REVISITING THE DYNAMICS OF TARGET OPTIMIZATION IN LARGE LANGUAGE MODEL EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge editing has emerged as a promising approach for efficiently updating knowledge in large language models (LLMs). These methods first optimize a target hidden state that steers the model toward a new output, and then edit the model parameters to map the original hidden state to this target. While most existing methods focus on the second stage of parameter editing to preserve other knowledge, we find that the geometry of upstream target hidden-state optimization is also critical to editing locality. For LLMs equipped with normalization layers such as LayerNorm or RMSNorm, we find that changes in the direction of the target hidden state are essential for injecting new knowledge, whereas the magnitude contributes little. However, we also find that the norm of the target hidden state continuously accumulates during gradient descent optimization. We further show that this unnecessary norm growth increases interference with non-target knowledge, thereby degrading editing locality. Building on this insight, we propose Spherical Editing (SE), a simple yet effective method that constrains the target hidden state to remain on the sphere with the same radius as the original state throughout optimization. By eliminating unnecessary radial growth while retaining essential directional changes, SE reduces interference with other knowledge without sacrificing editing success. Extensive experiments on three LLMs, two benchmarks, and 10,000 edits demonstrate that SE consistently improves editing locality while maintaining editing effectiveness. Code is available at `https://anonymous.4open.science/r/se-review-artifact-5869/`.

1 INTRODUCTION

Large language models (LLMs) acquire extensive factual knowledge during pre-training, yet their knowledge can become outdated, incomplete, or incorrect. Model editing aims to update specific model behaviors without expensive retraining while preserving unrelated knowledge and general capabilities (Meng et al., 2023). Among existing approaches, locate-then-edit methods have emerged as a particularly influential paradigm. These methods identify a small set of knowledge-critical feed-forward layers and formulate their output projections as associative memories: an input key k associated with an edit request should be mapped from its original hidden state h to a target hidden state m that elicits the desired answer. Once m is obtained through gradient-based activation optimization, the corresponding parameter update can be computed efficiently, often in closed form (Meng et al., 2022; 2023). Figure 1 provides an intuitive illustration of this two-stage locate-then-edit process, with further details introduced in §3.

Despite its central role in this pipeline, the geometry of the target hidden state remains largely overlooked. Existing methods typically treat m as an unconstrained vector and optimize all of its dimensions directly using the likelihood of the target answer. This optimization typically simultaneously results in two geometrically distinct discrepancies between h and m: *direction* and *magnitude*. These two components are implicitly assumed to contribute jointly to successful knowledge injection. However, this assumption has not been systematically examined.

In this paper, we try to systematically study these two components and find that magnitude contributes little to the new knowledge, which is supported by both theoretical analysis and empirical verification. Our insights include: 1. In modern LLMs, the gradient of the prediction loss *w.r.t.* m is always orthogonal to m due to the effect of LayerNorm or RMSNorm. That is, the gradient lies

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

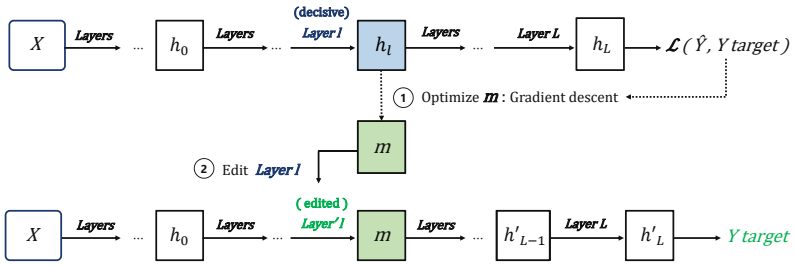


Figure 1: Overview of locate-then-edit model editing. ① In the upstream target construction stage, the target hidden state m of the decisive token at the decisive layer l is obtained by gradient descent on the loss of the target answer. ② In the downstream parameter updating stage, layer l is edited so that the original hidden state h_l is mapped to m, allowing the subsequent layers to produce the target answer.

in the tangent space of the hypersphere at m, meaning that the reduction in loss is mostly driven by the rotation of m, rather than by a change in its magnitude. 2. Existing iterative gradient-descent-based optimization almost invariably increases the magnitude of m. 3. When we rescale m to retain its direction but have the same norm as the original h, steering success remains nearly unchanged. However, magnitude does affect locality (i.e., the preservation of non-target knowledge) a lot.

Motivated by these findings, we introduce *Spherical Editing* (SE). Rather than optimizing m in unconstrained Euclidean space, SE differentiably reparameterizes the target on the sphere of radius $||h||$ in every forward pass, so that the edit is realized through rotation without radial stretching. By construction, $||m|| = ||h||$ throughout optimization, while the direction of m is optimized through the normalization. Experiments across three LLMs and two commonly used benchmarks show the effectiveness of our method.

2 Related Work

Locate-then-edit paradigm. Recent work on parameter editing often follows a *locate-then-edit* (LTE) paradigm: selecting tokens and layers associated with a fact, constructing target hidden states, and updating selected parameters to incorporate the new knowledge. ROME (Meng et al., 2022) uses causal tracing to identify sites for editing, and MEMIT (Meng et al., 2023) extends this approach to updates across multiple layers and facts. Later methods, including RECT (Gu et al., 2024), EMMET (Gupta et al., 2024), PMET (Li et al., 2024), PRUNE (Ma et al., 2025), AdaEdit (Li & Chu, 2025), AlphaEdit (Fang et al., 2025), LTI (Zhang et al., 2025), AnyEdit (Jiang et al., 2025a), BLUE (Li et al., 2025a), and NSE (Jiang et al., 2025b), develop this framework through changes to localization, optimization, regularization, or sequential editing. These methods study how selected parameter updates can incorporate new facts while preserving other knowledge. Recent work also considers more complex settings, such as propagating the ripple effects of an edit to logically related facts (Dong et al., 2025).

Target construction of LTE. The LTE paradigm consists of two stages: (1) constructing the target hidden state, which plays a role analogous to a label in supervised learning, and (2) updating the parameters to reach this target. Most prior work focuses on the second stage, namely how to solve or regularize the parameter update, while the construction of the target itself is much less explored. MEMIT (Meng et al., 2023) keeps the target close to the original hidden state with an ℓ_2 term, FE (Liu et al., 2026a) obtains the targets of multiple layers by forward propagation from the first edited layer, and EAMET (Dai et al., 2026) aligns the distribution of residuals with that of keys. All of them treat the target as a free vector. We instead separate its direction from its magnitude, and show that constraining the magnitude alone already improves locality.

Other methods for knowledge editing. A parallel line of work supplies updated information through retrieved examples or instructions at inference time (Zheng et al., 2023; Jiang et al., 2024). These methods use context to express new facts, although they may first train the model to use this in-

formation. Other approaches learn an editor that predicts parameter updates: MEND (Mitchell et al., 2022a), KE (De Cao et al., 2021), InstructEdit (Zhang et al., 2024), and RLEdit (Li et al., 2025b) train auxiliary networks that can then process new edit requests. T-Patcher (Huang et al., 2023), SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023), WISE (Wang et al., 2024), and KDE (Xu et al., 2025) use additional parameters or memory components to support editing. Their designs differ in how new knowledge is stored and when it is used during inference.

