

Fine-Tuning Was Not Broken but Diffuse: Precise Knowledge Editing via Projected Error Signals

Shahar Katz, Lior Wolf

Blavatnik School of Computer Science, Tel Aviv University

Avi Caciularu

Google Research

Abstract

Transformer-based large language models (LLMs), trained via backpropagation on massive textual corpora, encode substantial factual knowledge, yet may also retain false or outdated associations. Prior work on factual editing in LLMs shows that fine-tuning (FT) often induces a repelling effect, degrading generalization and overwriting non-targeted knowledge, motivating alternative approaches such as rank-1 optimizations. In this work, we revisit knowledge editing in LLMs through FT by analyzing the activations and Vector-Jacobian Products (VJPs), the error signals produced during backpropagation when learning new facts. We show that each layer and token induces its own error signals, and that attempts to edit a specific association can inadvertently encourage the model to memorize the entire prompt in a diffuse or inefficient manner. Building on this analysis, we introduce **Precise-FT (PFT)**, a method that constructs gradient updates from a small set of projected error signals most strongly associated with the learned fact, achieving strong performance across multi-knowledge editing benchmarks and LLMs. Our code is available at: <https://github.com/shaharkatz3/PFT>.

1 Introduction

Transformer-based large language models (LLMs) encode a substantial amount of factual knowledge during pre-training, which may become outdated, misleading, or incorrect over time. A substantial body of work has therefore studied how to edit specific factual associations in a precise manner, without deleting other associations or damaging the LLMs’ general capabilities to generate text.

Early work in this area included continuous fine-tuning (FT) and hyper-network-based editing approaches (Zhu et al., 2020; Mitchell et al., 2021). A pivotal work by Meng et al. (2022) proposed a low-rank, closed-form solution based on a rank-one update (Bau et al., 2020), which steers model activations by mapping entity representations to those that produce the desired target objects. Follow-up works, such as MEMIT (Meng et al., 2023), extended this approach to support multiple editing scenarios simultaneously and to reduce the degradation that such edits impose on the model (Ma et al., 2025; Gu et al., 2024; Fang et al., 2025). The success of these approaches has shifted attention away from FT, despite FT being the most natural mechanism for updating a model in the same optimization framework used during pre-training (Yao et al., 2023; Zhang et al., 2024).

In this work, we revisit FT for precise knowledge editing through two complementary lenses. First, we analyze the Vector-Jacobian Products (VJPs), the per-token error signals produced during backpropagation. We show that FT updates are influenced by VJPs from all tokens in the prompt, rather than being restricted to those directly relevant to the edited fact, and that this diffuse signal can cause edits to interfere with one another across training steps. Second, we examine the structure of FT gradients and identify noise components induced by the proximity of layer activations in the embedding space. To mitigate this effect, we leverage coordinate-wise statistics at each layer to reshape the update directions, a process we refer to as covariance smoothing.

Building on these observations, we introduce **Precise-FT (PFT)**, a method that refines the fine-tuning process by selecting and projecting the signals that contribute to each gradient update. Empirically, PFT achieves state-of-the-art performance across multiple knowledge editing benchmarks and LLMs on both massive and sequential editing tasks, exhibiting negligible degradation of overall model capabilities compared to existing methods. Unlike closed-form locate-and-edit methods, PFT directly controls which token-level FT error signals contribute to iterative updates, without constraining each association to a single rank-one injection.

In summary, our contributions are: (i) a mechanistic analysis of FT, showing that VJPs induce distributed error signals that lead to interference and imprecise updates; (ii) the identification of a key FT failure mode driven by similarity between edited and unrelated activations; (iii) the introduction of **Precise-FT (PFT)**, which builds updates from relevant token-level error signals with covariance-based smoothing; (iv) a new analysis of the rank-1 methods, presenting their smoothing aspect; and (v) extensive empirical evaluation demonstrating that PFT achieves state-of-the-art performance across multiple benchmarks, while preserving general model capabilities and enabling efficient, scalable editing.

2 Related Work

Knowledge editing studies how to update factual associations in LLMs. For example, given the prompt “The T-rex was a type of”, a model may produce the answer “dinosaur”. Previous work formalizes such knowledge using triplets of the form (s, r, o) , where a relation r connects a subject s (e.g., T-rex) to an object o (e.g., dinosaur). Widely used benchmarks for evaluating this task include CounterFact (Meng et al., 2022) and ZSRE (Levy et al., 2017), where a new object is given for a set of associations, such as “The T-rex was a type of *music*”.

Early work on knowledge editing attempted to solve this problem through continued fine-tuning (FT) of pre-trained LLMs (Zhu et al., 2020). These approaches were limited in their ability to generalize beyond the prompts used during FT and often caused unintended changes to unrelated knowledge. Recent attempts to revisit FT either rely on large amounts of external data or substantial computational resources (Gangadhar & Stratos, 2024). Meta-learning approaches, such as MEND (Mitchell et al., 2021), also added a hypernetwork to predict parameter updates, introducing additional model components and training overhead. Despite these efforts, FT remains limited in its ability to perform precise knowledge editing efficiently without failing to preserve the model’s broader capabilities.

As interpretability research progressed, studies pointed to the role of MLP layers as key components responsible for storing factual knowledge, serving as associative key-value memories (Geva et al., 2021; Elhage et al., 2021). Building on this observation, Rank-One Model Editing (ROME) (Meng et al., 2022) employs a locate-and-edit approach: it first optimizes a value vector for each edited relation, and then injects it into the target layer using the normal equations (Lang, 2012), effectively replacing the iterative nature of FT with a single closed-form update. The subsequent work of MEMIT (Meng et al., 2023), enabled multiple editing simultaneously by distributing updates across layers. RECT, PRUNE and AlphaEdit (Gu et al., 2024; Ma et al., 2025; Fang et al., 2025) improved MEMIT’s robustness in batch editing, where associations are given in a sequential setting. In spite of these advances, direct editing approaches still exhibit degradation as the number of edits grows and rely on low-rank updates that restrict the expressiveness of each modification. See Appendix A for further details about closed-form solution methods.

Another line of work proposes augmenting LLMs with additional parameters or external memory components that store edited knowledge. Methods such as WISE (Wang et al., 2024; Hu et al., 2024), MELO (Yu et al., 2024), and prompt-based editing approaches (Madaan et al., 2022) avoid modifying the internal parameters of the base model and instead route edited knowledge through auxiliary mechanisms. In this work, we focus on methods that directly modify the parameters of pre-trained LLMs, and therefore (like other SOTA knowledge editing methods) do not compare against approaches that rely on external editing modules.

3 Background

In this section we review the Generative Pre-trained Transformer (GPT) architecture and the backward-pass formulation used during training and FT. For further details on transformer LLMs we refer the reader to Vaswani et al. (2017); Brown et al. (2020).

Transformer Language Models We consider GPT-based autoregressive LLMs composed of L transformer blocks connected through a shared residual stream. Given an input prompt of n tokens $(t_1, \dots, t_n)$, the model embeds each token and predicts the next token conditioned on all previous ones.

Let $x_i^l \in \mathbb{R}^d$ denote the hidden representation (activation) of token i at layer l where $d \in \mathbb{N}$ is the hidden dimension. Each transformer block consists of a self-attention module followed by an MLP (feed-forward network). Ignoring layer normalization and dropout for clarity, the hidden state update can be written as $x_i^{l+1} = x_i^l + \text{Attn}(x_i^l) + \text{MLP}(x_i^l + \text{Attn}(x_i^l))$. The attention mechanism mixes information between tokens, while the MLP operates independently on each token representation. Prior work suggests that much of the static factual knowledge stored in LLMs resides in the MLP layers (Meng et al., 2022; Fang et al., 2025). The MLP typically consists of two linear matrices separated by a nonlinearity $\sigma(\cdot)$:

$$\text{MLP}(x) = W_2 \sigma(W_1 x^\top), \quad W_1, W_2^\top \in \mathbb{R}^{d_2 \times d}. \quad (1)$$

After the final transformer block, the representation of the last token is projected using a decoding matrix $D \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $\mathcal{V}$ is the vocabulary. The probability distribution over the next token is then obtained via $\hat{y}_i = D x_i^L$, $p(t_{i+1} | t_{\leq i}) = \text{softmax}(\hat{y}_i)$.

Backward Pass and VJPs During training and FT, model parameters are updated using backpropagation (Le Cun, 1988; Katz & Wolf, 2025). While the forward pass propagates token activations through the network, the backward pass propagates error signals in the opposite direction. Training typically minimizes the negative log-likelihood (NLL) loss between the predicted token distribution and the target token with indices t . Backpropagation computes gradients with respect to every activation in the network. For each token i at layer l , the backward signal is: $\mathcal{L} = \text{NLL}(p, t) = \sum_{i=1}^n -\log p_i[t]$, $\delta_i^l = \frac{\partial \mathcal{L}}{\partial x_i^l} \in \mathbb{R}^d$. These signals, δ_i^l , are commonly referred to as Vector-Jacobian Products (VJPs).

Gradients with respect to a weight matrix W can be expressed as the sum of outer products between each of the input’s tokens’ activations and their corresponding backward signals. Finally, parameters are then updated using a learning rate η :

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{i=1}^n x_i^\top \frac{\partial \mathcal{L}}{\partial (W x_i^\top)} = \sum_{i=1}^n x_i^\top \delta_i, \quad W \leftarrow W - \eta \left(\frac{\partial \mathcal{L}}{\partial W} \right)^\top \quad (2)$$

4 Activations and VJP-Based Analysis of FT

In this section, we examine how the activations and VJPs produced during FT shape the resulting gradients and influence the model’s future inferences. Throughout this section, we use three representative models: GPT2-XL (1.5B) (Brown et al., 2020), GPT-J (6B) (Wang & Komatsuzaki, 2021), and Llama-3 (8B, Instruct) (Grattafiori et al., 2024).

We begin by revisiting an observation by Katz et al. (2024) regarding the effect of a single FT step. Consider a weight matrix W that is updated using a single gradient step on a prompt whose corresponding activations are $x_1, \dots, x_n$, with backward signals (VJPs) $\delta_1, \dots, \delta_n$. The gradient of this step is given by a sum of outer products between activations and VJPs. Following the update rule in Equation 2, we analyze how a new activation $\tilde{x}$ is processed by the updated weights $\tilde{W}$:

$$\tilde{W} \tilde{x}^\top = \left(W - \eta \left(\frac{\partial \mathcal{L}}{\partial W} \right)^\top \right) \tilde{x}^\top = W \tilde{x}^\top - \eta \left(\sum_{i=1}^n x_i^\top \delta_i \right)^\top \tilde{x}^\top = W \tilde{x}^\top - \eta \sum_{i=1}^n \delta_i^\top (x_i \tilde{x}^\top) \quad (3)$$

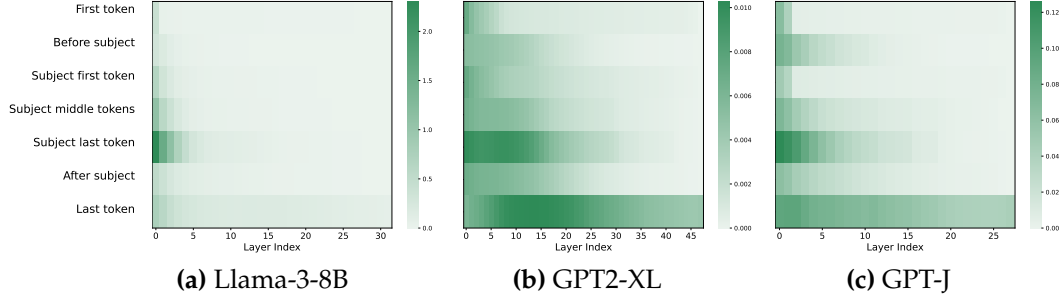


Figure 1: The mean norm of the VJPs with respect to each transformer layer’s output.

This decomposition consists of two components: the original output $W\tilde{x}^\top$, and an additive term induced by the update, $\eta \sum_i \delta_i^\top (x_i \tilde{x}^\top)$. The second component is directed along each VJP $\delta_i^\top$ and scaled by the inner product $x_i \tilde{x}^\top$, i.e., the similarity between the new activation and the activations observed during FT.

For precise editing, we would prefer that the output for unrelated activations remain unchanged, for example, when $\tilde{x}$ corresponds to a prompt from another domain of the FT’s one. This would require approximate orthogonality between $\tilde{x}$ and the activations observed during FT: $x_i \tilde{x}^\top \approx 0$. Otherwise, the output for $\tilde{x}$ will deviate from its original value, introducing unintended changes that may propagate through subsequent layers.

4.1 Affected Tokens

Closed-form solutions are limited to rank-one updates due to their reliance on key-value pair injections (Appendix A). Consequently, given a relation (s, r, o) , prior work such as ROME and MEMIT (Meng et al., 2022; 2023) focuses on modifying only the activation of the final subject token s , overlooking potential contributions from other tokens.

However, as shown in Equation 3, each token in the prompt contributes to the gradient through its corresponding VJP. To study this effect, we sample 100 relations from CounterFact (Meng et al., 2022) and measure the norms of the VJPs δ_i at selected layers during individual FT steps. Following prior work that identifies the second MLP projection matrix W_2 as a primary location of stored information in LLMs (Geva et al., 2021; Meng et al., 2023), we focus our analysis on these layers. The mean VJPs’ norms of W_2 are presented in Figure 1, grouped by layer index and token positions within the prompt.

As expected, the largest VJP norms are observed at the last subject token, consistent with Meng et al. (2022), and in the prompt’s last token, where the calculation of the loss is the closest. However, we also notice high norms in other tokens of the subject tokens, especially its middle one, where the binding of the subject entity is aggregated between multiple tokens (Elhage et al., 2021; Geva et al., 2023).

Additionally, we observe non-negligible contributions from additional token positions, particularly early tokens in the prompt and layers. This observation has important implications: consider two prompts, where only one is used during FT: the edited prompt “The T-rex was a type of”, and a non-edited prompt “The T-bone is a type of”. Under the GPT2-XL tokenizer, both prompts share the same three-token prefix “The T-”, and therefore induce the same activations for these prefix tokens. As a result, FT updates computed on the edited prompt may also affect the forward pass of the non-edited prompts, especially when it accumulates across training steps.

This analysis suggests that while it is beneficial to ignore error signals originating from non-subject tokens, restricting updates solely to the final subject token is suboptimal, as relevant information may be distributed across the subject’s multiple tokens.

4.2 Correlation Between Different Tokens

While the previous section focused on tokens that are shared across prompts and therefore induce identical activations, we now study the correlation between different tokens. In particular, we focus on subject tokens, which reduces the influence of common filler and structural tokens and highlights semantic differences between prompts.

Consider two prompts, where only one is used during FT: the edited prompt “The T-rex was a type of”, and a non-edited prompt “The T-bone is a type of”. While the prefixes are similar, the subject suffixes differ, “rex” vs. “bone”, leading to distinct token representations. In this section, we quantify how similar their activations are. We sample 100 relations from CounterFact and compute the average cosine similarity between activations corresponding to different subject suffixes across models and layers. We observe that this similarity ranges between 0.05 and 0.25, depending on the model and layer index.

This result indicates that even semantically distinct tokens exhibit non-negligible correlation in their representations. Formally, in Equation 3, the interaction $x_i^\top \tilde{x}$ is rarely zero, implying that VJPs δ_i contribute to the output of activations that are not directly associated with the edited prompt. As a consequence, editing one prompt may influence the processing of another, even when the tokens differ, leading to unintended propagation of editing signals.

