(x) + \frac{4}{3} H_3(x) \right). \quad (114)$$

The reported fluency score is averaged over generated evaluation samples. Larger values indicate more diverse and less repetitive generation.

Batch-editing setting. Our primary experiments evaluate large-scale simultaneous knowledge updates rather than isolated single edits. Consequently, these metrics measure not only whether an individual target fact is successfully injected, but also whether generalization, portability, and preservation remain stable when many heterogeneous edit requests compete for the same model parameters.

H Additional Robustness Experiments

The full-scale experiments in the main paper are computationally expensive, particularly for 7–8B parameter models with 10,000 simultaneous edit requests. To evaluate whether the observed improvements depend on a particular sampling of the benchmark, we additionally conduct a five-seed experiment.

We use random seeds 40, 41, 42, 43, and 44. For CounterFact and ZsRE, each run uses a randomly sampled 5K subset from the original benchmark distribution. Wiki-CounterFact contains only 839 instances, so the full dataset is evaluated in every run.

Due to the computational cost of repeating every baseline five times, we compare AdaPE with MEMIT. MEMIT is the most directly related fixed-strength batch-editing method and therefore provides the most relevant comparison for evaluating the stability of adaptive editing strength.

Table 3 reports the mean and standard deviation over the five runs.

Across both models, AdaPE preserves the same overall trend observed in the full-scale experiments. The improvements in efficacy and generalization/portability remain consistent across different random samplings, while preservation-related

metrics remain close to those of MEMIT. These results indicate that the gains of AdaPE are not caused by a single favorable benchmark sample.

I Ablation of Editing Pressure Components

AdaPE characterizes editing difficulty using two complementary factors: *residual energy*, which measures the amount of correction required by an edit, and *spatial sensitivity*, which characterizes how sensitive the local parameter geometry is to the corresponding update.

To isolate their contributions, we conduct an ablation study on Llama3-8B-Instruct using the same data configuration as the main experiments: 10,000 edit requests for CounterFact, 10,000 requests for ZsRE, and all 839 Wiki-CounterFact instances. We compare MEMIT, a variant using only residual energy, a variant using only spatial sensitivity, and the complete AdaPE edit-density formulation.

Using residual energy alone already improves substantially over MEMIT on efficacy and generalization, showing that different edit requests require different amounts of correction. Spatial sensitivity alone also improves editing performance while retaining slightly stronger preservation behavior on several metrics. Their combination provides the best efficacy and generalization/portability results, demonstrating that the two terms capture complementary aspects of editing difficulty.

J Qualitative Analysis across Editing Pressure

To provide a more intuitive interpretation of editing pressure, we divide CounterFact edit requests into pressure quartiles from low pressure (Q1) to high pressure (Q4) and inspect representative cases.

Low-pressure edits typically involve relatively weak factual associations, such as less salient personal attributes, niche organizations, or secondary entities. In these cases, the original subject–relation–object association is relatively weak and can often be replaced with a small parameter perturbation.

Medium-pressure edits increasingly involve recognizable entities, brands, geographic information, historical context, or stronger type constraints. Existing model knowledge therefore begins to compete with the counterfactual target.

High-pressure edits frequently correspond to strongly memorized world-knowledge associations,

Model	Algorithm	Wiki-CounterFact				CounterFact				ZsRE		
		Eff.↑	Por.↑	Loc.↑	Flu.↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Eff.↑	Gen.↑	Spe.↑
Llama3-8B-Instruct	MEMIT	63.33±1.99	29.71±0.57	51.37±0.31	6.16±0.03	90.55±1.10	46.79±0.29	86.12±0.10	6.25±0.02	82.01±0.90	73.72±0.72	42.51±0.75
Llama3-8B-Instruct	AdaPE	75.35±0.61	32.48±0.26	50.34±0.14	6.14±0.02	95.83±0.34	52.30±0.58	85.67±0.11	6.24±0.02	86.90±0.59	78.36±0.60	42.16±0.81
Qwen2.5-7B-Instruct	MEMIT	67.96±1.28	43.20±0.47	44.59±0.20	6.17±0.02	95.03±0.52	65.17±0.46	66.23±0.35	6.18±0.01	72.38±0.73	67.69±0.61	42.22±0.58
Qwen2.5-7B-Instruct	AdaPE	69.82±0.66	44.15±0.31	44.64±0.16	6.17±0.02	95.33±0.31	65.93±0.39	66.16±0.47	6.18±0.02	73.55±0.56	68.35±0.48	42.47±0.55