3 PRELIMINARIES

Notation. Let f_θ be a pre-trained LLM with parameters θ, and let f_{θ^*} denote the model after editing. The *edit set* collects the factual associations to be installed,

$$S^* = \{(q_i^*, y_i^*)\}_{i=1}^n, \tag{1}$$

where q_i^* is a query that elicits a specific fact (e.g., *"The president of the US is"*), y_i^* is the desired output (e.g., *"Trump"*), and n is the number of edits. To keep the remaining knowledge intact, one additionally maintains a *preservation set*

$$S = \{(q_j, y_j)\}_{j=n+1}^{n+u}, \tag{2}$$

whose queries q_j are drawn from a background corpus such as Wikipedia. Crucially, the labels here are *not* the ground-truth continuations in the corpus, but the model's own predictions, $y_j = f_\theta(q_j)$: the aim is to freeze the pre-edit behavior of f_θ outside S^*, not to improve it. As shown below, this seemingly minor design choice is precisely what makes a closed-form update possible.

General editing objective. Knowledge editing seeks to install the target associations while perturbing the model's behavior on all other inputs as little as possible:

$$\theta^* = \arg\min_{\hat{\theta}} \sum_{i=1}^{n} \mathcal{L}_1\big(f_{\hat{\theta}}(q_i^*),\, y_i^*\big) \;+\; \lambda \sum_{j=n+1}^{n+u} \mathcal{L}_2\big(f_{\hat{\theta}}(q_j),\, y_j\big)\,, \tag{3}$$

where $\mathcal{L}_1$ enforces the target edits and $\mathcal{L}_2$ penalizes drift on non-target knowledge.

Locate-then-edit. Optimizing Eq. 3 directly using gradient descent is prohibitively expensive due to the huge value of u. The *locate-then-edit* (LTE) paradigm instead first identifies where a fact is stored and then applies a targeted update at that location. Specifically, it finds that if we steer the hidden state of a key decisive token (usually the last token of the subject) at a decisive layer (usually an MLP layer) to the target hidden state, we can get the target final output (Meng et al., 2023).

Let W denote the weight matrix selected for editing, k_i the hidden state of the decisive token at the layer immediately preceding W, and m_i the idealized output of W that induces the target y_i^*; the latter is obtained by treating m_i as a free parameter and minimizing by gradient descent the cross-entropy between the resulting model output and y_i^* (see Appendix A.1 for the details about how to get m_i). The edited weights are then $W^* = W + \Delta^*$ with

$$\Delta^* = \arg\min_{\Delta}\Big(\sum_{i=1}^{n} \big\|\Delta k_i - (m_i - h_i)\big\|^2 + \frac{\lambda}{u} \sum_{j=n+1}^{n+u} \big\|\Delta k_j\big\|^2 \Big), \tag{4}$$

where k is the input hidden state of W, $r_i = m_i - h_i$ is the residual that the update must supply for the i-th edit, and the factor $1/u$ turns the second sum into an empirical average over the preservation set.

Solving Eq. 4 by gradient descent is impractical, as u is typically enormous (on the order of 4×10^7 by default). The decisive advantage of LTE over fine-tuning is that Eq. 4 is a ridge-regularized least-squares problem and thus admits a closed-form, single-step solution:

$$\Delta^* = R\,K_I^\top \Big(K_I K_I^\top + \lambda C\Big)^{-1}, \qquad C = \frac{1}{u} K_J K_J^\top, \tag{5}$$

where capital letters stack the corresponding vectors column-wise ($K_I = [k_1, \ldots, k_n]$, $K_J = [k_{n+1}, \ldots, k_{n+u}]$, $M_I = [m_1, \ldots, m_n]$, $R = [r_1, r_2, \ldots, r_n]$) and R collects the residuals of all

edits. Note that the preservation set enters only through the second-moment matrix $K_J K_J^\top$, which can be estimated once from the background corpus and cached offline. This efficient solution has made LTE one of the dominant paradigms in the knowledge editing literature.

How to get m. The target m is obtained by gradient descent with the model parameters frozen. The hidden state of the decisive token at the edited layer is treated as an optimizable variable, initialized at h and forwarded through the remaining layers, and the loss is the cross-entropy of the target answer y^* under the resulting output. Consider the target hidden state as m and the subsequent LLM layers as f; then

$$m^* = \arg\min_m \mathcal{L}(f(m), y^*). \tag{6}$$

See Appendix A.1 for details.

4 Motivation and Method

4.1 The dynamics of target optimization

This part will demonstrate our observation that the norm operation (e.g., LayerNorm and RMSNorm) will greatly eliminate the radial gradient through normalized computational branches.

Let $\mathcal{L}(m)$ denote the prediction loss obtained by replacing h with m and forwarding it through the remaining model, and let

$$g(m) = \nabla_m \mathcal{L}(m). \tag{7}$$

We next characterize how normalization shapes this gradient.

Theorem 1 (Orthogonality under scale invariance) *Suppose that a differentiable function* $\mathcal{J} : \mathbb{R}^d \to \mathbb{R}$ *is locally invariant to positive rescaling at* $m \neq 0$*, i.e.,*

$$\mathcal{J}(\alpha m) = \mathcal{J}(m) \quad \textit{for all } \alpha \textit{ in a neighborhood of } 1. \tag{8}$$

Then

$$m^\top \nabla_m \mathcal{J}(m) = 0. \tag{9}$$

The proof is in Appendix A.6.

Remark. In this paper, we define m as the hidden state after the residual addition (see Appendix A.5 for details). The norm operation (e.g., LayerNorm and RMSNorm) in modern LLMs satisfies Eq. 8 well if the numerical constant ε can be omitted (see Appendix A.7). As a result, we can expect the near orthogonality between the gradient and m because the minor radial gradient can only arise through unnormalized residual side paths.

Empirical verification 1. To validate orthogonality in practice, we compare the radial and tangential components of the gradients during gradient descent on m. The results are shown in Figure 2. We see that the radial gradient is negligible compared to the tangential gradient. This means that the gradient primarily guides m to rotate to reduce the loss. The same holds beyond this single sample: over every gradient step of the first 300 zsRE edits of each model, the median ratio $\|g_{\text{rad}}\|/\|g_{\text{tan}}\|$ is 1.6%, 1.5% and 1.3% on GPT2-XL, Qwen2.5-7B-Instruct and GLM-4-9B-Chat, and the angle between $g(m)$ and m stays within 2.8° of 90° for 95% of the steps (Appendix A.2).