5 PFT: Precise-FT

In this section we present **Precise-FT (PFT)**, a knowledge editing method for LLMs, designed to mitigate the two phenomena identified in section 4 and to enable targeted modification of factual associations while minimizing unintended changes to unrelated knowledge.

Selective Error Signals. As shown in subsection 4.1, standard FT propagates error signals from all tokens in the prompt, each contributing its own VJP. However, only a subset of these tokens, primarily those corresponding to the subject s , are directly relevant to the edited relation. Therefore, PFT modifies the backward pass by selectively constructing gradients using only the VJPs corresponding to the subject’s suffix tokens, which are most relevant to the desired edit. Formally, we define the updated gradients as:

$$\Delta_{\text{suffix}} = \sum_{i \in s \text{ suffix}} x_i^\top \frac{\partial \mathcal{L}}{\partial (W x_i^\top)} = \sum_{i \in s \text{ suffix}} x_i^\top \delta_i, \quad W \leftarrow W - \eta \Delta_{\text{suffix}}^\top. \quad (4)$$

This formulation restricts the update to a subset of token-level gradient components, localizing the edit to the representations most relevant to the target association. We note that the decision to use subject suffixes is motivated not only by the large VJP norms observed at these positions in Figure 1, but also by prior work linking subject suffixes to the formation of entity representations (Kaplan et al., 2025; Morand et al., 2025).

Given a maximum suffix length k and a subject containing m tokens, PFT selects only the final $\min(k, m)$ subject tokens and never extends the selection to non-subject tokens. We use $k = 4$, the median subject length under the GPT2 tokenizer, which captures most subject spans while bounding computation. In subsection C.6, we study the effect of varying the length of the selected suffix and analyze its impact on editing performance, leaving adaptive per-example selection as a direction for future work.

Smoothing Projection. We revisit the formation of token-level gradients (Equation 2): given an activation $x \in \mathbb{R}^d$ and its corresponding VJP δ , the gradient is given by their outer product. If x is highly correlated with typical activations from the same layer, denoted by $K_0 = \{\hat{k}_i\} \subseteq \mathbb{R}^d$, then its contribution may introduce unintended interference, as discussed in subsection 4.2. To mitigate this effect, we seek a transformed vector x' that remains close to x while being approximately orthogonal to the vectors of K_0 . Formally, we solve:

$$x' = \arg \min_x \|\hat{x} - x\|_2^2 + \|\hat{x}^\top K_0\|_2^2, \quad (5)$$

This optimization admits the following solution via regularized least squares:

$$x_{\text{smooth}}^\top = x'^\top = x^\top \left(K_0 K_0^\top + I \right)^{-1}. \quad (6)$$

We rewrite Equation 6 for batch editing given a set of activations $x_1, \dots, x_n$:

$$X = [x_1 \mid x_2 \mid \dots \mid x_n] \in \mathbb{R}^{d \times n}, \quad X_{\text{smooth}}^\top = X^\top (K_0 K_0^\top + I)^{-1} = X^\top M^{-1} \quad (7)$$

However, computing matrix inverses at every FT step is computationally prohibitive, with computational complexity $O(d^3)$. Instead of recomputing matrix inverses at each step, we perform a one-time LU decomposition (Van Loan & Golub, 1996) and efficiently compute the smoothed activations via:

$$M = P_{\text{perm}}^\top L U, \quad X_{\text{smooth}}^\top = X^\top M^{-1} = X^\top U^{-1} L^{-1} P_{\text{perm}} \quad (8)$$

where P_{perm} is a permutation matrix and $L, U \in \mathbb{R}^{d \times d}$ are triangular matrices. This reduces the per-step complexity to $O(d^2 n)$, which is comparable to standard matrix multiplications, making the approach practical for iterative FT.

We note that the LU decomposition could be replaced with naive matrix inversion, as well as Cholesky decomposition since $M = M^\top$ is symmetric and positive semi-definite. However, we find the LU decomposition to be more stable, as numerical noise and finite-sample estimation of K_0 may lead to deviations from exact symmetry or positive definiteness.

In Appendix A, we show via a new reformulation that methods such as MEMIT (Meng et al., 2023) incorporate similar smoothing mechanisms. However, these approaches rely on closed-form updates, which limit iterative application and constrain the update to low-rank structure, in contrast to PFT.

Additional Implementation Considerations. In this section, we briefly describe key design decisions in our implementation. Additional details are provided in Appendix B, along with ablation studies that validate these choices.

Materialize the Covariance Matrix. To compute Equation 6, we require an estimate of K_0 . Following prior work (Meng et al., 2022), we collect activations from general text using $n_0 = 100,000$ samples from Wikipedia. In practice, instead of storing $K_0 \in \mathbb{R}^{d \times n_0}$, we compute its covariance matrix $K_0 K_0^\top \in \mathbb{R}^{d \times d}$: $K_0 K_0^\top = \frac{\lambda_0}{n_0} \sum_{i=1}^{n_0} \hat{k}_i \hat{k}_i^\top$, $\lambda_0 \in \mathbb{R}$, which has constant size and is significantly more memory-efficient, since $d \ll n_0$. This covariance matrix captures the statistical correlations between activation dimensions. Consequently, the transformation in Equation 6 can be interpreted as covariance smoothing, suppressing directions in the activation space that are strongly correlated with typical representations. Finally, to account for the evolving activation distribution during FT, we maintain an updated covariance estimate by accumulating batch statistics $K_0 K_0^\top \leftarrow K_0 K_0^\top + X X^\top$ and periodically recompute the LU decomposition every 1,500 steps to reflect these updates.

Layer Selection. We apply the editing to the second MLP matrix, W_2 , following previous editing methods selection (Meng et al., 2022). In particular, MEMIT selects a subset of W_2 layers, typically five, based on causal mediation analysis (Pearl, 2022; Vig et al., 2020), identifying layers where the influence of the last subject token is strongest. Since PFT operates on the suffix of subject tokens, we adopt a similar number of edited layers, but shift them slightly by 1 – 3 layers earlier based on grid search tuning. This choice reflects layers that build entity representations, allowing us to better localize edits at the level of subject encoding (Meng et al., 2022; Elhage et al., 2021).

6 Experiments

Smoothing Projection To evaluate the effect of the smoothing transformation used by PFT, we repeat the experiment from subsection 4.2 and measure cosine similarity between subject-suffix activations before and after applying covariance smoothing. The results are shown in Figures 2(a) and 2(b) for three randomly sampled prompts using GPT-J, and summarized across layers in Figure 2(c). We observe that this transformation significantly reduces correlations between different activations, driving cosine similarity from values in the range 0.1–0.25 toward near orthogonality.

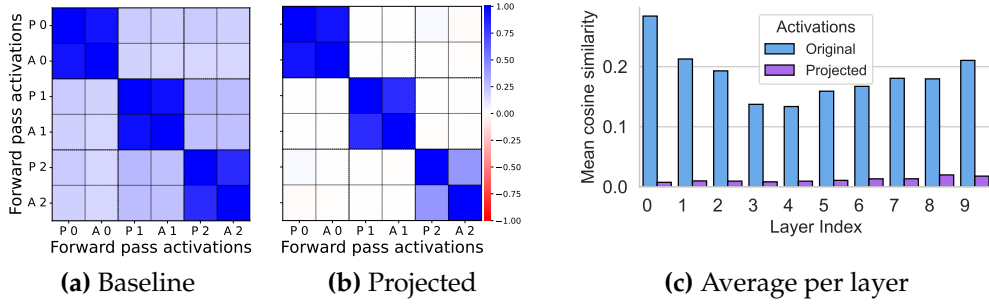


Figure 2: GPT-J cosine similarity between forward pass activations of last subject tokens, before and after projections. In (a) and (b) we visualize three associations, each with its base prompt (P) and one augmentation (A), sampled from the MLP of layer 1.

Taken together with the results from [section 4](#), our analysis shows that VJPs encode updates that are inherently distributed across tokens and correlated in the activation space. These observations motivate the need for methods that explicitly control which error signals contribute to the update.

Massive and Sequential Knowledge Editing We evaluate PFT on the task of knowledge editing using GPT2-XL, GPT-J, and Llama-3. We use these three models to match prior knowledge-editing evaluations, such as [Meng et al. \(2022\)](#) and [Fang et al. \(2025\)](#).

For comparison with prior work, we include ROME ([Meng et al., 2022](#)), MEMIT ([Meng et al., 2023](#)), PRUNE ([Ma et al., 2025](#)), RECT ([Gu et al., 2024](#)), NSE ([Jiang et al., 2025b](#)) and AlphaEdit ([Fang et al., 2025](#)). For comparison with standard fine-tuning, we use the FT method proposed by [Zhu et al. \(2020\)](#), which updates only a single W_2 layer and has been widely adopted in prior comparative studies as the representative fine-tuning baseline ([Meng et al., 2022](#); [Fang et al., 2025](#)); we refer to this baseline as *FT (old)* to distinguish it from our own FT-based editing implementations. Our selection of methods focuses on approaches that directly modify the second MLP projection matrix W_2 , without introducing additional parameters or architectural components, such as in [Wang et al. \(2024\)](#), and without the need for additional model training ([Mitchell et al., 2021](#)). Furthermore, we restrict our comparison to efficient methods that (i) can be executed on a single GPU within a few hours, and (ii) do not require training on large auxiliary datasets.

In addition, we also include three ablations: (i) our implementation of FT, *FT-Updated*, which follows the PFT pipeline without selective VJPs or smoothing, (ii) a variant of PFT that incorporates selective VJPs but without smoothing, *PFT w/o smooth* and (iii) a PFT variant with only covariance smoothing, *PFT w/o s.VJPs*. All ablations use the same PFT optimization pipeline, including the same edited layers and hyperparameters.

We evaluate two datasets: CounterFact ([Meng et al., 2022](#)) and ZSRE ([Levy et al., 2017](#)). Given an edited relation (s, r, o) , we report the editing criteria of **Efficacy (Eff.)**, **Generalization (Gen.)** (paraphrase score), **Specificity (Spe.)** (Neighborhood score). We follow prior work ([Meng et al., 2023](#); [Fang et al., 2025](#)), and provide for CounterFact the success score of each criterion (whether the new answer o achieves higher probability compared to ground truth answer) and for ZSRE we report the metric accuracy (whether the desired answer receives the highest probability). In addition, we provide for the harmonic scores of these three criteria, known as **Editing Score**. Following [Meng et al. \(2023\)](#), we also provide for CounterFact a generation-based **Fluency (Flu.)** metric, defined as a weighted combination of bi-gram and tri-gram generation entropy.

Following prior work ([Meng et al., 2023](#); [Fang et al., 2025](#)) we consider two evaluation paradigms: (i) one-shot editing, where 10,000 relations are edited simultaneously, and (ii) sequential (batch) editing, where 2,000 relations are processed in batches of 100 across 20 iterations. Tables 1 and 2 show that PFT achieves the strongest overall performance on CounterFact in both one-shot and sequential editing. In the one-shot setting, PFT outperforms the best baseline by 1–8 editing-score points across the three models, while in

Method	LLM	CounterFact (success)					ZSRE (accuracy)			
		Eff.	Gen.	Spe.	Flu.	Score	Eff.	Gen.	Spe.	Score
Pre-Edited		15.2	17.6	83.5	618.5	22.3	26.4	25.8	27.0	26.4
ROME	GPT-J	48.7	49.5	51.1	563.9	49.8	19.3	17.8	0.9	2.4
MEMIT		99.0	88.7	73.8	615.3	85.9	95.4	89.4	26.4	50.4
PRUNE		85.5	82.0	56.8	557.7	72.3	30.4	29.3	15.8	23.0
RECT		88.8	64.8	79.2	619.0	76.3	71.5	63.1	26.6	44.4
NSE		92.8	73.6	<u>76.5</u>	603.0	80.1	44.8	43.3	25.8	35.7
AlphaEdit		<u>99.1</u>	86.5	68.5	589.8	82.8	87.0	76.5	22.3	43.2
FT (old)		97.9	72.3	34.5	591.3	56.6	0.0	0.0	0.0	0.0
FT (updated)		100.0	<u>95.0</u>	47.1	417.9	71.9	<u>99.8</u>	<u>98.2</u>	3.9	10.8
PFT w/o smooth		66.5	62.3	52.2	505.4	59.7	0.1	0.1	0.0	0.1
PFT w/o s.VJPs		98.9	74.3	50.4	535.1	69.1	99.9	98.5	25.9	51.0
PFT		99.0	95.9	71.7	607.6	87.0	<u>99.8</u>	97.5	24.5	49.1
Pre-Edited		21.8	24.2	78.3	613.0	30.0	22.2	21.3	24.2	22.5
ROME	GPT2-XL	50.3	48.9	51.9	598.9	50.3	19.5	17.4	1.3	3.5
MEMIT		79.5	65.5	69.9	<u>609.1</u>	71.2	63.9	59.3	26.0	42.3
PRUNE		84.4	77.2	58.2	595.1	71.5	30.8	29.1	16.0	23.2
RECT		62.5	48.7	74.6	610.4	60.1	49.3	45.4	25.7	37.0
NSE		74.0	59.3	<u>70.1</u>	554.1	67.2	34.9	33.1	22.5	29.0
AlphaEdit		93.1	77.0	64.6	587.0	<u>76.5</u>	57.7	52.7	20.8	35.5
FT (old)		98.3	64.3	47.2	546.2	64.0	57.8	54.5	6.3	15.4
FT (updated)		82.0	<u>79.6</u>	52.8	327.3	68.7	91.9	90.0	17.1	37.4
PFT w/o smooth		58.3	53.4	52.5	567.6	54.6	15.1	13.8	0.3	0.8
PFT w/o s.VJPs		100.0	61.9	50.5	575.3	65.3	99.9	92.8	26.8	51.7
PFT		<u>99.0</u>	93.1	63.6	591.1	82.0	<u>99.7</u>	<u>90.4</u>	24.2	<u>48.1</u>
Pre-Edited		6.9	9.3	89.5	623.5	11.4	35.7	34.8	31.8	34.0
ROME	Llama-3-8B	50.0	50.0	48.4	448.6	49.5	0.0	0.0	23.4	0.0
MEMIT		94.4	76.0	82.0	614.8	<u>83.5</u>	86.9	82.8	32.5	55.2
PRUNE		93.8	83.3	59.6	599.5	76.0	21.4	18.9	18.1	19.3
RECT		60.9	42.3	87.6	618.9	58.3	71.7	66.6	32.3	50.0
NSE		75.6	47.3	<u>86.7</u>	590.2	65.4	45.9	45.7	31.5	39.8
AlphaEdit		90.6	72.4	81.6	613.3	80.9	84.8	80.8	32.5	54.6
FT (old)		100.0	91.7	35.7	410.6	61.3	63.9	58.9	45.0	54.7
FT (updated)		100.0	97.0	43.0	100.8	68.8	99.8	97.8	12.3	29.6
PFT w/o smooth		99.0	92.2	51.7	204.8	74.5	93.7	87.4	4.0	10.9
PFT w/o s.VJPs		100.0	73.2	52.4	439.0	70.2	<u>96.3</u>	<u>92.6</u>	<u>34.2</u>	59.5
PFT		98.7	<u>94.1</u>	84.7	<u>618.3</u>	92.1	91.2	84.1	31.8	<u>55.2</u>

Table 1: Knowledge editing of 10,000 relations for CounterFact and ZSRE. The best scores are highlighted in bold and the second best are underlined. The colored rows highlight our method, PFT, and its ablations.

the sequential setting its advantage remains consistent at 1–6 points. The strongest overall results are obtained on Llama-3, where PFT reaches an editing score of 92.1 in one-shot editing and 93.9 in sequential editing, while exhibiting almost no degradation in fluency. Across models, the performance gains of PFT are driven primarily by consistently stronger generalization, while maintaining high specificity.