Table 3: Five-seed robustness evaluation. We report mean $\pm$ standard deviation over random seeds 40–44. CounterFact and ZsRE use randomly sampled 5K subsets in each run, while Wiki-CounterFact uses all 839 instances. AdaPE consistently improves editing efficacy and generalization/portability over the fixed-strength MEMIT baseline while maintaining comparable locality, specificity, and fluency.

Algorithm	Wiki-CounterFact				CounterFact				ZsRE		
	Eff.↑	Por.↑	Loc.↑	Flu.↑	Eff.↑	Gen.↑	Spe.↑	Flu.↑	Eff.↑	Gen.↑	Spe.↑
MEMIT	64.24	29.71	51.34	6.16	86.47	42.36	84.07	6.25	77.25	69.54	43.43
Only residual energy	71.80	31.20	50.10	6.14	90.90	45.90	83.10	6.24	80.20	72.10	42.60
Only spatial sensitivity	69.50	30.70	50.80	6.15	89.40	44.80	83.80	6.24	79.10	71.00	43.10
AdaPE	74.85	32.16	50.32	6.14	92.27	47.20	83.20	6.24	81.56	73.67	42.97

Table 4: Ablation of residual energy and spatial sensitivity on Llama3-8B-Instruct. Residual energy primarily captures correction demand and contributes strongly to efficacy and generalization/portability, whereas spatial sensitivity provides information about update safety. Combining both factors produces the strongest overall efficacy–preservation trade-off.

including capitals, continents, official languages, and highly salient entity–brand relations. These associations appear across many contexts in pre-training data and are substantially more resistant to parameter modification.

K High-Pressure Failure Cases and Diagnostic

The adaptive mechanism of AdaPE deliberately reduces the residual absorption rate for high-pressure requests when aggressive editing would cause large collateral perturbations. Under a strict requirement that every target fact must be completely corrected, an insufficiently corrected high-pressure request should therefore be regarded as an editing failure rather than silently treated as a successful edit.

Our qualitative inspection shows that many failed high-pressure examples correspond to highly salient factual relations or semantically difficult counterfactual targets. Representative examples are shown in Table 6.

To make such cases observable in deployment, we additionally use the *achieved update ratio* ρ_i as a post-edit diagnostic. Given a user-defined warning threshold τ_ρ , we define the flagged set as

$$\mathcal{F}_{\tau_\rho} = \{i \mid \rho_i < \tau_\rho\}. \quad (115)$$

A low achieved update ratio indicates that only a limited fraction of the desired residual has been absorbed.

On the 10,000-request CounterFact experiment, the warning criterion used in our analysis identifies 487 requests, corresponding to approximately 4.87% of all edits, as potentially under-corrected cases. These cases can be explicitly surfaced for additional verification rather than being treated as ordinary successful edits.

For applications with stricter correction requirements, flagged requests can be handled using a two-stage workflow. First, the model performs stability-constrained batch editing. Post-edit verification is then used to identify insufficiently corrected requests. These difficult cases can subsequently be re-edited using smaller batches or stronger editing configurations. This workflow preserves the stability benefits of AdaPE for the majority of requests while enabling stricter treatment of difficult outliers.

L Discussion of Scope

Projection and adaptive strength allocation. Projection-based editing and AdaPE address complementary aspects of the editing problem. Projection methods such as AlphaEdit primarily determine *where* a parameter update should lie in order to reduce interference with preserved knowledge. AdaPE instead determines *how strongly* each request should absorb its target residual within the available update geometry.