We denote the gradient at time step t as $g_t = \nabla_{m_t} \mathcal{L}(m_t)$, and the gradient descent update as $m_{t+1} = m_t - \eta g_t$. Then we have the following conclusion:

Theorem 2 (The magnitude of m **grows continuously)** *If the radial gradient is zero, then the iterative gradient-descent-based optimization invariably increases the magnitude of* m*:*

$$\nabla_{m_t} \mathcal{L}(m_t) \perp m_t \Longrightarrow ||m_{t+1}|| \geq ||m_t||, \quad \forall\, t, \eta > 0. \tag{10}$$

The proof is in Appendix A.8.

Remark. Although the gradient primarily guides m to rotate to reduce the loss, gradient descent also causes the magnitude of m to increase passively (Figure 3). As a result, the optimized m will consistently have a larger magnitude than the corresponding original hidden state h.

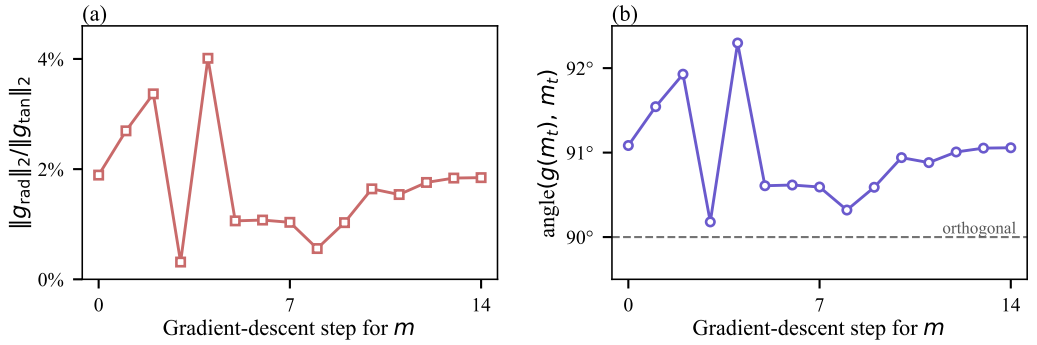


Figure 2: **The gradient remains nearly tangential throughout the optimization of m.** We track one random data sample (see Appendix A.2 for a larger-scale verification over the first 300 zsRE edits) from the CounterFact dataset on the Qwen2.5-7B-Instruct model. (a) The ratio $\|g_{\text{rad}}(m)\|_2/\|g_{\text{tan}}(m)\|_2$ of the radial to the tangential component of $g(m)$, which stays below 4.1%. (b) The angle between $g(m)$ and m, where $90°$ indicates that $g(m)$ lies in the tangent space at m; it stays within $2.3°$ of $90°$. This observation is consistent with normalization suppressing radial gradients, such that gradient descent primarily rotates m.

Empirical verification 2. We also validate this theoretical observation with empirical results. The results are in Figure 4. We see that, in practice, the magnitude of m does grow a lot.

4.2 Radial Stretching Affects Locality

A natural follow-up intuition is that unnecessary growth in the magnitude of m may make the downstream weight editing (i.e., Eq. 5) need more effort to achieve it, potentially causing greater interference with non-target knowledge and compromising editing locality. We will first formally discuss this issue theoretically, and then practically verify it with empirical results.

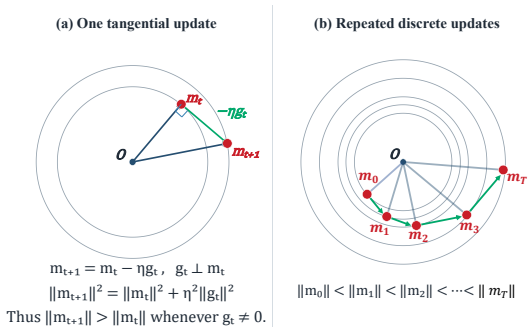


Figure 3: Geometry of tangential gradient descent. (a) One update: a step along the tangent direction at m_t leaves the sphere, so $\|m_{t+1}\| > \|m_t\|$ whenever $g_t \neq 0$. (b) Repeated updates: the magnitude grows at every step.

Consider the original hidden state as h and the target hidden state got from gradient descent as m. We denote m_{rot} as the hidden state that has the same magnitude as h and the same direction as m, that is, $m_{rot} = \frac{m||h||}{||m||}$. Replacing m with m_{rot} leaves the success of steering the model to the target answer nearly unchanged (Appendix A.3). From Eq. 5, we know that steering h to m will need the following parameter change $\Delta_{h\to m}$:

$$\Delta_{h\to m} = (m-h)s^\top, \qquad s = \frac{C^{-1}k}{k^\top C^{-1}k}, \tag{11}$$

where k is the key of h (see Sec. 3). Consequently, for arbitrary non-target knowledge k', we have

$$||\Delta_{h\to m}k'|| = ||(m-h)(s^\top k')|| = s^\top k'||m-h||. \tag{12}$$

Note that $s^\top k'$ is a scalar. Thus we know $||\Delta_{h\to m}k'|| \propto ||m-h||$ and we naturally have the following conclusion:

Theorem 3 (Radial stretching damages locality) *Compared to rotation-based steering, radial stretching leads to greater perturbation to non-target knowledge:*

$$\|\Delta_{h\to m}k'\| \geq \|\Delta_{h\to m_{\text{rot}}}k'\|, \quad \forall k'. \tag{13}$$

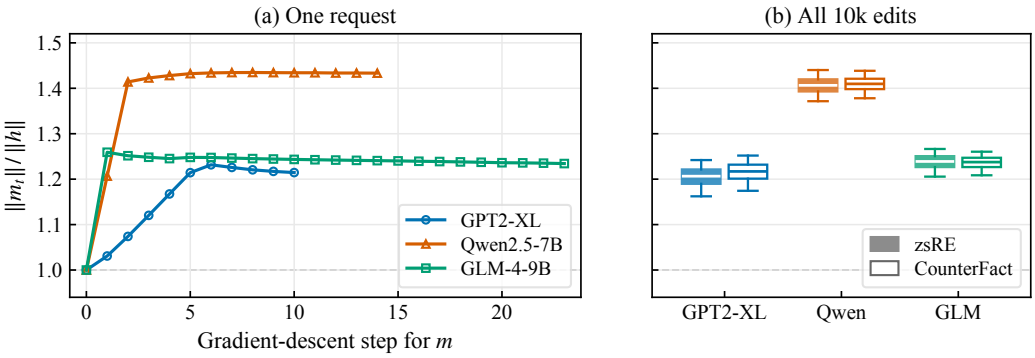


Figure 4: **The magnitude of m grows during the optimization of m.** (a) $\|m_t\|/\|h\|$ along the gradient-descent steps of one data sample from the COUNTERFACT dataset on each model; the magnitude rises within the first few steps and then stays far above $\|h\|$. (b) Distribution of $\|m\|/\|h\|$ over all 10k edits of each benchmark; boxes span the quartiles and whiskers the 5th and 95th percentiles. The medians are 1.21, 1.41 and 1.24 on the three models.

Table 1: $||\Delta k_j||^2$ for 10k edits on two widely used benchmarks with three different LLMs. Replacing the raw target m with the rescaled target m_{rot} consistently reduces the perturbation across all three models and both benchmarks.