On ZSRE, PFT remains highly competitive, although its advantage is smaller. Across both one-shot and sequential editing, PFT consistently belongs to the top group of methods and is typically ranked first or second. For example, in the sequential setup, PFT achieves a mean editing score that exceeds AlphaEdit by 0.3 points on GPT-J and Llama-3, while trailing by only 0.1 points on GPT2-XL.

Ablation Study The comparison among the PFT ablations further clarifies the source of PFT’s gains; adding selective VJPs alone, *PFT w/o smooth*, does not consistently improve over the updated FT baseline, although it often leads to better fluency, suggesting reduced

Method	LLM	CounterFact (success)					ZSRE (accuracy)			
		Eff.	Gen.	Spe.	Flu.	Score	Eff.	Gen.	Spe.	Score
Pre-Edited		15.8	18.1	83.4	617.1	22.9	27.8	27.1	27.5	27.5
ROME	GPT-J	57.9	56.6	50.6	535.3	54.8	65.8	64.1	16.6	32.9
MEMIT		99.0	91.0	77.9	616.8	88.4	97.4	94.5	29.3	54.6
PRUNE		91.8	87.7	62.2	575.2	78.2	53.7	50.5	24.0	37.5
RECT		97.7	78.9	78.3	<u>617.2</u>	84.1	96.0	91.6	<u>28.5</u>	53.2
NSE		93.8	75.6	80.9	605.9	82.8	45.3	44.2	27.1	36.8
AlphaEdit		99.8	<u>94.3</u>	78.1	615.6	<u>89.7</u>	99.3	96.2	27.9	53.3
FT (old)		75.2	63.8	36.5	559.7	53.2	0.0	0.0	0.0	0.0
FT (updated)		65.2	52.9	46.3	333.1	53.7	9.8	8.6	0.4	1.0
PFT w/o smooth		54.2	51.3	49.8	579.5	51.7	0.4	0.4	0.0	0.1
PFT w/o s.VJPs		76.1	69.6	58.5	597.5	67.3	96.2	93.8	24.6	48.6
PFT		<u>99.2</u>	97.2	<u>78.8</u>	617.4	90.8	<u>97.9</u>	<u>95.2</u>	28.4	<u>53.6</u>
Pre-Edited		22.1	24.4	78.0	611.3	30.3	23.7	22.8	25.0	23.8
ROME	GPT2-XL	56.4	52.4	51.8	613.1	53.4	46.9	42.4	14.5	26.3
MEMIT		92.1	79.0	73.1	606.0	80.7	85.3	79.8	27.1	49.0
PRUNE		88.2	84.5	63.0	589.3	76.8	36.5	34.1	20.6	28.5
RECT		86.8	71.5	<u>73.8</u>	<u>610.9</u>	76.8	83.1	75.4	24.5	45.4
NSE		82.6	68.8	74.9	575.6	75.0	39.3	37.8	24.9	32.6
AlphaEdit		99.2	<u>91.2</u>	69.4	603.0	<u>84.6</u>	92.4	85.9	25.6	48.7
FT (old)		66.2	57.0	52.2	573.5	57.9	30.6	28.4	1.7	4.5
FT (updated)		46.9	48.5	54.4	459.6	49.7	10.9	11.4	2.7	5.4
PFT w/o smooth		50.6	50.0	51.5	530.7	50.7	4.7	4.7	0.6	1.4
PFT w/o s.VJPs		88.0	68.0	64.8	601.3	72.3	<u>93.8</u>	87.3	<u>25.9</u>	49.5
PFT		<u>98.7</u>	94.2	70.3	595.7	85.8	94.2	<u>87.2</u>	25.2	48.6
Pre-Edited		7.7	10.5	89.3	620.4	12.6	37.0	36.4	31.8	34.9
ROME	Llama-3-8B	48.5	48.1	49.2	354.5	48.6	0.0	0.0	19.9	0.0
MEMIT		94.5	89.3	67.4	593.4	82.0	89.6	85.2	32.3	55.7
PRUNE		82.4	65.3	55.3	577.9	65.9	26.8	24.2	17.4	22.1
RECT		94.5	84.4	72.8	601.8	83.0	87.6	82.8	32.3	55.1
NSE		86.7	56.7	88.5	591.4	74.1	47.0	47.3	32.3	40.9
AlphaEdit		<u>98.8</u>	<u>91.4</u>	75.1	<u>606.5</u>	<u>87.3</u>	93.4	89.1	32.5	56.9
FT (old)		92.7	83.5	43.0	408.3	65.2	41.5	38.7	41.4	40.5
FT (updated)		92.5	87.0	49.5	268.5	70.6	67.9	65.8	7.6	18.6
PFT w/o smooth		87.5	79.2	49.0	296.3	67.4	21.4	20.1	1.7	4.5
PFT w/o s.VJPs		91.0	71.7	71.2	599.0	77.0	98.7	95.2	<u>34.1</u>	60.0
PFT		98.9	96.4	<u>87.3</u>	619.0	93.9	<u>96.3</u>	<u>93.3</u>	32.0	<u>57.3</u>

Table 2: Sequential knowledge editing using CounterFact and ZSRE: editing 2,000 relations across 20 iterations, 100 relations at a time. The colored rows highlight our method, PFT, and its ablations.

overfitting to the edited examples. In particular, across all settings, naive FT yields the weakest fluency scores, indicating that it tends to memorize the new edits at the expense of the models’ original capabilities. An even improved fluency score is presented by the variant that only utilizes smoothing, *PFT w/o s.VJPs*, which also achieves the strongest average results on ZSRE with GPT2 and Llama-3, highlighting the importance of smoothing as a central component of robust editing. However, the fact that *PFT w/o s.VJPs* achieves strong ZSRE scores despite relatively poor fluency, such as 439 with Llama-3, suggests that the ZSRE metrics are less sensitive to broader model degradation than the CounterFact evaluation.

Post-Editing Weight-Space Analysis. To complement the empirical analysis, we also measure the parameter changes induced by different editing methods using Llama-3-8B. After sequentially editing, we compute the difference between each edited W_2 matrix and its pre-edit value, and report the largest singular value of this difference at each of the five edited layers. The values are 15.4, 25.2, 30.5, 32.3, 49.3 for MEMIT; 1.0, 1.0, 1.1, 1.7, 3.6 for

AlphaEdit; and 2.5, 1.2, 0.6, 0.4, 0.3 for PFT. The lower average and maximum singular values suggest that PFT achieves its edits through smaller changes along the principal directions of the weight-space update. From an interpretability perspective, this analysis also highlights a meaningful difference between PFT and previous methods: MEMIT and AlphaEdit learn interventions in the residual stream after the last layer they aim to edit, placing increasing weight on deeper layers. In contrast, PFT places relatively more weight on earlier edited layers. This behavior is consistent with the VJP norms shown in Figure 1, which themselves can be viewed as a gradient-based attribution signal (Achtibat et al., 2024).

We further evaluate PFT in Appendix C, where we (i) show that it preserves the original model’s capabilities at a level comparable to methods such as AlphaEdit on GLUE (Wang et al., 2018), (ii) analyze PFT runtime, showing that PFT remains efficient, matching the practical scalability of prior editing methods, and (iii) we extend our evaluation to six additional models from the Llama-3.2 and Qwen2.5 families, ranging from 1B to 14B parameters, demonstrating that PFT generalizes across model families and scales.

7 Conclusions, Limitations and Future Work

In this work, we leveraged insights from mechanistic interpretability to analyze failure modes of FT, showing how multiple error signals can encode unintended information and interfere with one another. Building on these observations, we introduced PFT, a knowledge editing method that integrates selective smoothing into the FT pipeline. Our analysis highlights the connection between this smoothing mechanism and closed-form editing approaches, while demonstrating how it can be applied efficiently and iteratively within FT.

Limitations Most of the hyperparameters used by PFT were adopted from prior work to ensure fair comparison, rather than being optimized for PFT, as detailed in Appendix B. For this reason PFT edits only the second MLP matrix W_2 , suggesting that further gains may be achieved by tuning PFT’s hyperparameters and its layer selection.

Regarding our knowledge editing evaluation, we have used established benchmarks such as CounterFact and ZSRE (Meng et al., 2022; Levy et al., 2017). Prior studies have highlighted limitations of such datasets to fully capture generalization and broader impacts on model behavior (Cohen et al., 2024; Gu et al., 2024). Despite these limitations, there is currently no widely accepted alternative evaluation framework. As a result, existing benchmarks are used as proxies for broader real-world scenarios. Accordingly, we follow standard evaluation protocols to ensure comparability with prior work.

Future Work A limitation of PFT is its reliance on structured editing settings in which the subject and target are explicitly identifiable. During the review process, we explored extending PFT to unstructured knowledge-editing settings, including UnKEBench (Deng et al., 2025) and AKEW (MQUAKE) (Jiang et al., 2025a; Wu et al., 2024; Zhong et al., 2023). Such settings require modifying the VJP selection procedure and editing scope employed by both our method and the competing approaches considered in our work. We therefore report these preliminary results on OpenReview¹ and leave a systematic investigation of unstructured knowledge editing for future work.

Acknowledgments

This work was supported by a Tel Aviv University Center for AI and Data Science (TAD) grant and by Len Blavatnik and the Blavatnik Family Foundation. This research was also supported by the Ministry of Innovation, Science & Technology, Israel (1001576154) and the Michael J. Fox Foundation (MJFF-022407). SK is supported by the Google PhD Fellowship. The contribution of SK is part of a PhD thesis research conducted at Tel Aviv University.

¹<https://openreview.net/forum?id=8UbvMbX5PQ>

Ethics Statement

This work addresses the problem of knowledge editing in LLMs, with the goal of enabling precise correction of false or outdated factual associations. The ability to update model knowledge in a targeted manner has clear societal benefits, including the removal of harmful misinformation and the correction of factual errors that could mislead users.

At the same time, we acknowledge that knowledge editing techniques carry potential for misuse. Methods that efficiently modify factual associations in LLMs could, in principle, be employed to deliberately inject false information or biased content into deployed models. We encourage the research community and practitioners to consider the governance implications of such capabilities and to develop appropriate safeguards.

References

- Reduan Achibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In Forty-first International Conference on Machine Learning, 2024.
- David Bau, Steven Liu, Tongzhou Wang, Jun-Yan Zhu, and Antonio Torralba. Rewriting a deep generative model. In European conference on computer vision, pp. 351–369. Springer, 2020.
- Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. TAC, 7(8):1, 2009.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.
- Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. Transactions of the Association for Computational Linguistics, 12:283–298, 2024.
- Jingcheng Deng, Zihao Wei, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. Everything is editable: Extend knowledge editing to unstructured data in large language models. In International Conference on Learning Representations, volume 2025, pp. 628–652, 2025.
- William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In Proceedings of the third international workshop on paraphrasing (IWP2005), 2005.
- N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits, 2021. URL <https://transformer-circuits.pub/2021/framework/index.html>.
- Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Alphaedit: Null-space constrained model editing for language models. In The Thirteenth International Conference on Learning Representations, 2025.
- Govind Krishnan Gangadhar and Karl Stratos. Model editing by standard fine-tuning. In Findings of the Association for Computational Linguistics: ACL 2024, pp. 5907–5913, 2024.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 5484–5495, 2021.

- Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL <https://aclanthology.org/2023.emnlp-main.751>.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.934. URL <https://aclanthology.org/2024.emnlp-main.934/>.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3476–3503, 2024.
- Houcheng Jiang, Junfeng Fang, Ningyu Zhang, Mingyang Wan, Guojun Ma, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Anyedit: Edit any knowledge encoded in language models. In *Forty-second International Conference on Machine Learning*, 2025a. URL <https://openreview.net/forum?id=aJI0Bur0Ef>.
- Houcheng Jiang, Junfeng Fang, Tianyu Zhang, Baolong Bi, An Zhang, Ruipeng Wang, Tao Liang, and Xiang Wang. 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)*, pp. 16678–16702, 2025b.
- Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. From tokens to words: On the inner lexicon of llms. In *International Conference on Learning Representations*, volume 2025, pp. 70068–70092, 2025.
- Shahar Katz and Lior Wolf. Reversed attention: On the gradient descent of attention layers in gpt. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1125–1152, 2025.
- Shahar Katz, Yonatan Belinkov, Mor Geva, and Lior Wolf. Backward lens: Projecting language model gradients into the vocabulary space. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2390–2422, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.142. URL <https://aclanthology.org/2024.emnlp-main.142>.
- DP Kingma. Adam: a method for stochastic optimization. In *Int Conf Learn Represent*, 2014.
- Serge Lang. *Introduction to Linear Algebra*. Undergraduate Texts in Mathematics. Springer New York, 2012. ISBN 9781461210702. URL https://books.google.co.il/books?id=c_NEBAAAQBAJ.
- Y Le Cun. A theoretical framework for backpropagation. In *Proceedings of the 1988 Connectionist Models Summer School*, 1988.

- Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In Roger Levy and Lucia Specia (eds.), *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 333–342, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/K17-1034. URL <https://aclanthology.org/K17-1034/>.
- Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025. URL <https://arxiv.org/abs/2502.16982>.
- Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *International Conference on Learning Representations*, volume 2025, pp. 17302–17329, 2025.
- Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. Memory-assisted prompt editing to improve gpt-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2833–2861, 2022.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36, 2022.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *International Conference on Learning Representations*, 2023.
- Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2021.
- Victor Morand, Josiane Mothe, and Benjamin Piwowarski. On the representations of entities in auto-regressive large language models. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 433–451, Suzhou, China, 2025. Association for Computational Linguistics. ISBN 979-8-89176-346-3. doi: 10.18653/v1/2025.blackboxnlp-1.25.
- Judea Pearl. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pp. 373–392. 2022.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.
- Charles F Van Loan and G Golub. Matrix computations (johns hopkins studies in mathematical sciences). *Matrix Computations*, 5:32, 1996.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 63–76, 2019.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. *Advances in Neural Information Processing Systems*, 33:12388–12401, 2020.

- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP, pp. 353–355, 2018.
- Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.
- Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. arXiv preprint arXiv:1805.12471, 2018.
- Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.
- Xiaobao Wu, Liangming Pan, William Yang Wang, and Anh Tuan Luu. AKEW: Assessing knowledge editing in the wild. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pp. 15118–15133, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.843.
- Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. Editing large language models: Problems, methods, and opportunities. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 10222–10240, 2023.
- Lang Yu, Qin Chen, Jie Zhou, and Liang He. Melo: Enhancing model editing with neuron-indexed dynamic lora. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pp. 19449–19457, 2024.
- Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. arXiv preprint arXiv:2401.01286, 2024.
- Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 15686–15702, 2023.
- Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. Modifying memories in transformer models. ArXiv, abs/2012.00363, 2020.

A Model Editing via Locate-and-Edit, Closed-Form Solutions

In this section, we briefly review how the normal equation (Lang, 2012) has been used as a closed-form solution in locate-and-edit methods. This formulation shares similarities in both structure and effect with PFT’s smoothing projection. We discuss these connections and highlight the key differences between its use in prior methods and its role within PFT. We follow the notation used by MEMIT and AlphaEdit (Meng et al., 2023; Fang et al., 2025), referring the reader to the original works for further details. In Table 3 we summarize the main differences between MEMIT-inspired methods.