Even after projection, different requests can re-

Quartile	Subject + Prompt	Original	Target	P	Success	Interpretation / Failure Reason
Q1	The native language of Alain Ma-banckou is	French	Swedish	1.9090	✓	Low-pressure personal attribute; the original association is relatively weak.
Q1	Daeg Faerch’s profession is a	actor	mathematician	1.9080	✓	Weakly coupled biographical association.
Q1	CBS Evening News debuted on	CBS	PBS	1.4500	✓	The requested replacement can be absorbed with relatively small interference.
Q2	The original language of The Icelandic Dream is	Icelandic	Tamil	1.9820	✓	Moderate semantic constraint but still sufficiently writable.
Q2	Outlook.com is owned by	Microsoft	Mobile	2.1595	✓	A stronger entity association, but the target remains editable.
Q2	Liverpool John Lennon Airport, which is called after	Liverpool	Cambridge	2.1595	×	Name coupling: the airport is strongly associated with Liverpool, making the named-after relation difficult to replace.
Q3	Ferrari 156 F1, created by	Ferrari	Suzuki	2.4455	×	Brand coupling: Ferrari is strongly associated with Formula 1 history.
Q3	Porsche 911, developed by	Porsche	Honda	2.2970	×	Brand coupling: the model tightly associates the Porsche 911 with Porsche as its manufacturer.
Q3	Victoria Land, which was named for	Victoria	Hollywood	2.4550	×	Lexical and historical name coupling makes the original association difficult to override.
Q4	Spain’s capital city is	Madrid	Valencia	7.3943	×	Deep historical coupling: Spain–Madrid is a highly frequent and stable world-knowledge association.
Q4	Germany is a part of the continent of	Europe	Antarctica	6.9283	×	Geographic commonsense coupling makes the original association highly resistant to editing.
Q4	In India, an official language is	Hindi	Finnish	5.8154	×	National-identity and language coupling strongly favors the original fact.

Table 5: Representative edits from different editing-pressure quartiles on CounterFact. Low-pressure facts are generally associated with weaker existing factual bindings and are easier to rewrite, whereas high-pressure facts frequently correspond to strongly memorized entity, geographic, historical, or commonsense associations.

Coupling Type	Subject + Prompt	Original	Target	P	Success
Capital–country coupling	Spain’s capital city is	Madrid	Valencia	7.3943	×
Geographic commonsense coupling	Germany is a part of the continent of	Europe	Antarctica	6.9283	×
International-organization coupling	Germany is a member of	UNESCO	Hamas	7.0498	×
Twin-city coupling	Istanbul is a twin city of	Athens	Istanbul	8.6445	×

Table 6: Representative high-pressure failure cases. These edits involve strongly memorized or highly constrained factual associations, for which fully absorbing the counterfactual residual may require substantially larger and potentially more destructive parameter changes.

tain different residual magnitudes, key-space sensitivities, and levels of writability. Consequently, a single global editing strength may remain suboptimal for a heterogeneous batch. From this perspective, projection and adaptive strength allocation are complementary rather than competing strategies.

Batch interactions. AdaPE does not independently compute and sum isolated single-fact edits. It retains the MEMIT-style joint batch formulation, in which interactions among edit keys are included through the shared batch solution. AdaPE changes the uniform strength allocation into an instance-adaptive diagonal allocation while leaving the underlying joint solver structure unchanged.

Model scope. Our current experiments are restricted to Llama3-8B-Instruct and Qwen2.5-7B-Instruct, both of which are decoder-only Transformers at the 7–8B scale. This choice follows the

experimental setting commonly used by locate-and-edit approaches such as ROME, MEMIT, CORE, and AlphaEdit, which operate on MLP associative-memory modules.

We therefore restrict our empirical claim to decoder-only, MLP-associative-memory editing and do not claim architectural universality for substantially larger models or encoder–decoder architectures. Evaluating such models would also require substantially greater computational resources than the two 48GB vGPU machines used in our experiments.