Target	zsRE			CounterFact		
	GPT2-XL	Qwen2.5-7B	GLM-4-9B	GPT2-XL	Qwen2.5-7B	GLM-4-9B
m	1.69×10^{10}	7.49×10^{9}	1.25×10^{7}	1.66×10^{10}	1.28×10^{10}	1.69×10^{7}
m_{rot}	$\mathbf{1.28 \times 10^{10}}$	$\mathbf{4.35 \times 10^{9}}$	$\mathbf{9.13 \times 10^{6}}$	$\mathbf{1.26 \times 10^{10}}$	$\mathbf{7.58 \times 10^{9}}$	$\mathbf{1.21 \times 10^{7}}$

The proof is in Appendix A.9.

We further validate Theorem 3 empirically. From Eq. 4, we know the locality is achieved by the regularizer $\sum_j ||\Delta k_j||$, so we use this value to quantify the perturbation to non-target representations (Liu et al., 2026b). We replace the target m produced by a recent backbone method FE (Liu et al., 2026a) with $m_{rot} = \frac{m||h||}{||m||}$ and compare the corresponding results.

As shown in Table 1, replacing the raw target m with its rescaled counterpart m_{rot} reduces this perturbation by 24%, 42%, and 27% on GPT2-XL, Qwen2.5-7B-Instruct, and GLM-4-9B-Chat on zsRE, and by 24%, 41%, and 28% on COUNTERFACT, respectively. These results empirically verify that radial stretching damages locality.

4.3 METHOD

Till now, we have got the following insights: (1) The gradients are mostly tangential, guiding the hidden state to reduce the loss mainly through rotation (i.e., the radial magnitude contributes little to the loss reduction). (2) The radial magnitude will grow passively. (3) The growth in magnitude causes damage to locality (the preservation of non-target knowledge).

Based on the above observations, our method then becomes natural and straightforward. The direction of m encodes the desired edit, whereas preserving the magnitude of h helps maintain locality. We therefore adopt a simple yet concise strategy, *Spherical Editing* (SE): instead of using the raw m as the steering target, we rescale it to $m_{rot} = \frac{m||h||}{||m||}$ to match the magnitude of the original hidden state h and use a reparameterization trick for the gradient descent optimization. Consider the original hidden state as h, the subsequent LLM layers as f, and the optimizable residual as r; the formal formulation of SE is

$$m_{rot} = \frac{(h+r)||h||}{||h+r||}, \quad r^* = \arg\min_r \mathcal{L}(f(m_{rot}), y^*), \quad m^*_{rot} = \frac{(h+r^*)||h||}{||h+r^*||} \tag{14}$$

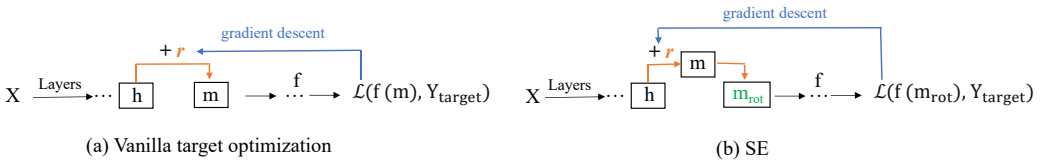


Figure 5: The architecture of our SE.

Table 2: Results on the zsRE dataset with $n = 10$k edits on GPT2-XL, Qwen2.5-7B-Instruct and GLM-4-9B-Chat.

Method	Target Knowledge		Non-target Knowledge			
	Success	Accuracy	D_{KL}	Top-1	Top-5	Top-10
GPT2-XL						
MEMIT	66.82	46.33	1.804	44.46	29.18	30.90
WISE	7.72	0.31	3.612	7.42	4.36	9.31
RECT	65.78	45.65	1.838	44.37	29.65	30.48
PRUNE	66.30	45.95	1.790	44.55	29.25	30.98
AlphaEdit	72.43	53.82	1.947	46.84	28.52	29.91
RLEdit	9.14	0.52	3.684	8.11	6.02	10.76
FE	76.00	56.62	1.583	60.86	33.16	32.74
SE (ours)	**76.01**	**56.84**	**1.236**	**64.55**	**36.10**	**37.20**
Qwen2.5-7B-Instruct						
MEMIT	82.28	59.09	4.595	5.26	16.40	22.05
WISE	22.84	5.08	7.436	18.27	21.16	23.08
RECT	79.83	56.88	4.631	5.44	16.68	22.77
PRUNE	81.95	58.70	4.540	5.34	16.48	22.12
AlphaEdit	88.22	71.72	4.856	10.37	21.88	25.77
RLEdit	37.26	14.62	6.842	13.88	19.42	21.08
FE	91.94	76.79	4.184	15.85	28.27	31.62
SE (ours)	**92.91**	**79.00**	**3.950**	**18.97**	**32.25**	**35.33**
GLM-4-9B-Chat						
MEMIT	85.67	67.58	1.731	44.56	51.50	50.55
WISE	29.16	7.62	3.984	19.84	24.18	26.52
RECT	85.51	67.52	1.772	45.29	52.37	51.21
PRUNE	85.42	67.34	1.700	44.70	51.65	50.72
AlphaEdit	82.74	66.21	1.862	43.21	50.72	49.74
RLEdit	43.18	13.84	3.748	19.62	26.17	28.04
FE	86.02	67.88	1.185	54.12	60.34	59.39
SE (ours)	**86.43**	**67.91**	**1.033**	**54.91**	**61.96**	**60.47**

The overall architecture comparison between vanilla target optimization and SE is in Figure 5.

5 Experiments

5.1 Setup

Baselines. FE (Liu et al., 2026a) is a recent improved version of the classical MEMIT (Meng et al., 2023) framework, and it achieves considerable improvements compared to previous methods. So we take FE as the backbone and main baseline. We also compare with several other recent methods: WISE (Wang et al., 2024), RECT (Gu et al., 2024), PRUNE (Ma et al., 2025), AlphaEdit (Fang et al., 2025) and RLEdit (Li et al., 2025b).

Models, datasets, and metrics. We evaluate on three LLMs: GPT2-XL (1.5B), Qwen2.5-7B-Instruct and GLM-4-9B-Chat. We follow FE to use two widely used datasets: zsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022). On each dataset we edit 10,000 facts in a single batch, the same setting as MEMIT. MEMIT and FE are run with the official code of FE, with

Table 3: Results on the COUNTERFACT dataset with $n = 10$k edits on GPT2-XL, Qwen2.5-7B-Instruct and GLM-4-9B-Chat.