Method	Constraint	Interpretation
ROME (Meng et al., 2022)	rank-one edit	edits a single fact by applying a rank-one update to a selected MLP layer
MEMIT (Meng et al., 2023)	multi-layer direct editing	extends ROME-style editing across multiple MLP layers to edit many relations simultaneously
RECT (Gu et al., 2024)	MEMIT + relative-change regularization	regularizes the edit update by retaining principal weight changes and suppressing smaller relative perturbations
PRUNE (Ma et al., 2025)	MEMIT + condition-number / spectral restraint	restrains singular values during sequential editing to limit perturbation growth and preserve general abilities
NSE (Jiang et al., 2025b)	MEMIT + neuron-level sequential editing	performs sequential editing by selecting and updating only selected activated neurons across multiple layers while computing target representations with the original weights
AlphaEdit (Fang et al., 2025)	MEMIT + null-space projection	projects the edit update onto the null space of the layer’s covariance matrix (preserved knowledge) before applying it

Table 3: Comparison of locate-and-edit, closed-solution editing methods.

The formulation of the MLP in Equation 1 suggests its key-value structure. Editing methods such as MEMIT modify only the second projection matrices of the MLP layers, $W_2 \in \mathbb{R}^{d \times d_2}$. Given a relation $(s, r, o)_i$, such methods proceed as follows: (i) they collect the average activation $k_i \in \mathbb{R}^{d_2}$ (key) corresponding to the last token of the subject s , (ii) they learn a value vector $v_i \in \mathbb{R}^d$ such that if the residual stream of that layer plot v_i - the output of the model will be o . (iii) Finally, they compute a closed-form update that encodes v_i into W_2 while minimizing deviations from the original model behavior.

To balance these objectives, ROME additionally collects key-value pairs $(\hat{k}, \hat{v})$ from general text, such as random sentences from Wikipedia, representing pairs of model’s original inputs and outputs. Let K_0, V_0 denote n_0 such original pairs, and K_1, V_1 denote the n_1 edited relations:

$$K_0 = [\hat{k}_1 \mid \hat{k}_2 \mid \dots \mid \hat{k}_{n_0}] \in \mathbb{R}^{d_2 \times n_0}, \quad V_0 = [\hat{v}_1 \mid \hat{v}_2 \mid \dots \mid \hat{v}_{n_0}] \in \mathbb{R}^{d \times n_0}, \quad (9)$$

$$K_1 = [k_1 \mid k_2 \mid \dots \mid k_{n_1}] \in \mathbb{R}^{d_2 \times n_1}, \quad V_1 = [v_1 \mid v_2 \mid \dots \mid v_{n_1}] \in \mathbb{R}^{d \times n_1}. \quad (10)$$

The update is obtained by solving the following optimization problem via:

$$\Delta = \arg \min_{\Delta} \left(\|(W_2 + \Delta)K_1 - V_1\|^2 + \|(W_2 + \Delta)K_0 - V_0\|^2 \right), \quad (11)$$

$$\Delta = (V_1 - W_2 K_1) K_1^T \left(K_0 K_0^T + K_1 K_1^T \right)^{-1}, \quad (12)$$

and the updated weights are given by $W_2 \leftarrow W_2 + \Delta$.

Aspect	MEMIT	PFT (Ours)
Optimization target		
Learned vectors	Value vector v_i	VJPs δ_i
Sharing optimization	Single target shared by all augmentations of optimized association	Individual for every augmentation (prompt)
Amount	1	Up to the length of the subject suffix
Optimization steps	Consecutive iterations	Global epochs
Optimized in parallel to other optimized target	No	Yes
Order of execution	Consecutive, optimize one association after another	Mixed; optimized multiple association in one batch
Smoothing		
Smoothed operator	Mean activations K_1	Activations X
Token affected	Last subject token	Subject suffix tokens
Shared by	Shared by all augmentations of optimized association	Individual for every augmentation (prompt)
Smoothing matrix	$M = K_0 K_0^\top + K_1 K_1^\top$	$M = K_0 K_0^\top + I$
Handling of correlations	Implicit	Explicit
Matrix inversion	Least squares	LU-decomposition
Update formula per sample i	$(v_i - W_2 k_i^\top) k_i^\top M^{-1}$	$\delta_i x_i^\top M^{-1}$
Update formula (batch)	$(V_1 - W_2 K_1) K_1^\top M^{-1}$	$\frac{\partial \mathcal{L}}{\partial (W X^\top)} X^\top M^{-1}$
Update procedure		
Maximum rank of the change induce by one association	1	number of epochs $\times$ subject suffix length
Maximum rank of the change after updating n relations	n	$n \times$ number of epochs $\times$ subject suffix length
Update	Two-stage: estimate all targets, then apply a single global update	Iterative: updates applied incrementally through FT steps
Batching across associations	No; processes associations independently	Natural; supports mixing multiple relations within batches
Adaptability / flexibility	Requires recomputing targets for new edits	Easily extends to continuous and mixed-data editing
Implementation	Requires caching keys/values and solving linear systems	Extension of standard FT

Table 4: Comparison between the locate-and-edit approach of MEMIT and PFT.

This update can be decomposed and interpreted as the product of two components: the value-based term, $(V_1 - W_2 K_1)$, and a smoothed key term, $K_{\text{smooth}} = K_1^\top (K_0 K_0^\top + K_1 K_1^\top)^{-1}$.

Similarity between ROME and PFT. We recall the smoothing projection used in PFT:

$$X_{\text{smooth}} = X^\top (K_0 K_0^\top + I)^{-1} = X^\top M^{-1}. \quad (13)$$

PFT’s smoothed activations X_{smooth} resemble MEMIT’s smoothed keys K_{smooth} , as both project inputs (token activations in PFT and averaged keys in MEMIT) through the covariance matrix $K_0 K_0^\top$. A key difference is that MEMIT additionally incorporates K_1 inside the inverse, whereas in PFT this term is not explicitly available during iterative FT. However, in PFT we assume that the activations observed during each FT step, $X \in \mathbb{R}^{d_2 \times n}$, contain significantly fewer vectors than $K_0 \in \mathbb{R}^{d_2 \times n_0}$. For example, MEMIT typically uses $n_0 \approx 10^5$, while in FT n is on the order of tens. Since both sets of vectors originate from the same layer and have comparable scales, it follows that

$$\|K_0 K_0^\top\| \gg \|XX^\top\|. \quad (14)$$

This allows us to approximate

$$K_0 K_0^\top + XX^\top \approx K_0 K_0^\top, \quad (15)$$

highlighting the connection between MEMIT and PFT smoothing.

We now consider the value-based component [Equation 11](#), $(V_1 - W_2 K_1)$. Its analogue in PFT is given by the VJPs δ_i (see [Equation 2](#)), which act as token-level optimization targets. In this section we refer to both quantities as optimization targets, as both are obtained through gradient-based learning optimization across multiple steps. The key differences between the two optimization targets are as follows:

- In closed-form methods, the optimization target is shared across a relation and its augmentations. In PFT, each prompt induces its own target via its corresponding VJPs.
- Closed-form solutions constrain the update to rank-one structure, whereas PFT allows higher-rank updates determined by the number of selected tokens and the number of iterations.
- Closed-form methods first estimate optimization targets for each relation and then apply a global update once. In contrast, PFT updates the model incrementally through iterative FT steps and multiple iterations.

These differences highlight the flexibility of PFT: while both approaches aim to encode new information into the model, PFT naturally supports iterative updates and partial data availability, making it more suitable for scalable and dynamic editing scenarios. To summarize PFT and MEMIT-inspired methods, see [Table 4](#).

B Extended Implementation Details

This section provides full implementation details for PFT, complementing the method description in section 5. Figure 3 provides an overview illustration of PFT.

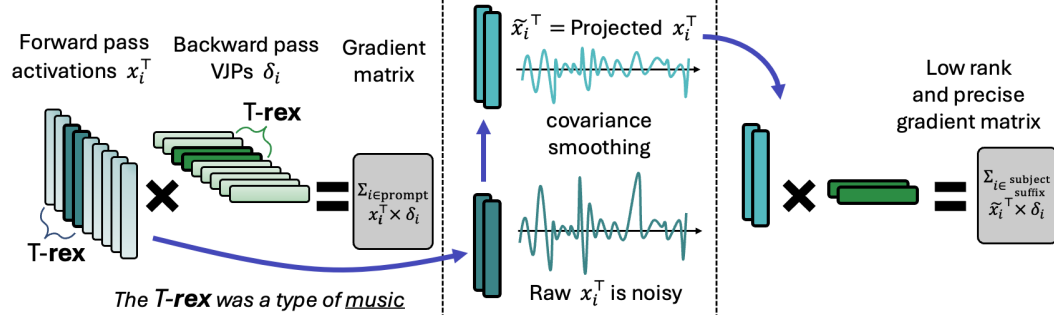


Figure 3: An illustration depicting Precise-FT (PFT). The error signals of naive FT, the VJPs, are per token and parameter signals, which are summed up together into the gradient matrix. PFT filters and smooths the VJPs, resulting in better knowledge editing performance.

Augmentations. Following prior work such as ROME (Meng et al., 2022), MEMIT (Meng et al., 2023), and AlphaEdit (Fang et al., 2025), we augment each edited relation during fine-tuning. Given a target relation (s, r, o) , we perform FT not only on the original prompt, but also on five augmented variants constructed by appending randomly sampled prefixes generated by the model. This improves generalization across paraphrases and reduces overfitting to a single prompt formulation.

Sample Shuffling. Given the full set of edited relations, including their augmented variants, we randomly shuffle all samples before each epoch. This ensures that updates are applied in a diverse order and reduces bias introduced by fixed training sequences.

Per-Sample Probability Monitoring. During each epoch, we monitor the probability assigned to the target object o for each sample, instead of relying solely on the average loss. To accelerate training, we define a probability threshold $T \in [0, 1]$: once a sample exceeds this threshold, it is temporarily removed from subsequent optimization steps. When the number of remaining samples falls below 5% of the original set at the beginning of an epoch, we reintroduce all samples and continue training. This procedure allows focuses computation on unresolved edits while maintaining stability across epochs.

For completeness, we briefly provide the following equations to emphasize the equivalence of monitoring per-sample loss and the probability of its target answer o . Using the notation from section 3, where p is the output probability vector predicted by the model:

$$\mathcal{L} = NLL(p, o) = -\log p[o] \quad (16)$$

$$p[o] \geq T \iff -\log p[o] \leq -\log T, \quad p[o] \geq T \iff \mathcal{L} = NLL(p, o) \leq -\log T \quad (17)$$

Learning Rate. Each model uses a single base learning rate η_0 across all experiments in section 6. We further adapt this learning rate in two ways: (i) we apply a decay factor of 0.95 at the end of each epoch, and (ii) we scale the initial learning rate based on the number of edited relations N using a logarithmic schedule:

$$\eta = \eta_0 \left(\frac{N}{100} \right)^\alpha, \quad \alpha = \log(0.5). \quad (18)$$

This scaling reflects the need for smaller updates when editing larger sets of relations.

Optimizer. PFT uses vanilla stochastic gradient descent (SGD), which enables direct and fine-grained control over gradient updates. This choice supports precise, localized modifications at the neuron level, avoiding cross-relation interference that can arise from more complex optimizer dynamics.

Additional Parameters. A summary of all hyperparameters (params) used in our experiments is provided in Table 5. This includes standard fine-tuning configurations such as weight clipping, data types, and hardware settings. The table explains how each hyperparameter was selected: most of them were taken from MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2025) and were not optimized by us. Other hyperparameters were selected using grid search as part of the ablation study in subsection C.6.

Hyperparameter	GPT-J	GPT2-XL	Llama-3-8B	Param justification
Epochs	25	25	25	Match the number of optimization steps used by MEMIT/AlphaEdit
Number of augmentations	5	5	5	Match the number of augmentations used by MEMIT
Covariance update weight λ_0 (section 5)	150000	200000	150000	Taken from MEMIT/AlphaEdit
Number of samples in the construction of the covariance matrices	100000	100000	100000	Taken from MEMIT/AlphaEdit
Covariance update interval (steps)	1500	1500	1500	Grid search
Number of VJPs per optimized prompt (suffix length)	4	4	4	Grid search
Initial weights clipping range	0.0005	0.0005	0.0005	In order to make this param adaptable to each matrix, this actual clipping range is the product of the optimized matrix and this value
Initial learning rate η_0	0.7	1.55	0.00175	Grid search
Learning rate decay per epoch	0.95	0.95	0.95	
Batch size	24	24	24	
Number of optimized layers	5	5	5	Matches or uses fewer layers than MEMIT/AlphaEdit
Layer indices	1 – 5	10 – 14	1 – 5	1–3 layers before the ones optimized by MEMIT/AlphaEdit
Probability threshold T	0.8	0.7	0.99	Grid search
Data type	FP16	FP32	BF16	
Hardware (Nvidia GPU)	L40s (48GB)	A6000 (48GB)	A100 (80GB)	Note that in practice, all runs consume less than 48GB

Table 5: Hyperparameter configuration across models.

C Extended Experiments

In [section 6](#), we presented the main experimental results comparing PFT to existing knowledge editing methods. In this section, we provide additional analyses and evaluations for completeness. We begin by extending our interpretability-based analysis from [subsection 4.2](#), providing results across all evaluated models. We then revisit the main CounterFact ([Meng et al., 2022](#)) and ZSRE ([Levy et al., 2017](#)) results, including additional metrics and runtime comparisons, demonstrating that PFT remains both effective and computationally efficient. Overall, these extended experiments further support the effectiveness and robustness of PFT, reinforcing its position as a strong method for knowledge editing.

C.1 Extended Analysis of Correlation Between Different Tokens

In [subsection 4.2](#), we showed that activations corresponding to different tokens exhibit non-negligible correlations. During FT, these activations are encoded into the model parameters, leading to unintended interactions across prompts with correlated representations.

While the main text illustrated this phenomenon for a single model, [Figure 4](#) extends this analysis to all three models considered in this work: GPT-J, GPT2-XL, and Llama-3. The figure reports cosine similarity between the final subject tokens across the first 10 layers, showing consistent correlations in the range of 0.05 to 0.25 across models and layers. Applying covariance smoothing projection reduces these correlations to near orthogonality, providing empirical support for the mechanisms underlying PFT.

We further repeat this analysis for additional token positions. For example, [Figure 5](#) presents the correlation between the last three tokens of subject suffixes, revealing similar patterns. These consistent trends suggest that the observed correlations reflect a shared geometric structure in the activation spaces of transformer layers.

Taken together, these results highlight a fundamental limitation of naïve FT: correlated activations lead to unintended coupling between edits. This observation directly motivates the use of smoothing and projection in PFT to reduce such interference.

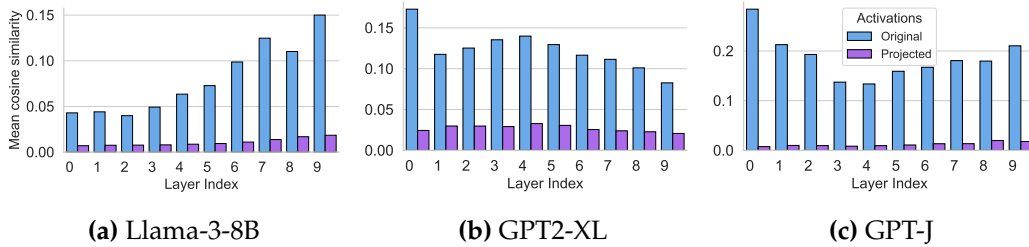


Figure 4: Cosine similarity between forward pass activations of last subject tokens, before and after projection.

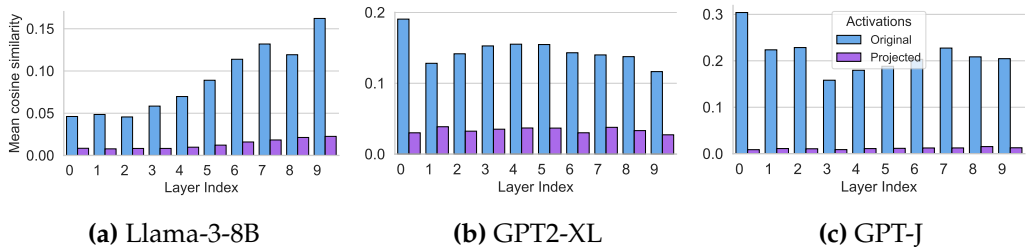


Figure 5: Cosine similarity between forward pass activations of the third-from-last subject tokens, before and after projections.

Method	LLM	CounterFact (success)				
		Eff.	Gen.	Spe.	Flu.	Score
Pre-Edited		15.2 ± 35.9	17.6 ± 33.0	83.5 ± 25.1	618.5 ± 15.1	22.3
ROME	GPT-J	48.7 ± 50.0	49.5 ± 39.3	51.1 ± 28.5	563.9 ± 23.9	49.8
MEMIT		99.0 ± 9.8	88.7 ± 27.0	73.8 ± 26.9	615.3 ± 17.9	85.9
PRUNE		85.5 ± 35.2	82.0 ± 32.3	56.8 ± 31.4	557.7 ± 26.0	72.3
RECT		88.8 ± 31.5	64.8 ± 42.1	79.2 ± 25.9	619.0 ± 14.3	76.3
NSE		92.8 ± 25.8	73.6 ± 38.5	76.5 ± 25.7	603.0 ± 24.6	80.1
AlphaEdit		<u>99.1 ± 9.3</u>	86.5 ± 28.9	68.5 ± 28.6	589.8 ± 23.4	82.8
FT (old)		97.9 ± 14.2	72.3 ± 36.2	34.5 ± 29.8	591.3 ± 16.6	56.6
FT (updated)		100.0 ± 2.2	95.0 ± 17.5	47.1 ± 31.7	417.9 ± 40.6	71.9
PFT w/o smooth		66.5 ± 47.2	62.3 ± 41.8	52.2 ± 34.3	505.4 ± 54.5	59.7
PFT w/o s.VJPs		98.9 ± 10.2	74.3 ± 35.1	50.4 ± 29.3	535.1 ± 27.6	69.1
PFT		99.0 ± 9.9	95.9 ± 16.7	71.7 ± 27.6	607.6 ± 24.8	87.0
Pre-Edited		21.8 ± 41.3	24.2 ± 36.9	78.3 ± 28.1	613.0 ± 21.6	30.0
ROME	GPT2-XL	50.3 ± 50.0	48.9 ± 40.2	51.9 ± 31.9	598.9 ± 5.8	50.3
MEMIT		79.5 ± 40.4	65.5 ± 42.3	69.9 ± 28.6	609.1 ± 22.4	71.2
PRUNE		84.4 ± 36.3	77.2 ± 36.3	58.2 ± 31.5	595.1 ± 22.4	71.5
RECT		62.5 ± 48.4	48.7 ± 44.4	74.6 ± 27.7	610.4 ± 22.0	60.1
NSE		74.0 ± 43.9	59.3 ± 42.3	70.1 ± 27.4	554.1 ± 39.5	67.2
AlphaEdit		93.1 ± 25.3	77.0 ± 35.8	64.6 ± 29.1	587.0 ± 25.4	<u>76.5</u>
FT (old)		98.3 ± 12.7	64.3 ± 39.9	47.2 ± 33.3	546.2 ± 50.4	64.0
FT (updated)		82.0 ± 38.4	79.6 ± 36.0	52.8 ± 34.5	327.3 ± 60.2	68.7
PFT w/o smooth		58.3 ± 49.3	53.4 ± 46.6	52.5 ± 43.3	567.6 ± 16.9	54.6
PFT w/o s.VJPs		100.0	61.9 ± 39.2	50.5 ± 31.2	575.3 ± 32.3	65.3
PFT		<u>99.0 ± 10.1</u>	93.1 ± 21.1	63.6 ± 29.3	591.1 ± 30.2	82.0
Pre-Edited		6.9 ± 25.4	9.3 ± 25.0	89.5 ± 18.5	623.5 ± 15.3	11.4
ROME	Llama-3-8B	50.0 ± 50.0	50.0 ± 42.2	48.4 ± 35.3	448.6 ± 17.1	49.5
MEMIT		94.4 ± 22.9	76.0 ± 37.2	82.0 ± 22.1	614.8 ± 20.5	83.5
PRUNE		93.8 ± 24.2	83.3 ± 29.7	59.6 ± 28.7	599.5 ± 29.2	76.0
RECT		60.9 ± 48.8	42.3 ± 44.1	87.6 ± 19.2	618.9 ± 19.0	58.3
NSE		75.6 ± 43.0	47.3 ± 43.0	86.7 ± 18.6	590.2 ± 35.3	65.4
AlphaEdit		90.6 ± 29.2	72.4 ± 39.1	81.6 ± 22.5	613.3 ± 20.6	80.9
FT (old)		<u>100.0 ± 1.4</u>	91.7 ± 22.6	35.7 ± 35.6	410.6 ± 59.5	61.3
FT (updated)		100.0	97.0 ± 13.5	43.0 ± 30.0	100.8 ± 29.7	68.8
PFT w/o smooth		99.0 ± 10.0	92.2 ± 21.1	51.7 ± 27.4	204.8 ± 51.9	74.5
PFT w/o s.VJPs		100.0 ± 1.7	73.2 ± 35.1	52.4 ± 30.5	439.0 ± 36.8	70.2
PFT		98.7 ± 11.5	<u>94.1 ± 20.0</u>	84.7 ± 20.8	<u>618.3 ± 20.9</u>	92.1

Table 6: Knowledge editing of 10,000 CounterFact relations. The best scores are highlighted in bold and the second best are underlined.

C.2 Knowledge Editing Extended Evaluation

In [section 6](#), we evaluated PFT on CounterFact ([Meng et al., 2023](#)) under two settings: *one-shot editing*, where 10,000 relations are edited simultaneously, and *sequential (batch) editing*, where 2,000 relations are processed in batches of 100 over 20 iterations. In [Tables 6](#) and [Table 7](#) we provide these results including variance scores for CounterFact and [Tables 8](#) and [Table 9](#) for ZSRE.

In this section, we also extend this evaluation to a broader range of dataset sizes. [Figure 6](#) reports the *one-shot editing* Editing Score for CounterFact across varying numbers of relations: 100, 500, 1,000, 5,000, and 10,000. Similarly, [Figure 7](#) presents the CounterFact results for *sequential editing* with 1,500, 2,000, and 3,000 total edits.

Across all settings, PFT consistently achieves the best performance. PFT achieve similar or better editing scores for small sizes of datasets, such as 1,000 edits, and showing better performances with greater dataset, have only a small degradation in performances even

Method	LLM	CounterFact (success)				
		Eff.	Gen.	Spe.	Flu.	Score
Pre-Edited		15.8 $\pm$ 36.4	18.1 $\pm$ 33.8	83.4 $\pm$ 25.1	617.1 $\pm$ 17.8	22.9
ROME	GPT-J	57.9 $\pm$ 49.4	56.6 $\pm$ 40.6	50.6 $\pm$ 31.4	535.3 $\pm$ 30.1	54.8
MEMIT		99.0 $\pm$ 9.7	91.0 $\pm$ 24.9	77.9 $\pm$ 25.7	616.8 $\pm$ 16.9	88.4
PRUNE		91.8 $\pm$ 27.4	87.7 $\pm$ 27.6	62.2 $\pm$ 32.1	575.2 $\pm$ 39.1	78.2
RECT		97.7 $\pm$ 15.0	78.9 $\pm$ 34.7	78.3 $\pm$ 25.7	617.2 $\pm$ 17.1	84.1
NSE		93.8 $\pm$ 24.1	75.6 $\pm$ 37.9	80.9 $\pm$ 24.6	605.9 $\pm$ 27.0	82.8
AlphaEdit		99.8 $\pm$ 5.0	<u>94.3 $\pm$ 19.3</u>	78.1 $\pm$ 25.5	615.6 $\pm$ 19.4	<u>89.7</u>
FT (old)		75.2 $\pm$ 43.2	63.8 $\pm$ 42.9	36.5 $\pm$ 38.7	559.7 $\pm$ 14.2	53.2
FT (updated)		65.2 $\pm$ 47.6	52.9 $\pm$ 42.5	46.3 $\pm$ 36.7	333.1 $\pm$ 54.3	53.7
PFT w/o smooth		54.2 $\pm$ 49.8	51.3 $\pm$ 44.3	49.8 $\pm$ 37.8	579.5 $\pm$ 18.4	51.7
PFT w/o s.VJPs		76.1 $\pm$ 42.6	69.6 $\pm$ 38.6	58.5 $\pm$ 28.8	597.5 $\pm$ 21.2	67.3
PFT		<u>99.2 $\pm$ 9.2</u>	97.2 $\pm$ 14.5	<u>78.8 $\pm$ 25.7</u>	617.4 $\pm$ 16.4	90.8
Pre-Edited		22.1 $\pm$ 41.5	24.4 $\pm$ 36.9	78.0 $\pm$ 28.3	611.3 $\pm$ 21.2	30.3
ROME	GPT2-XL	56.4 $\pm$ 49.6	52.4 $\pm$ 40.6	51.8 $\pm$ 32.6	613.1 $\pm$ 8.8	53.4
MEMIT		92.1 $\pm$ 27.0	79.0 $\pm$ 35.6	73.1 $\pm$ 27.9	606.0 $\pm$ 26.7	80.7
PRUNE		88.2 $\pm$ 32.2	84.5 $\pm$ 31.2	63.0 $\pm$ 31.1	589.3 $\pm$ 32.7	76.8
RECT		86.8 $\pm$ 33.9	71.5 $\pm$ 40.2	<u>73.8 $\pm$ 27.8</u>	<u>610.9 $\pm$ 25.3</u>	76.8
NSE		82.6 $\pm$ 37.9	68.8 $\pm$ 40.1	74.9 $\pm$ 27.0	575.6 $\pm$ 40.0	75.0
AlphaEdit		99.2 $\pm$ 9.2	<u>91.2 $\pm$ 23.1</u>	69.4 $\pm$ 28.8	603.0 $\pm$ 27.5	<u>84.6</u>
FT (old)		66.2 $\pm$ 47.3	57.0 $\pm$ 42.4	52.2 $\pm$ 35.3	573.5 $\pm$ 51.5	57.9
FT (updated)		46.9 $\pm$ 49.9	48.5 $\pm$ 46.0	54.4 $\pm$ 38.6	459.6 $\pm$ 94.4	49.7
PFT w/o smooth		50.6 $\pm$ 50.0	50.0 $\pm$ 48.7	51.5 $\pm$ 47.1	530.7 $\pm$ 10.8	50.7
PFT w/o s.VJPs		88.0 $\pm$ 32.5	68.0 $\pm$ 38.6	64.8 $\pm$ 28.6	601.3 $\pm$ 22.0	72.3
PFT		<u>98.7 $\pm$ 11.3</u>	94.2 $\pm$ 19.8	70.3 $\pm$ 28.3	595.7 $\pm$ 37.2	85.8
Pre-Edited		7.7 $\pm$ 26.6	10.5 $\pm$ 26.8	89.3 $\pm$ 18.7	620.4 $\pm$ 19.5	12.6
ROME	Llama-3-8B	48.5 $\pm$ 50.0	48.1 $\pm$ 46.1	49.2 $\pm$ 43.1	354.5 $\pm$ 17.2	48.6
MEMIT		94.5 $\pm$ 22.7	89.3 $\pm$ 26.6	67.4 $\pm$ 31.5	593.4 $\pm$ 30.1	82.0
PRUNE		82.4 $\pm$ 38.1	65.3 $\pm$ 40.1	55.3 $\pm$ 36.2	577.9 $\pm$ 28.2	65.9
RECT		94.5 $\pm$ 22.8	84.4 $\pm$ 30.3	72.8 $\pm$ 28.2	601.8 $\pm$ 28.4	83.0
NSE		86.7 $\pm$ 34.0	56.7 $\pm$ 42.3	88.5 $\pm$ 18.3	591.4 $\pm$ 36.1	74.1
AlphaEdit		<u>98.8 $\pm$ 10.7</u>	<u>91.4 $\pm$ 24.0</u>	75.1 $\pm$ 26.3	<u>606.5 $\pm$ 22.7</u>	<u>87.3</u>
FT (old)		92.7 $\pm$ 26.1	83.5 $\pm$ 33.3	43.0 $\pm$ 37.1	408.3 $\pm$ 82.4	65.2
FT (updated)		92.5 $\pm$ 26.4	87.0 $\pm$ 29.3	49.5 $\pm$ 36.8	268.5 $\pm$ 56.6	70.6
PFT w/o smooth		87.5 $\pm$ 33.1	79.2 $\pm$ 35.3	49.0 $\pm$ 33.5	296.3 $\pm$ 119.7	67.4
PFT w/o s.VJPs		91.0 $\pm$ 28.6	71.7 $\pm$ 37.7	71.2 $\pm$ 25.9	599.0 $\pm$ 24.3	77.0
PFT		98.9 $\pm$ 10.4	96.4 $\pm$ 16.3	<u>87.3 $\pm$ 19.6</u>	619.0 $\pm$ 17.3	93.9

Table 7: Sequential knowledge editing using CounterFact: editing 2,000 relations across 20 iterations, 100 relations at a time.

with 10,000 edits or 3,000 sequential ones. These results demonstrate that PFT generalizes effectively across different editing scales, maintaining strong performance from small to large numbers of edited relations.