Method	Target Knowledge		Non-target Knowledge			
	Success	Accuracy	D_{KL}	Top-1	Top-5	Top-10
GPT2-XL						
MEMIT	81.96	50.99	1.240	52.03	51.23	<u>50.30</u>
WISE	17.82	1.04	3.876	46.20	42.65	40.88
RECT	80.66	49.69	1.271	52.15	<u>51.68</u>	50.27
PRUNE	80.91	50.39	<u>1.219</u>	52.34	51.41	**50.49**
AlphaEdit	88.47	66.73	1.473	51.65	49.38	47.31
RLEdit	19.84	6.12	2.936	24.18	28.73	29.64
FE	<u>93.30</u>	<u>75.17</u>	1.324	<u>53.28</u>	49.95	48.42
SE (ours)	**93.32**	**75.35**	**1.218**	**55.26**	**51.70**	50.19
Qwen2.5-7B-Instruct						
MEMIT	98.77	85.16	2.374	41.08	41.55	39.91
WISE	27.08	6.74	8.362	21.77	24.08	25.69
RECT	98.65	84.32	2.422	41.32	41.76	39.71
PRUNE	98.62	84.95	2.340	41.17	41.68	40.04
AlphaEdit	99.17	92.24	2.598	40.21	39.28	37.14
RLEdit	51.26	21.08	6.428	11.96	18.84	20.71
FE	<u>99.40</u>	<u>96.71</u>	<u>1.257</u>	<u>55.57</u>	<u>57.36</u>	<u>56.15</u>
SE (ours)	**99.52**	**96.96**	**0.925**	**58.92**	**60.06**	**59.24**
GLM-4-9B-Chat						
MEMIT	99.08	96.73	1.077	60.12	57.38	56.66
WISE	21.62	5.71	5.274	24.68	29.16	31.27
RECT	99.01	96.52	1.096	60.21	57.32	56.42
PRUNE	99.02	96.62	1.060	60.26	57.46	56.75
AlphaEdit	98.44	96.21	1.252	58.34	55.42	53.28
RLEdit	42.36	16.27	3.844	24.21	29.08	30.76
FE	**99.28**	<u>97.53</u>	<u>0.940</u>	<u>63.99</u>	<u>62.03</u>	<u>61.44</u>
SE (ours)	<u>99.27</u>	**97.67**	**0.851**	**66.35**	**64.49**	**64.11**

hyperparameters tuned per model and dataset; SE keeps the edited layers and covariance statistics of FE and changes only the construction of the target. We report metrics on both target knowledge and non-target knowledge. For target knowledge, **Success** (Meng et al., 2022; 2023) is the teacher-forcing measure widely used in the literature, and **Accuracy** (Yang et al., 2025; Liu et al., 2026a) requires exact token match under free-form generation. For non-target knowledge, conventional ground-truth-based metrics have been shown to be inaccurate (Liu et al., 2026b); we therefore follow Liu et al. (2026b) to use more accurate metrics. They include two aspects: D_{KL}, the KL divergence between the output token distributions before and after editing, and Top-k, the overlap ratio of the top-k outputs, both computed on the locality prompts of each dataset.

5.2 RESULTS

Comparison with baseline methods. Tables 2 and 3 report the results on two widely used benchmarks and three different LLMs. SE gets the best performance on non-target knowledge in all six settings, lowering D_{KL} and raising Top-1, Top-5 and Top-10 at the same time; the largest Top-1 gains are on GPT2-XL on ZSRE (60.86 to 64.55) and on Qwen2.5-7B-Instruct on COUNTERFACT (55.57 to 58.92). On target knowledge SE matches or exceeds other methods in every column except the Success of GLM-4-9B-Chat on COUNTERFACT, where FE and SE differ by 0.01.

We also try to apply SE to the classic MEMIT backbone, and the results are reported in Appendix A.4. We consistently get improved performance.

Editing scaling. To see how the methods scale, we vary the number of edits from 32 to 10,000 on Qwen2.5-7B-Instruct. As shown in Figure 6, the three methods are close on target knowledge up to about 1,000 edits; beyond that, MEMIT drops first, and SE stays at or above FE. On non-target

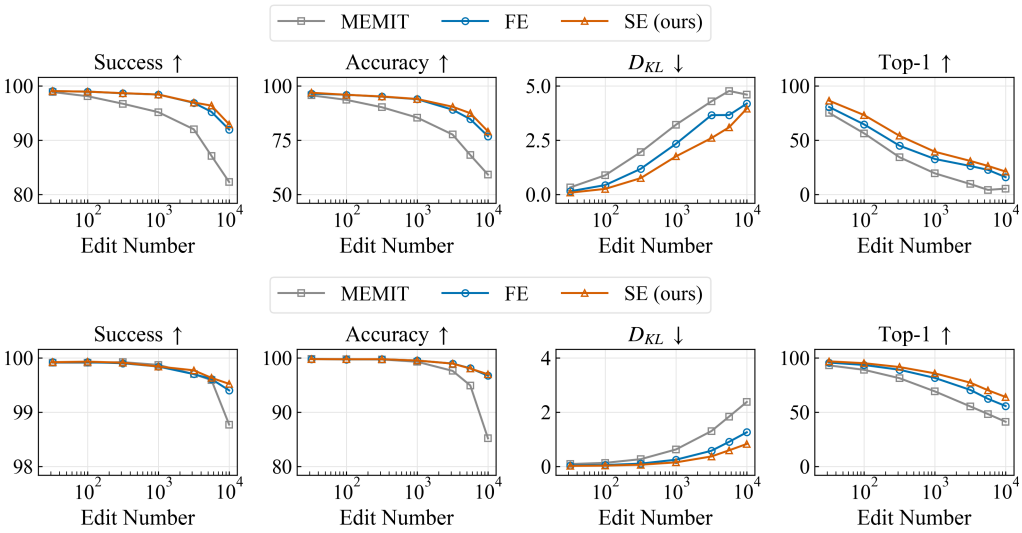


Figure 6: Scaling curves plot performance change against different editing numbers (log-scale) on Qwen2.5-7B-Instruct; top: the ZSRE dataset, bottom: the COUNTERFACT dataset.

Table 4: Sequential editing on GPT2-XL with 2,000 COUNTERFACT edits in 20 batches of 100. Bold values indicate improvements over the corresponding base method.

Method	Target Knowledge		Non-target Knowledge			
	Success	Accuracy	D_{KL}	Top-1	Top-5	Top-10
AlphaEdit	99.45	96.85	1.242	57.70	52.27	50.24
SE (on AlphaEdit)	**99.75**	**98.55**	**0.874**	**65.14**	**59.49**	**57.96**
FE	98.20	93.90	0.900	64.41	58.67	56.47
SE (on FE)	**99.50**	**96.89**	**0.821**	**66.57**	**61.93**	**59.47**
MEMIT	97.90	91.00	1.544	45.10	43.31	41.67
SE (on MEMIT)	**99.60**	**97.70**	**1.045**	**58.43**	**54.58**	**52.63**

knowledge the gap opens early and widens with the number of edits: SE has the lowest D_{KL} and the highest Top-1 at every scale on both datasets.

Sequential editing. We also test the performance of our method on sequential editing. Following the sequential editing setting of AlphaEdit (Fang et al., 2025), we apply 2,000 COUNTERFACT edits to GPT2-XL in 20 batches of 100, where each host follows the sequential procedure of its released code and keeps previously edited keys in its preservation term. SE remains effective in this setting, improving FE, MEMIT, and AlphaEdit on all six metrics (Table 4).

6 CONCLUSION

This paper studies the geometry of the target hidden state in locate-then-edit model editing. Because of normalization, the gradient of the prediction loss is nearly orthogonal to the target, so gradient descent rotates the target and increases its magnitude as a side effect, and the extra magnitude enlarges the perturbation to non-target knowledge. SE removes this side effect by keeping the target on the sphere of radius $\|h\|$ at every step. It modifies only the construction of the target, introducing neither additional parameters nor computational overhead. Experiments on three LLMs and two benchmarks show that SE consistently improves locality without sacrificing editing success, with its advantage becoming increasingly pronounced as the number of edited samples grows.

AI USE STATEMENT

In this work, we have not used generative AI tools for any prohibited or undisclosed research tasks listed in the ICLR 2027 AI Policy for Authors. We used generative AI tools to assist with software development and to edit the paper for readability. The proposed method was implemented by the authors. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies how the target hidden state is constructed in locate-then-edit knowledge editing. It involves no human subjects or personal data: all experiments use publicly released models and the public ZSRE and COUNTERFACT benchmarks. Like other knowledge editing methods, SE could be misused to insert false or harmful facts into a language model, so edited models should be evaluated before they are deployed.

REPRODUCIBILITY STATEMENT

The SE differentiable target-reparameterization implementation and experiment configurations for GPT2-XL and Qwen2.5-7B-Instruct and GLM-4-9B-Chat on CounterFact and zsRE, including FE reference settings, are available at `https://anonymous.4open.science/r/se-review-artifact-5869/`. The code supplement documents integration with the upstream FE implementation, dependencies, data selection, and the official evaluation procedure, together with the configuration coverage and validation scope.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yanbo Dai, Zhenlan Ji, Zongjie Li, and Shuai Wang. EAMET: Robust massive model editing via embedding alignment optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VJGVfJJ2wW.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, 2021.

Zilu Dong, Xiangqing Shen, Zinong Yang, and Rui Xia. ChainEdit: Propagating ripple effects in LLM knowledge editing through logical rule-guided chains. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13558–13571. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.665. URL https://aclanthology.org/2025.acl-long.665/.

Junfeng Fang, Houcheng Jiang, Kun Wang, Yunshan Ma, Jie Shi, Xiang Wang, Xiangnan He, and Tat-Seng Chua. AlphaEdit: Null-space constrained knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=HvSytvg3Jh.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. Model editing harms general abilities of large language models: Regularization to the rescue. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16801–16819. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.934. URL https://doi.org/10.18653/v1/2024.emnlp-main.934.

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

Tom Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*, volume 36, pp. 47934–47959, 2023.

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

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27510–27533. PMLR, 2025a. URL https://proceedings.mlr.press/v267/jiang25b.html.

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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16678–16702. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.acl-long.815/.

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

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. Association for Computational Linguistics, 2017. doi: 10.18653/v1/k17-1034. URL https://doi.org/10.18653/v1/K17-1034.

Qi Li and Xiaowen Chu. AdaEdit: Advancing continuous knowledge editing for large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4127–4149, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.208. URL https://aclanthology.org/2025.acl-long.208/.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. PMET: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18564–18572. AAAI Press, 2024. doi: 10.1609/aaai.v38i17.29818. URL https://doi.org/10.1609/aaai.v38i17.29818.

Xiaopeng Li, Shangwen Wang, Shasha Li, Shezheng Song, Bin Ji, Jun Ma, and Jie Yu. Rethinking residual distribution in locate-then-edit model editing. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://doi.org/10.52202/085713-2624.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 34920–34942, 2025b. URL https://openreview.net/forum?id=1jUXprrfcb.

Wei Liu, Hongkai Liu, Zhiying Deng, Yee Whye Teh, and Wee Sun Lee. From backward spreading to forward replay: Revisiting target construction in LLM parameter editing. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2605.00358.

Wei Liu, Haomei Xu, Hongkai Liu, Zhiying Deng, Ruixuan Li, Heng Huang, Yee Whye Teh, and Wee Sun Lee. Are we evaluating the edit locality of LLM model editing properly? *arXiv preprint arXiv:2601.17343*, 2026b. URL https://arxiv.org/abs/2601.17343.

Jun-Yu Ma, Hong Wang, Hao-Xiang Xu, Zhen-Hua Ling, and Jia-Chen Gu. Perturbation-restrained sequential model editing. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=bfI8cp8qmk.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations*. OpenReview.net, 2022a. URL https://openreview.net/forum?id=0DcZxeWfOPt.

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

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 *Advances in Neural Information Processing Systems*, volume 37, pp. 53764–53797, 2024.

Haoyu Xu, Pengxiang Lan, Enneng Yang, Guibing Guo, Jianzhe Zhao, Linying Jiang, and Xingwei Wang. Knowledge decoupling via orthogonal projection for lifelong editing of large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13194–13213, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.646. URL https://aclanthology.org/2025.acl-long.646/.

Wanli Yang, Fei Sun, Jiajun Tan, Xinyu Ma, Qi Cao, Dawei Yin, Huawei Shen, and Xueqi Cheng. The mirage of model editing: Revisiting evaluation in the wild. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15336–15354. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.745. URL https://aclanthology.org/2025.acl-long.745/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Shu Wu, Pengjie Ren, and Zhumin Chen. Uncovering overfitting in large language model editing. In *The Thirteenth International Conference on Learning Representations*. OpenReview.net, 2025. URL https://openreview.net/forum?id=t8qcGXaepr.

Ningyu Zhang, Bozhong Tian, Siyuan Cheng, Xiaozhuan Liang, Yi Hu, Kouying Xue, Yanjie Gou, Xi Chen, and Huajun Chen. InstructEdit: Instruction-based knowledge editing for large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 6633–6641, 2024. doi: 10.24963/ijcai.2024/733.

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

A APPENDIX

A.1 HOW TO GET THE TARGET HIDDEN STATE m

In Eq. 4, a critical step is to obtain the target hidden state m. Intuitively, m is an intermediate target: if the model can be made to produce this hidden state at the selected token and layer, the remaining layers should assign a high probability to the new answer. The standard MEMIT-style approach treats this hidden state as an optimizable variable and solves for it by gradient descent.

Concretely, suppose the rewrite query is tokenized as $Q = [q_1, \ldots, q_d, \ldots, q_n]$, where q_d is the decisive token, and the target answer is tokenized as $Y = [y_1, \ldots, y_p]$. Let layer l be the edited layer. Feeding the concatenated sequence $[Q, Y]$ through the first l layers gives the hidden states $h(Q, Y) = [h_{q_1}, \ldots, h_{q_d}, \ldots, h_{q_n}, h_{y_1}, \ldots, h_{y_p}]$, and the remaining layers give the prediction $\hat{Y} = f(h(Q, Y))$. The cross-entropy loss $H_c(Y, \hat{Y})$ is computed only on the answer tokens. With the model parameters frozen, the hidden state h_{q_d} is treated as the optimizable variable, and minimizing $H_c(Y, \hat{Y})$ with respect to it gives the target hidden state m.