Method	LLM	ZSRE (accuracy)			
		Eff.	Gen.	Spe.	Score
Pre-Edited		26.4 $\pm$ 28.3	25.8 $\pm$ 27.9	27.0 $\pm$ 25.7	26.4
ROME	GPT-J	19.3 $\pm$ 32.2	17.8 $\pm$ 31.3	0.9 $\pm$ 4.1	2.4
MEMIT		95.4 $\pm$ 17.1	89.4 $\pm$ 25.3	26.4 $\pm$ 25.0	50.4
PRUNE		30.4 $\pm$ 35.0	29.3 $\pm$ 34.6	15.8 $\pm$ 20.2	23.0
RECT		71.5 $\pm$ 35.4	63.1 $\pm$ 37.2	26.6 $\pm$ 25.3	44.4
NSE		44.8 $\pm$ 31.9	43.3 $\pm$ 31.8	25.8 $\pm$ 25.3	35.7
AlphaEdit		87.0 $\pm$ 25.3	76.5 $\pm$ 33.4	22.3 $\pm$ 23.5	43.2
FT (old)		0.0	0.0	0.0 $\pm$ 1.1	0.0
FT (updated)		99.8 $\pm$ 3.1	98.2 $\pm$ 11.2	3.9 $\pm$ 11.4	10.8
PFT w/o smooth		0.1 $\pm$ 1.3	0.1 $\pm$ 1.4	0.0 $\pm$ 1.3	0.1
PFT w/o s.VJPs		99.9 $\pm$ 1.7	98.5 $\pm$ 9.7	25.9 $\pm$ 26.3	51.0
PFT		99.8 $\pm$ 3.2	97.5 $\pm$ 13.1	24.5 $\pm$ 24.6	49.1
Pre-Edited		22.2 $\pm$ 26.3	21.3 $\pm$ 25.9	24.2 $\pm$ 24.4	22.5
ROME	GPT2-XL	19.5 $\pm$ 34.3	17.4 $\pm$ 32.8	1.3 $\pm$ 5.1	3.5
MEMIT		63.9 $\pm$ 38.4	59.3 $\pm$ 39.1	26.0 $\pm$ 25.8	42.3
PRUNE		30.8 $\pm$ 35.8	29.1 $\pm$ 35.0	16.0 $\pm$ 20.8	23.2
RECT		49.3 $\pm$ 38.6	45.4 $\pm$ 37.9	25.7 $\pm$ 25.2	37.0
NSE		34.9 $\pm$ 29.9	33.1 $\pm$ 29.9	22.5 $\pm$ 23.7	29.0
AlphaEdit		57.7 $\pm$ 38.6	52.7 $\pm$ 39.0	20.8 $\pm$ 23.4	35.5
FT (old)		57.8 $\pm$ 37.3	54.5 $\pm$ 38.2	6.3 $\pm$ 13.8	15.4
FT (updated)		91.9 $\pm$ 21.8	90.0 $\pm$ 23.9	17.1 $\pm$ 21.6	37.4
PFT w/o smooth		15.1 $\pm$ 28.1	13.8 $\pm$ 27.1	0.3 $\pm$ 2.3	0.8
PFT w/o s.VJPs		99.9 $\pm$ 1.8	92.8 $\pm$ 21.0	26.8 $\pm$ 27.1	51.7
PFT		99.7 $\pm$ 4.4	90.4 $\pm$ 24.7	24.2 $\pm$ 24.6	48.1
Pre-Edited		35.7 $\pm$ 29.8	34.8 $\pm$ 29.5	31.8 $\pm$ 22.5	34.0
ROME	Llama-3-8B	0.0	0.0 $\pm$ 0.6	23.4 $\pm$ 13.1	0.0
MEMIT		86.9 $\pm$ 24.4	82.8 $\pm$ 27.6	32.5 $\pm$ 22.3	55.2
PRUNE		21.4 $\pm$ 29.2	18.9 $\pm$ 27.4	18.1 $\pm$ 19.0	19.3
RECT		71.7 $\pm$ 32.6	66.6 $\pm$ 33.9	32.3 $\pm$ 22.3	50.0
NSE		45.9 $\pm$ 30.2	45.7 $\pm$ 30.5	31.5 $\pm$ 22.4	39.8
AlphaEdit		84.8 $\pm$ 26.1	80.8 $\pm$ 28.9	32.5 $\pm$ 22.2	54.6
FT (old)		63.9 $\pm$ 33.6	58.9 $\pm$ 34.7	45.0 $\pm$ 16.5	54.7
FT (updated)		99.8 $\pm$ 3.1	97.8 $\pm$ 12.1	12.3 $\pm$ 15.7	29.6
PFT w/o smooth		93.7 $\pm$ 19.1	87.4 $\pm$ 26.4	4.0 $\pm$ 8.7	10.9
PFT w/o s.VJPs		96.3 $\pm$ 9.7	92.6 $\pm$ 16.9	34.2 $\pm$ 22.1	59.5
PFT		91.2 $\pm$ 20.8	84.1 $\pm$ 28.2	31.8 $\pm$ 22.4	55.2

Table 8: Knowledge editing of 10,000 ZSRE relations.

Method	LLM	ZSRE (accuracy)			
		Eff.	Gen.	Spe.	Score
Pre-Edited		27.8 $\pm$ 29.1	27.1 $\pm$ 28.6	27.5 $\pm$ 26.0	27.5
ROME	GPT-J	65.8 $\pm$ 40.2	64.1 $\pm$ 39.8	16.6 $\pm$ 21.0	32.9
MEMIT		97.4 $\pm$ 11.6	94.5 $\pm$ 17.7	29.3 $\pm$ 26.9	54.6
PRUNE		53.7 $\pm$ 38.3	50.5 $\pm$ 38.1	24.0 $\pm$ 25.0	37.5
RECT		96.0 $\pm$ 14.6	91.6 $\pm$ 21.6	<u>28.5 $\pm$ 26.2</u>	53.2
NSE		45.3 $\pm$ 32.5	44.2 $\pm$ 32.4	27.1 $\pm$ 25.9	36.8
AlphaEdit		99.3 $\pm$ 6.3	96.2 $\pm$ 15.4	27.9 $\pm$ 26.1	53.3
FT (old)		0.0	0.0	0.0 $\pm$ 0.8	0.0
FT (updated)		9.8 $\pm$ 25.0	8.6 $\pm$ 22.4	0.4 $\pm$ 3.3	1.0
PFT w/o smooth		0.4 $\pm$ 5.3	0.4 $\pm$ 5.0	0.0 $\pm$ 0.8	0.1
PFT w/o s.VJPs		96.2 $\pm$ 14.6	93.8 $\pm$ 18.4	24.6 $\pm$ 25.1	48.6
PFT		<u>97.9 $\pm$ 10.3</u>	<u>95.2 $\pm$ 17.1</u>	28.4 $\pm$ 27.0	<u>53.6</u>
Pre-Edited		23.7 $\pm$ 27.3	22.8 $\pm$ 26.9	25.0 $\pm$ 24.4	23.8
ROME	GPT2-XL	46.9 $\pm$ 43.0	42.4 $\pm$ 41.9	14.5 $\pm$ 19.4	26.3
MEMIT		85.3 $\pm$ 27.5	79.8 $\pm$ 31.5	27.1 $\pm$ 26.6	<u>49.0</u>
PRUNE		36.5 $\pm$ 36.4	34.1 $\pm$ 35.5	20.6 $\pm$ 23.4	28.5
RECT		83.1 $\pm$ 29.2	75.4 $\pm$ 34.0	24.5 $\pm$ 25.1	45.4
NSE		39.3 $\pm$ 31.3	37.8 $\pm$ 31.3	24.9 $\pm$ 24.9	32.6
AlphaEdit		92.4 $\pm$ 19.4	85.9 $\pm$ 26.9	25.6 $\pm$ 25.9	48.7
FT (old)		30.6 $\pm$ 41.0	28.4 $\pm$ 40.4	1.7 $\pm$ 6.4	4.5
FT (updated)		10.9 $\pm$ 27.0	11.4 $\pm$ 27.7	2.7 $\pm$ 8.8	5.4
PFT w/o smooth		4.7 $\pm$ 19.2	4.7 $\pm$ 18.9	0.6 $\pm$ 3.8	1.4
PFT w/o s.VJPs		<u>93.8 $\pm$ 18.5</u>	87.3 $\pm$ 27.2	<u>25.9 $\pm$ 26.6</u>	49.5
PFT		94.2 $\pm$ 18.1	<u>87.2 $\pm$ 27.3</u>	25.2 $\pm$ 25.0	48.6
Pre-Edited		37.0 $\pm$ 30.2	36.4 $\pm$ 30.0	31.8 $\pm$ 22.6	34.9
ROME	Llama-3-8B	0.0	0.0	19.9 $\pm$ 16.3	0.0
MEMIT		89.6 $\pm$ 20.6	85.2 $\pm$ 24.6	32.3 $\pm$ 22.3	55.7
PRUNE		26.8 $\pm$ 31.4	24.2 $\pm$ 29.8	17.4 $\pm$ 18.9	22.1
RECT		87.6 $\pm$ 22.4	82.8 $\pm$ 26.6	32.3 $\pm$ 22.4	55.1
NSE		47.0 $\pm$ 30.8	47.3 $\pm$ 30.9	32.3 $\pm$ 22.6	40.9
AlphaEdit		93.4 $\pm$ 15.8	89.1 $\pm$ 21.7	32.5 $\pm$ 22.3	56.9
FT (old)		41.5 $\pm$ 32.3	38.7 $\pm$ 31.6	41.4 $\pm$ 15.5	40.5
FT (updated)		67.9 $\pm$ 35.0	65.8 $\pm$ 35.1	7.6 $\pm$ 12.7	18.6
PFT w/o smooth		21.4 $\pm$ 33.9	20.1 $\pm$ 32.8	1.7 $\pm$ 5.6	4.5
PFT w/o s.VJPs		98.7 $\pm$ 7.6	95.2 $\pm$ 15.9	<u>34.1 $\pm$ 22.0</u>	60.0
PFT		<u>96.3 $\pm$ 10.3</u>	<u>93.3 $\pm$ 17.1</u>	32.0 $\pm$ 22.6	<u>57.3</u>

Table 9: Sequential knowledge editing using ZSRE: editing 2,000 relations across 20 iterations, 100 relations at a time.

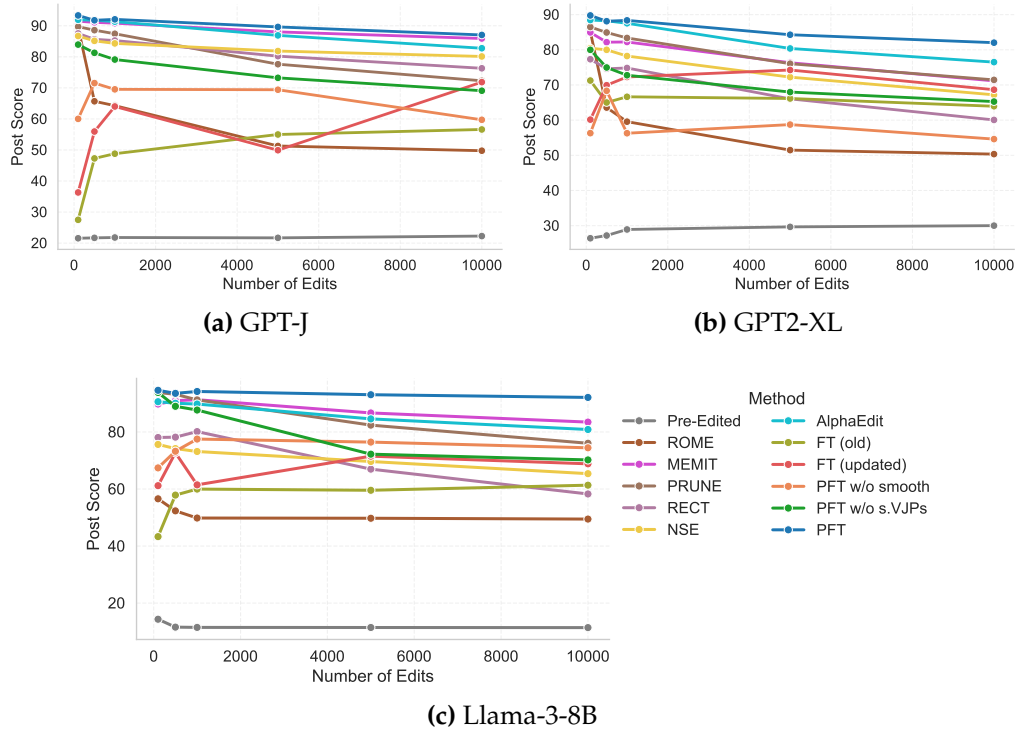


Figure 6: CounterFact Editing Score after one-shot editing versus number of edited relations.

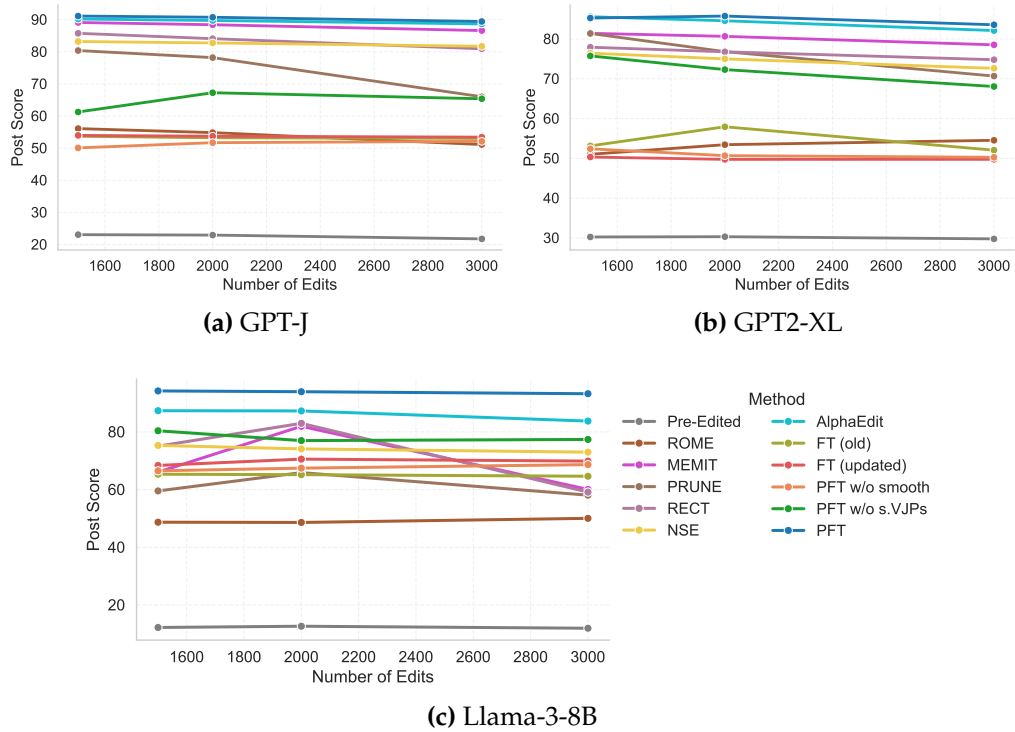


Figure 7: CounterFact Editing Score after sequential editing versus total number of edits.

C.3 Evaluation on Additional Model Families

The main experiments follow prior knowledge-editing evaluations and therefore use GPT2-XL, GPT-J, and Llama-3-8B, for which results and implementation from earlier editing methods are available (Meng et al., 2022; Fang et al., 2025). In order to examine PFT generalization to newer model families and a wider range of parameter scales, we additionally evaluate Llama-3.2-1B, Llama-3.2-3B, and the Qwen2.5 family at 1.5B, 3B, 7B, and 14B parameters, all used with their instruction-following versions. We apply the same evaluation settings used in our main experiments in section 6: one-shot editing of 10,000 relations and sequential editing of 2,000 relations. Because corresponding results for all prior editing methods are not available for these models, we only compare PFT with each model’s pre-edit baseline. The results are reported in Table 10.

Model	Method	10,000 one-shot edits		2,000 sequential edits	
		CounterFact	ZSRE	CounterFact	ZSRE
Llama-3.2-1B	Pre-Edited	23.8	28.5	25.6	30.0
	PFT	88.6	48.9	90.3	50.7
Llama-3.2-3B	Pre-Edited	49.5	31.2	49.1	32.3
	PFT	90.0	52.4	92.0	53.5
Qwen2.5-1.5B	Pre-Edited	24.9	32.9	25.5	34.7
	PFT	86.6	58.9	88.7	26.6
Qwen2.5-3B	Pre-Edited	20.7	34.2	21.4	35.6
	PFT	86.8	59.8	90.4	62.7
Qwen2.5-7B	Pre-Edited	18.8	35.5	20.7	36.6
	PFT	88.3	64.8	91.6	64.8
Qwen2.5-14B	Pre-Edited	14.3	37.9	15.8	38.9
	PFT	88.4	66.8	91.7	70.4

Table 10: Editing scores for PFT on additional model families.