Prior work shows that if the selected parameters are updated so that the original hidden state of the decisive token is mapped to m, the remaining layers are encouraged to generate the target answer (Meng et al., 2022; 2023). The preservation term in Eq. 4 regularizes the update so that non-target keys are affected as little as possible.

A.2 GRADIENT ORTHOGONALITY OVER MANY EDITS

Figure 2 tracks a single edit. Table 5 aggregates the same measurement over every gradient step of the first 300 ZSRE edits of each model, recorded along the optimization of m with the hyperparameters of the FE rows in Table 2; the number of steps differs across models because the optimization stops early once the loss falls below its threshold. The radial component of the gradient stays a few percent of the tangential one throughout, and the gradient never departs from the tangent space by more than 8.1° on any step of any edit.

Table 5: Radial-to-tangential ratio of the gradient of the prediction loss with respect to m, and the deviation of the angle between $g(m)$ and m from 90°, over all gradient steps of the first 300 ZSRE edits.

			$\|g_{\text{rad}}\|/\|g_{\text{tan}}\|$ (%)			angle to 90° (deg)	
Model	Edits	Steps	median	95th pct.	max	95th pct.	max
GPT2-XL	300	5136	1.62	4.84	9.78	2.77	5.59
Qwen2.5-7B-Instruct	300	3866	1.48	4.66	14.14	2.67	8.05
GLM-4-9B-Chat	300	14650	1.29	4.03	8.80	2.31	5.03

A.3 RADIAL SCAN ON ZSRE AND COUNTERFACT

To examine whether the magnitude of the optimized target m contributes meaningfully to the edit itself, we perform a radial scan on the first 300 edits of both ZSRE and COUNTERFACT. For each edit, we fix the direction of the optimized m and vary only its radius. Specifically, we inject $\alpha\|h\| \, m/\|m\|$ at the decisive token and evaluate the target loss and the success of predicting the target answer on the same prompts used to optimize m, without updating any model parameter.

As shown in Figure 7, on ZSRE the target loss remains nearly unchanged over a wide range of radii around $\|m\|$. Rescaling m to radius $\|h\|$, yielding m_{rot}, also largely preserves target-answer success: the success rates are 100.0%, 98.3%, and 98.0% on GPT2-XL, Qwen2.5-7B-Instruct, and GLM-4-9B-Chat, respectively, compared with 100.0%, 99.7%, and 98.3% for the raw m.

Figure 8 shows the same pattern on COUNTERFACT. The target loss is likewise nearly flat around $\|m\|$, and rescaling to m_{rot} yields target-answer success rates of 99.7%, 98.3%, and 100.0%, compared with 99.7%, 99.3%, and 100.0% for the raw m. The median ratios $\|m\|/\|h\|$ over these edits are 1.22, 1.41, and 1.24, respectively, consistent with the magnitude growth reported in Figure 4.

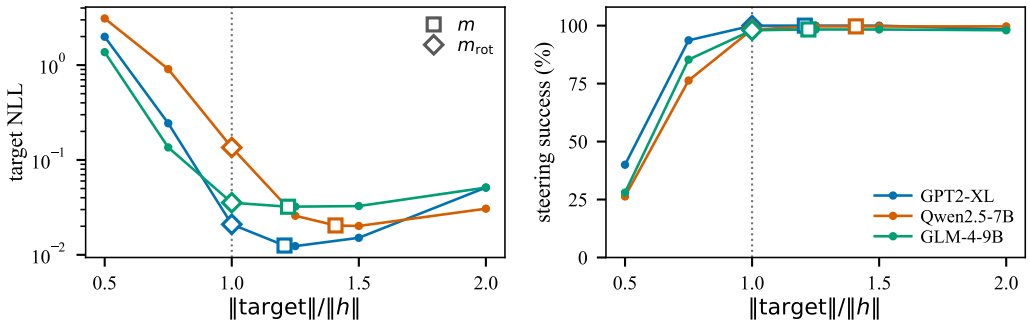


Figure 7: **Radial scan on the first** 300 **zsRE edits.** The direction of the optimized m is fixed while only its radius is varied relative to $\|h\|$. The left panel shows the target loss, and the right panel shows the fraction of edits for which the target answer is successfully predicted. Squares denote the raw m, while diamonds denote m_{rot}, obtained by rescaling m to radius $\|h\|$ without changing its direction.

Together, these results show that the direction of m carries the information needed for the edit, whereas the radial growth accumulated during optimization contributes little to achieving it.

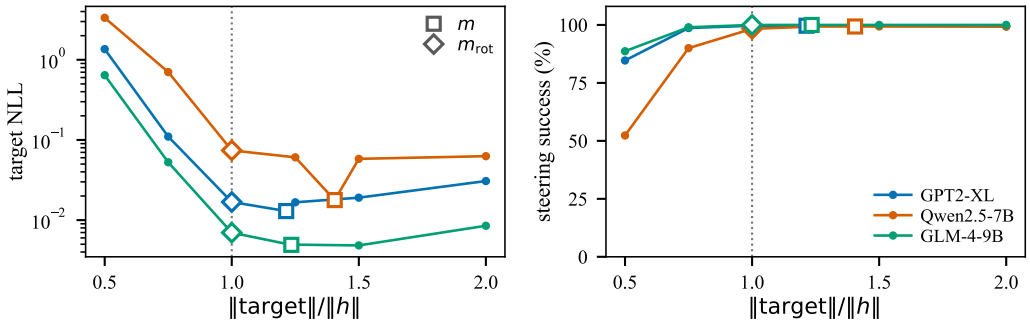


Figure 8: **Radial scan on the first** 300 **CounterFact edits.** The same protocol as in Figure 7 is used.

A.4 SE on the MEMIT host

MEMIT solves a single target at the deepest edited layer and then distributes the residual over the edited layers, whereas FE solves one target per layer. We apply the spherical projection inside the MEMIT solve, at the layer where the target is optimized, and change nothing else; the hyperparameters are the official MEMIT ones, in particular the clamp $c = 0.75$ and $\lambda = 15 \times 10^3$. Table 6 reports the result on GPT2-XL and zsRE with $n = 10$k edits. The per-layer Frobenius norms of the update are 48.0, 54.3, 64.4, 83.3 and 134.2, against 49.5, 56.2, 67.0, 87.0 and 139.8 for stock MEMIT, a ratio between 0.958 and 0.970 at every layer, so the gain is not bought by writing more. The targets themselves show the same picture as on the FE host: the projected target is shorter than the one MEMIT solves (mean norm 117.06 against 140.94) and points in almost the same direction (mean cosine 0.984).

A.5 The hidden state of m

We first clarify the location and geometry of the optimized target state. Consider an FFN output projection W at an editable layer. Let b denote the residual-stream state immediately before adding the FFN output, and let

$$v = Wk \tag{15}$$

Table 6: Applying the spherical projection inside the MEMIT solve, on GPT2-XL and zsRE with $n = 10$k edits. The host's own hyperparameters are unchanged.

Method	Target Knowledge		Non-target Knowledge			
	Success	Accuracy	D_{KL}	Top-1	Top-5	Top-10
MEMIT	66.82	46.33	1.804	44.46	29.18	30.90
MEMIT + SE	**70.55**	**50.89**	**1.646**	**48.76**	**31.44**	**33.16**

be the original FFN output for the decisive token. The post-residual hidden state is therefore

$$h = b + Wk. \tag{16}$$

Throughout this paper, the optimized target m is defined at the same post-residual location as h. Accordingly, the FFN output required to realize m is $v^* = m - b$, and the change that must be supplied by the parameter update satisfies

$$\begin{aligned} v^* - v &= (m - b) - Wk \\ &= m - (b + Wk) \\ &= m - h. \end{aligned} \tag{17}$$

Thus, defining m after the residual addition remains fully compatible with the standard LTE update: the desired FFN change is precisely the post-residual target displacement $m - h$.

A.6 Proof of Theorem 1

Proof. Define $\phi(\alpha) = \mathcal{J}(\alpha m)$. Eq. 8 implies $\phi'(1) = 0$. By the chain rule,

$$\begin{aligned} 0 = \phi'(1) &= \frac{\mathrm{d}}{\mathrm{d}\alpha}\mathcal{J}(\alpha m)\Big|_{\alpha=1} \\ &= m^\top \nabla_m \mathcal{J}(m). \end{aligned} \tag{18}$$

Hence the gradient lies in the tangent space

$$T_m \mathbb{S}^{d-1}_{|m|} = \{u \in \mathbb{R}^d : m^\top u = 0\} \tag{19}$$

of the hypersphere passing through m.

A.7 LayerNorm and RMSNorm

Normalization makes the branch gradient tangential. Both RMSNorm and LayerNorm are invariant to positive input rescaling when their numerical constant ε is omitted (Ba et al., 2016; Zhang & Sennrich, 2019). Specifically,

$$\mathrm{RMSNorm}(m) = \gamma \odot \frac{m}{\sqrt{|m|^2/d}} \tag{20}$$

satisfies

$$\mathrm{RMSNorm}(\alpha m) = \mathrm{RMSNorm}(m), \qquad \alpha > 0. \tag{21}$$

For LayerNorm, define the centering projection

$$P = I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top, \qquad c = Pm = m - \mu(m)\mathbf{1}, \tag{22}$$

where $\mu(m) = \mathbf{1}^\top m/d$. LayerNorm can be written as

$$\mathrm{LN}(m) = \gamma \odot \frac{Pm}{\sqrt{|Pm|^2/d}} + \beta. \tag{23}$$

It satisfies the stronger invariance

$$\mathrm{LN}(\alpha m + s\mathbf{1}) = \mathrm{LN}(m), \qquad \alpha > 0,\ s \in \mathbb{R}. \tag{24}$$

Let N denote either normalization operation and $J_N(m)$ its Jacobian. Differentiating

$$N(\alpha m) = N(m) \tag{25}$$

with respect to α at $\alpha = 1$ gives

$$J_N(m)m = 0. \tag{26}$$

For LayerNorm, differentiating with respect to the shift s additionally gives

$$J_{\mathrm{LN}}(m)\mathbf{1} = 0. \tag{27}$$

Now consider the next Pre-LN residual sublayer after the target state m:

$$x^+ = m + \mathcal{B}(N(m)), \tag{28}$$

where $\mathcal{B}$ denotes an attention or FFN branch. Let

$$q = \nabla_{x^+}\mathcal{L} \tag{29}$$

be the gradient arriving from the subsequent network. By the chain rule,

$$\begin{aligned} g(m) = \nabla_m \mathcal{L} &= \left(I + J_{\mathcal{B}} J_N\right)^\top q \\ &= \underbrace{q}_{g_{\mathrm{skip}}} + \underbrace{J_N^\top J_{\mathcal{B}}^\top q}_{g_{\mathrm{norm}}}, \end{aligned} \tag{30}$$

where the Jacobians are evaluated at the corresponding current activations. The two terms in Eq. 30 are respectively the gradient transmitted through the identity residual path and the gradient transmitted through the normalized computational branch.

Using Eq. 26, the normalized-branch component satisfies

$$\begin{aligned} m^\top g_{\mathrm{norm}} &= m^\top J_N^\top J_{\mathcal{B}}^\top q \\ &= \left(J_N m\right)^\top J_{\mathcal{B}}^\top q \\ &= 0. \end{aligned} \tag{31}$$

Therefore, normalization removes the radial gradient transmitted through the normalized branch exactly: g_{norm} is tangent to the hypersphere at m.

Effect of the numerical constant. The numerical constant ε used in practical implementations introduces a small radial leakage even within the normalized branch. Define

$$\begin{aligned} r_\varepsilon(m) &= \sqrt{|m|^2/d + \varepsilon}, \\ s_\varepsilon(m) &= \sqrt{|Pm|^2/d + \varepsilon}. \end{aligned} \tag{32}$$

Direct differentiation gives

$$\begin{aligned} J_{\mathrm{RMS},\varepsilon}(m)m &= \frac{\varepsilon}{r_\varepsilon(m)^3}\left(\gamma \odot m\right), \\ J_{\mathrm{LN},\varepsilon}(m)m &= \frac{\varepsilon}{s_\varepsilon(m)^3}\left(\gamma \odot Pm\right). \end{aligned} \tag{33}$$

The relative leakage is therefore of order

$$\mathcal{O}\left(\frac{\varepsilon}{\mathrm{Var}(m) + \varepsilon}\right), \tag{34}$$

which is negligible whenever the activation variance is much larger than ε.

A.8 The proof of Theorem 2

The proof is simple:

$$||m_{t+1}||^2 - ||m_t||^2 = -2\eta m_t^\top g_t + \eta^2 ||g_t||^2. \tag{35}$$

Since $m_t^\top g_t = 0$, we have

$$||m_{t+1}||^2 - ||m_t||^2 = \eta^2 ||g_t||^2 \geq 0. \tag{36}$$

Thus we have

$$||m_{t+1}|| \geq ||m_t|| \tag{37}$$

A.9 THE PROOF OF THEOREM 3

$$||\Delta_{h\to m}k'|| = s^\top k'||m-h||, \quad ||\Delta_{h\to m_{rot}}k'|| = s^\top k'||m_{rot}-h||. \tag{38}$$

We denote

$$h = au, \quad m = bv, \tag{39}$$

where $||u|| = ||v|| = 1$, and $a = ||h||$, $b = ||m||$. We have

$$||m-h||^2 - ||m_{rot}-h||^2 = (b-a)(b+a-2a\cos\theta), \tag{40}$$

where $\cos\theta = u^\top v$. Since $b \geq a$, we know $||m-h||^2 - ||m_{rot}-h||^2 \geq 0$. Thus, we have

$$||\Delta_{h\to m}k'|| \geq ||\Delta_{h\to m_{rot}}k'|| \tag{41}$$