Across the additional models, PFT obtains CounterFact editing scores between 86.6 and 90.0 in the one-shot setting and between 88.7 and 92.0 in the sequential setting. On ZSRE, the larger Qwen2.5 models show particularly strong results, reaching 66.8 for one-shot editing and 70.4 for sequential editing with Qwen2.5-14B. The exception is sequential editing with Qwen2.5-1.5B, where the ZSRE score decreases from 34.7 to 26.6 despite the strong CounterFact result. Overall, the results demonstrate that PFT can be applied across model families and scales.

C.4 Efficiency Comparison

In this section, we analyze the computational efficiency of PFT.

Hardware. PFT is designed to operate on a single GPU, in contrast to retraining-based approaches that often require multiple GPUs. In our experiments, we successfully ran PFT on models such as Llama-3-8B (BF16) and GPT-J (FP16) using a single NVIDIA A6000 GPU with 48GB of memory.

External Data for Covariance Matrix Collection. Similar to MEMIT (Meng et al., 2023), PFT requires a covariance matrix per edited layer to capture activation statistics for smoothing. This matrix is computed once by sampling activations from model inference and caching the results. In our implementation, we collect activations from 100,000 Wikipedia samples, following the setup of MEMIT.

Importantly, PFT does not require external data during the editing process itself, unlike continual FT methods that rely on additional training data. Furthermore, PFT does not use KL-divergence regularization, which is commonly employed in MEMIT and related approaches, as well as in standard continual FT.

Batch Utilization. MEMIT optimizes each relation separately, typically using one base prompt, 5 augmentations, and one additional KL-divergence prompt, which can be batched

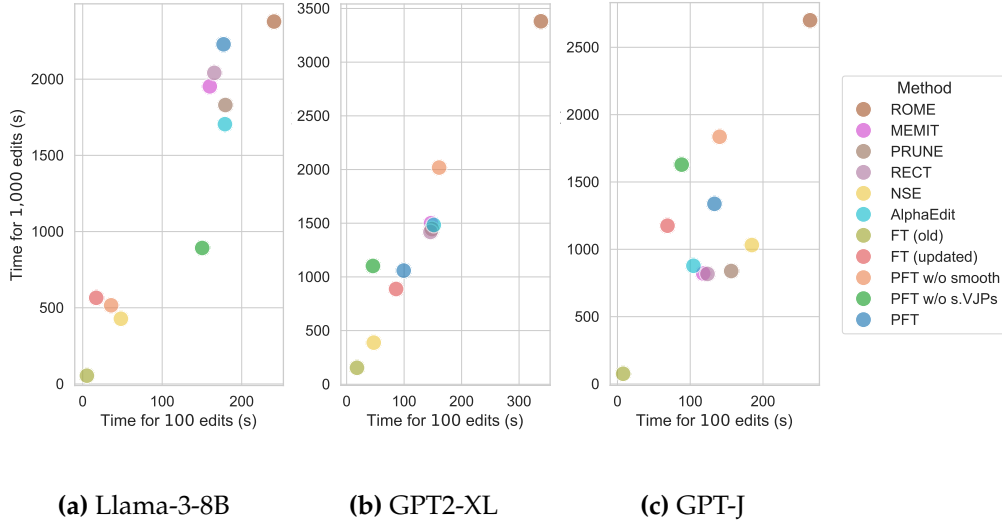


Figure 8: Running time of different editing methods for 1,000 and 100 relations.

only per relation. In contrast, PFT operates directly within the FT framework and allows batching across multiple relations. Specifically, PFT processes base prompts and augmentations jointly across different relations, enabling more efficient utilization of GPU resources.

Execution Time. We measure the execution time of PFT and compare it to other editing methods. For fairness, we exclude the one-time cost of covariance matrix collection, which is shared across MEMIT-style methods (Meng et al., 2023). Additionally, we report end-to-end execution times for methods such as MEMIT without assuming cached intermediate results across runs.

We evaluate two settings: (i) editing 1,000 relations in a one-shot setup, and (ii) editing 100 relations in a setup analogous to sequential editing. All Llama-3 experiments were conducted on NVIDIA A100 GPUs, while GPT-J and GPT2-XL experiments were conducted on NVIDIA A6000 GPUs.

The results, presented in Figure 8, show that most baseline methods based on MEMIT exhibit similar execution times, while NSE (Jiang et al., 2025b) and FT-based approaches are generally faster. The runtime of PFT varies across models: for Llama-3 and GPT-J, it achieves comparable execution time for 100 edits, with only a slight increase in runtime, while for GPT2-XL it is slightly faster.

Overall, PFT demonstrates competitive efficiency across all models and settings, with differences in execution time that are small and consistent with existing methods.

Summary. PFT provides a favorable trade-off between editing performance and computational efficiency, achieving improved editing results while maintaining competitive execution time.

C.5 General Language Modeling Evaluation

Effective knowledge editing should preserve a model’s general capabilities, including performance on standard language understanding tasks. Prior work (Meng et al., 2022; 2023) evaluates this aspect using generation entropy as a proxy for fluency, which we also report in section 6.

To further assess the impact of editing on general capabilities, we follow Fang et al. (2025)’s evaluation of Llama-3 which includes the GLUE benchmark (Wang et al., 2018). Specifically, we measure F1 scores across six tasks, using 100 samples per task: SST (Socher et al., 2013), MRPC (Dolan & Brockett, 2005), CoLA (Warstadt et al., 2018), RTE (Bentivogli et al., 2009),

MMLU (Hendrycks et al., 2021), and NLI (Williams et al., 2018). See Wang et al. (2018) for further details regarding each task.

We report results for both one-shot editing with up to 10,000 edits (Figure 9) and sequential editing (Figure 10) with up to 3,000 edits. In these comparisons, we include baselines specifically designed to mitigate degradation in general model capabilities, including PRUNE (Ma et al., 2025), NSE (Jiang et al., 2025b), and AlphaEdit (Fang et al., 2025). We also include MEMIT (Meng et al., 2023) as a representative baseline, along with our implementation of FT.

In the one-shot setting, all methods maintain relatively stable performance up to 10,000 edits, with only minor degradation observed with PRUNE on SST and AlphaEdit on MMLU. In contrast, in the sequential setting, we observe that MEMIT and PRUNE exhibit gradual performance degradation across most tasks as the number of edits increases from 1,500 to 3,000. AlphaEdit and NSE largely preserve performance with no change in F1 scores across the tested setups. Across all tasks, PFT demonstrates consistently stable performance, matching or outperforming NSE and AlphaEdit.

For completeness, we also repeat these experiments on GPT2-XL (Figure 11, Figure 12) and GPT-J (Figure 13, Figure 14). However, these models are relatively older and less capable than Llama-3, and their pre-edit performance is comparatively lower. As a result, their evaluation is noisier, particularly on tasks such as SST and MMLU. Nevertheless, on certain tasks, such as NLI, we observe similar trends to those seen with Llama-3, where PFT and strong baselines such as NSE maintain stable F1 scores even with up to 3,000 edits in the sequential setup.

Overall, these results highlight the robustness of PFT under iterative editing, demonstrating that it preserves general language understanding capabilities at a level comparable to the pre-edited model.

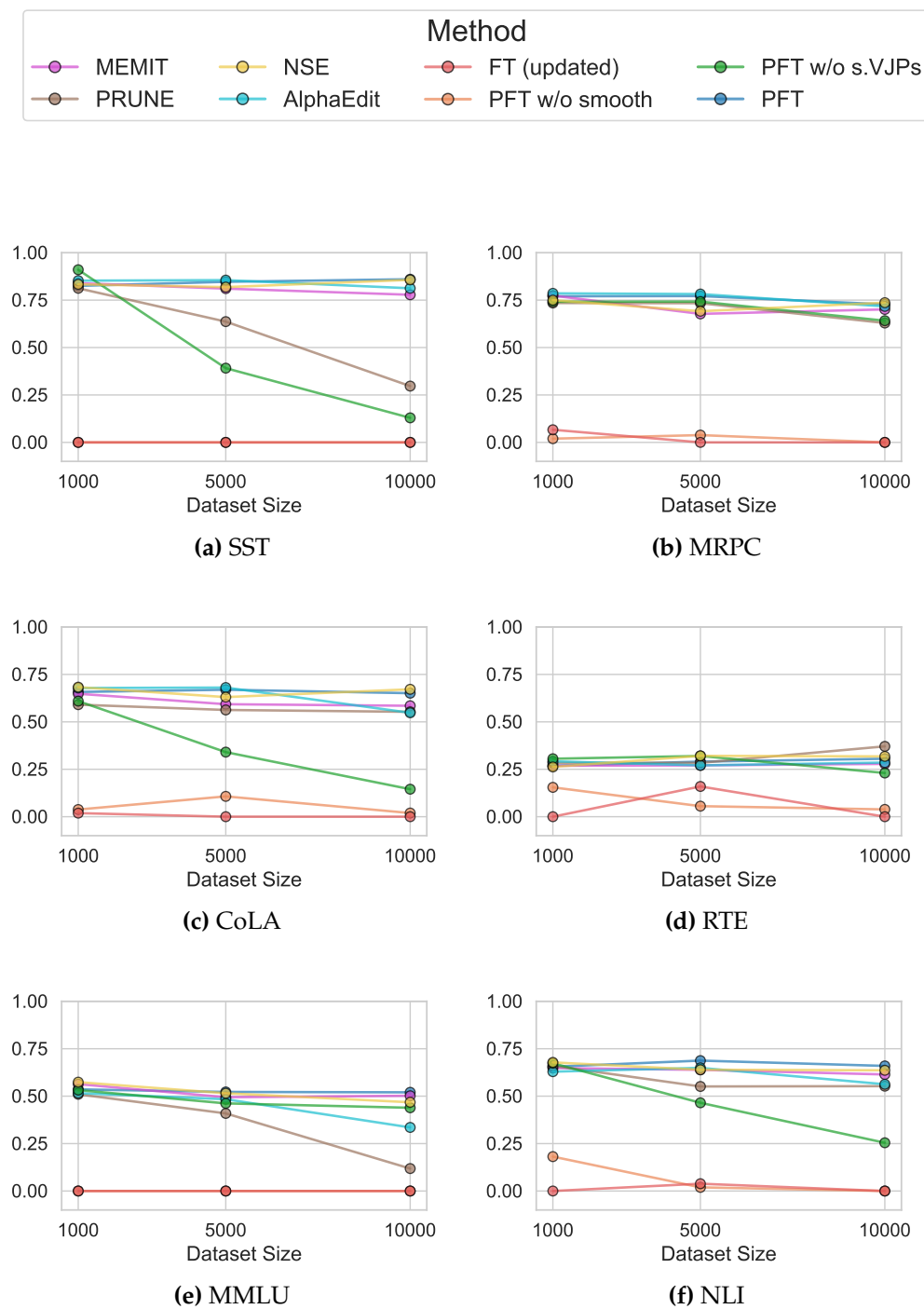


Figure 9: GLUE results for one-shot editing with Llama-3-8B.

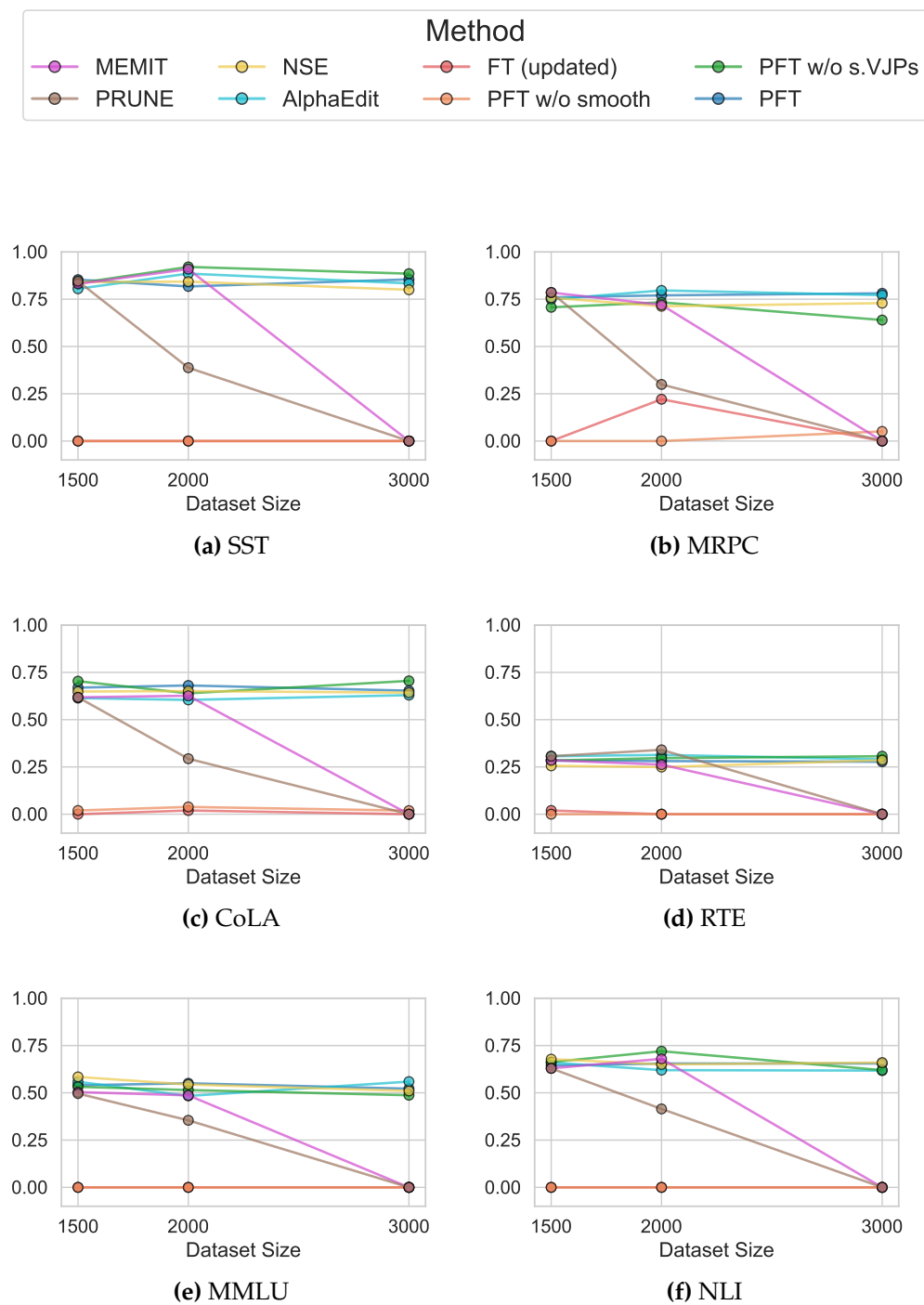


Figure 10: GLUE results for sequential editing with Llama-3-8B.

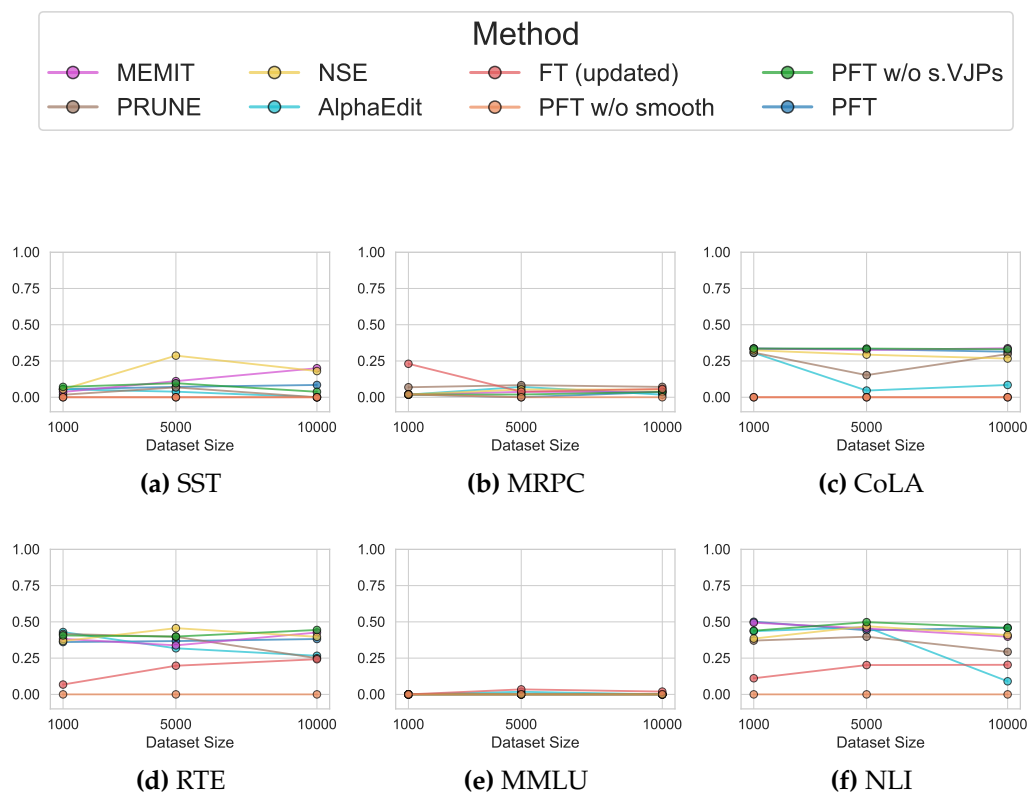


Figure 11: GLUE results for one-shot editing with GPT2-XL.

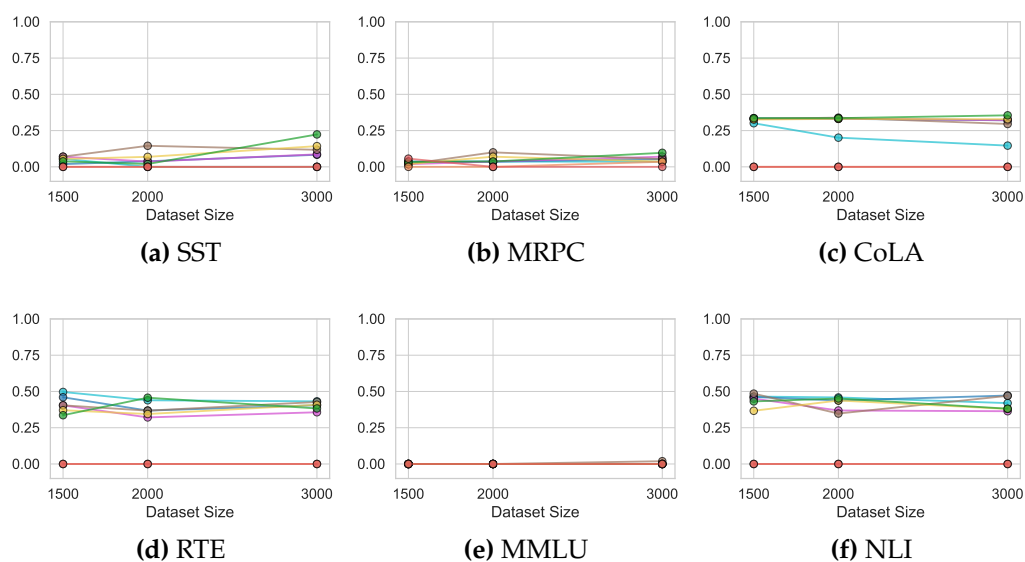


Figure 12: GLUE results for sequential editing with GPT2-XL.

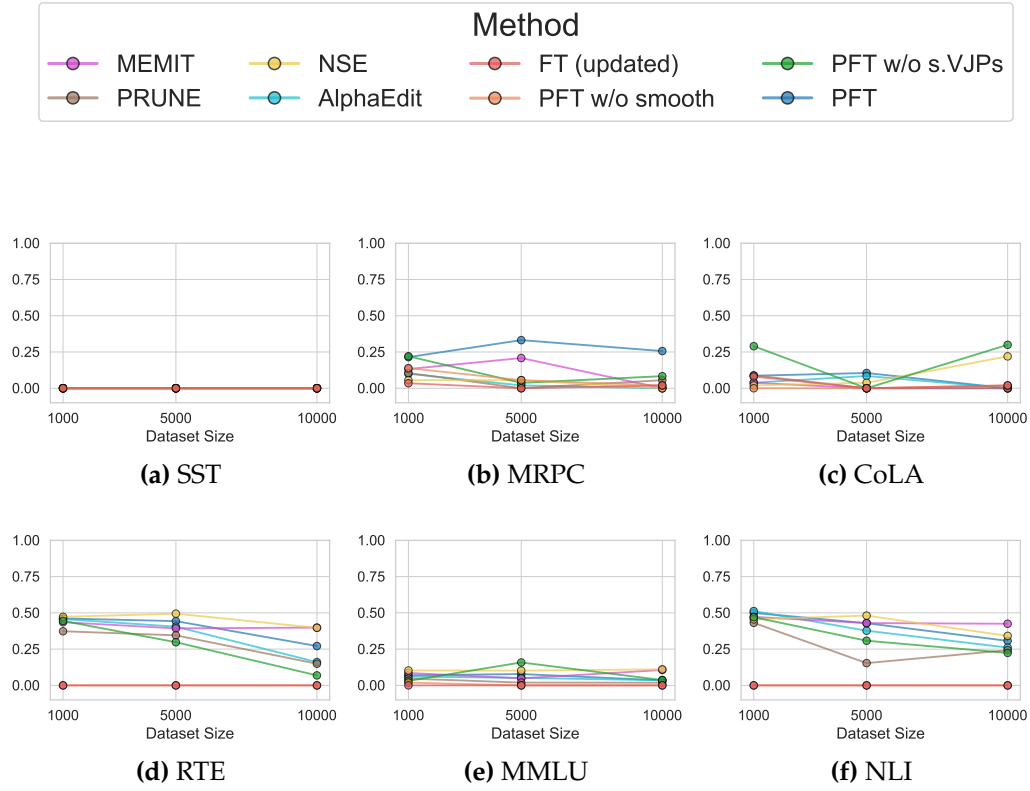


Figure 13: GLUE results for one-shot editing with GPT-J.

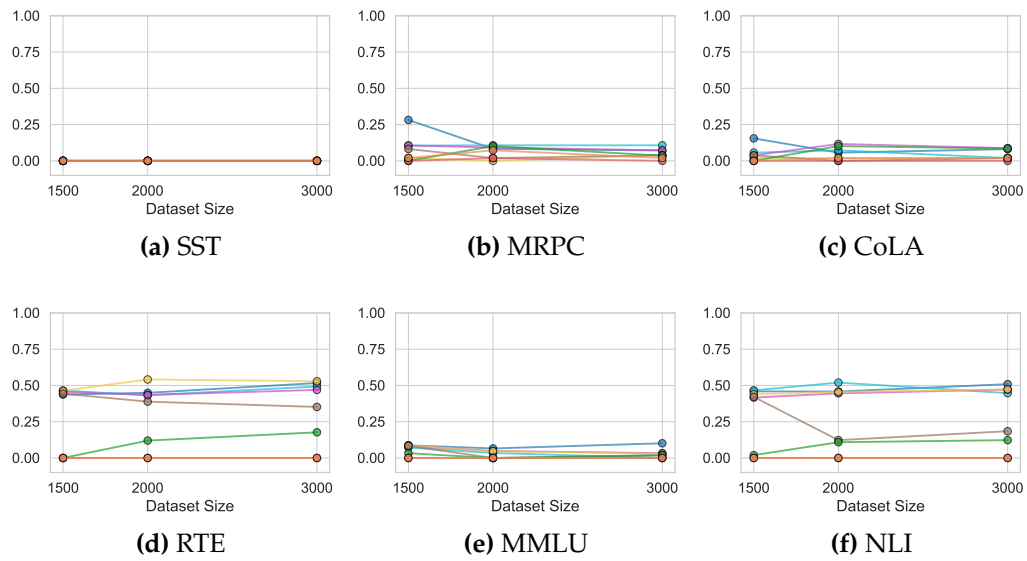


Figure 14: GLUE results for sequential editing with GPT-J.

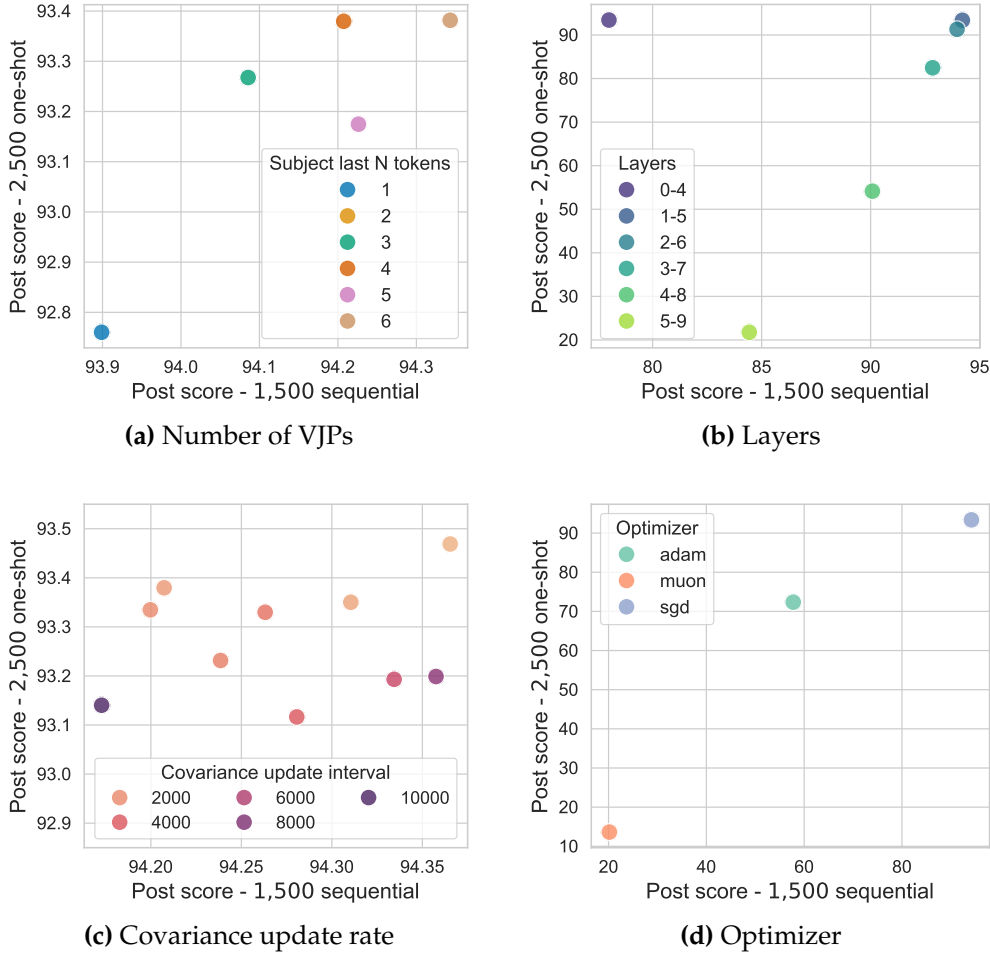


Figure 15: Knowledge editing performance of Llama-3 under different hyperparameter configurations.

C.6 Ablation Study

In this section, we analyze the effect of key hyperparameters in PFT. The goals of this study are: (i) to justify the chosen hyperparameter settings, (ii) to demonstrate the robustness and flexibility of PFT across different configurations, and (iii) to provide guidance for future work adopting PFT.

Throughout this section, we evaluate different configurations using Llama-3 (8B, Instruct) (Grattafiori et al., 2024) on CounterFact (Meng et al., 2022). We follow a setup similar to section 6, using one-shot editing with 2,500 relations and sequential editing with 1,500 relations. Unless stated otherwise, we refer to the base PFT configuration as the one described in Appendix B.

Number of VJPs. As described in section 5, PFT constructs gradient updates using only a subset of VJPs corresponding to suffix tokens of the subject. The number of selected tokens determines how many VJPs contribute to each update step. In section 6, we showed that selective VJPs improve generalization and fluency compared to using all token-level gradients. Here, we further analyze the impact of varying this number.

Figure 15(a) reports post-editing scores across different choices, ranging from a single VJP (only the last subject token) to all subject-token VJPs. All configurations achieve strong performance, with post-editing scores above 92.8 in the one-shot setting. Increasing the number of selected tokens further improves performance, reaching up to 93.4. At the same

time, using all subject-token VJPs does not significantly degrade performance, suggesting that the relevant signal is distributed across multiple tokens.

In practice, selecting a moderate number of VJPs provides a good trade-off between stability and computational efficiency, due to reduced matrix operations. Based on these observations, we adopt 4 VJPs in our main configuration, capturing most of the subject span while maintaining strong and stable performance.

Layer Selection. Motivated by prior work that edits only the second MLP projection matrix W_2 (Appendix A), PFT also restricts updates to this component. Methods such as MEMIT (Meng et al., 2023) and AlphaEdit (Fang et al., 2025) select 5–6 layers based on causal mediation analysis (Vig & Belinkov, 2019) of the last subject token. For Llama-3, MEMIT and AlphaEdit use layers 4–8. Since PFT utilizes both the last subject token and earlier tokens, we hypothesize that it may benefit from shifting the edited layers toward earlier positions. To test this, we perform a grid search over layer configurations for Llama-3. Results in Figure 15(b) show that selecting layers 1–5 yields the best performance, while shifting toward later layers (e.g., 5–9) leads to gradual degradation.

Interestingly, selecting earlier layers (0–4) achieves comparable performance in the one-shot setting, but results in lower performance in the sequential setting. This may stem from suboptimal hyperparameter tuning or from the increased sensitivity of earlier layers to accumulated edits. Overall, these findings highlight the importance of layer selection and suggest that mid-layer representations provide a favorable balance between expressivity and stability.

Covariance updates. As described in section 5, we update the covariance matrix $K_0 K_0^\top$ periodically according to $K_0 K_0^\top \leftarrow K_0 K_0^\top + X X^\top$, using an interval of 1,500 training steps. This choice is supported by the grid search in Figure 15(c), which evaluates update intervals from 500 to 10,000 steps. Overall, the update frequency has only a minor effect on performance, with variation limited to roughly 0.2 points in the sequential setting and 0.6 points in the one-shot setting. Still, performance degrades slightly at the longest interval of 10,000 steps, suggesting that more frequent covariance updates improve stability during longer optimization runs.

Optimizer. In section 5, we implement PFT with vanilla SGD rather than an adaptive optimizer. We justify this choice in Figure 15(d), where we compare SGD to Adam (Kingma, 2014) and MUON (Liu et al., 2025). Replacing SGD with adaptive optimizers leads to clear degradation, reducing sequential editing performance to around 60 with Adam and around 20 with MUON, although all runs share the same hyperparameters apart from the optimizer, and these settings may not be optimal for Adam or MUON. In knowledge editing, adaptive optimizers accumulate momentum and gradient statistics across examples, which may unintentionally mix distinct edited associations. Such mixing is likely misaligned with PFT’s goal of producing precise, association-specific updates.