Capacity Allocation at the Source: Sparse Target Optimization for LLM Knowledge Editing

Dongyao Chen[*]
Renmin University of China
dongyaochen@ruc.edu.cn

Jiayun Lei[*]
Renmin University of China
leijiayun@ruc.edu.cn

Zhiyuan Ji
Renmin University of China
zhiyuanji@ruc.edu.cn

Zhiying Deng[†]
Laboratory for Artificial Intelligence and New Forms of Education, Central China Normal University
zhiyingdzy@ccnu.edu.cn

Wei Liu[†]
School of Computing, National University of Singapore
weiliu87@nus.edu.sg

Xiaobo Sun[†]
Department of Human Genetics, Emory University School of Medicine
xiaobo.sun@emory.edu

Abstract

Knowledge editing has emerged as a promising approach for efficiently updating embedded knowledge in large language models (LLMs). It first computes an ideal target hidden state that steers the LLM toward the new output, and then edits the model parameters so that the original hidden state is mapped to this target. While most existing methods focus on the second stage of parameter editing in preserving other knowledge, we find that the upstream construction of the target hidden state also plays an important role in allocating the model's limited capacity space. We first establish a theoretical connection between interference in the parameter space and interference between target hidden states, showing that the two are positively correlated. Building on this insight, we propose a simple yet effective ℓ_1 regularization method that prunes away unnecessary "tentacles" of each knowledge vector, retaining only the essential components. With fewer extraneous tentacles, each edit affects less other knowledge, enabling better coexistence. Extensive experiments across multiple LLMs and benchmark datasets verify the effectiveness of our method. Code: https://github.com/wintersdragon-c/STO.

1 Introduction

Large language models (LLMs) encode a substantial amount of factual knowledge in their parameters, but that knowledge is inevitably incomplete, outdated, or erroneous. Updating such knowledge by continued pretraining or full fine-tuning is often computationally expensive and may undesirably perturb unrelated capabilities. This has motivated a growing line of work on model editing, whose goal is to modify a pretrained model so that it incorporates a small set of new facts while preserving unrelated behavior.

One of the mainstream paradigms in this literature is locate-then-edit (LTE) (Meng et al., 2022, 2023): First, it precisely locates a decisive token in the input text and identifies the decisive layer in the model (usually the MLP). Then, if the hidden state of the decisive token at the decisive layer can be steered toward a target hidden state (e.g, m in Figure 1), the model will ultimately output the desired target knowledge. Figure 1 provides a toy example for quick intuition. More details will be introduced in §3.

[*]Equal contribution.
[†]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

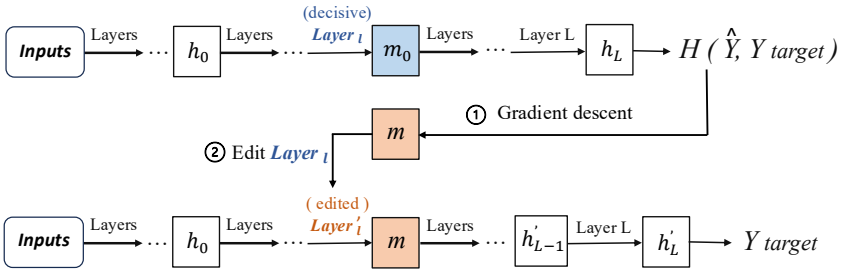


Figure 1: Overview of locate-then-edit model editing. The method first finds a target hidden state for the decisive token at the decisive layer via gradient descent (we call it upstream target construction), then edits a key subset of parameters to steer the original hidden state (m_0) toward this target (m), producing the desired output (Y_{target}) (we call it downstream parameter updating).

Despite its strengths, current LTE methods still face a central challenge: because different knowledge items are stored through the same set of parameters, how to better allocate this parameter space and reduce mutual interference among different knowledge, thereby improving the coexistence of more knowledge, remains an under-explored problem. While existing methods account for allocating new knowledge and old non-target knowledge to distinct subspaces when editing parameters based on the residual (i.e., during the second stage of Figure 1), the allocation at the upstream target construction stage remains less-explored.

In this work, we revisit model editing from the perspective of the residual representation (i.e., $m - m_0$ in Figure 1) that induces the final parameter update and find that we need to consider not only the downstream mechanism of parameter update, but also upstream target construction. The capacity space of an LLM is limited. The more new knowledge items that occupy the same region, the higher the interference among them, and the greater the disruption to that region. This not only lowers the success rate of injecting new knowledge into the region but also damages the old knowledge originally stored there. To address this, we attack the problem upstream, at the target construction stage: by spreading new knowledge more thinly across the capacity space, we make space for both new and old knowledge to coexist.

We first theoretically show that the interference between the parameter changes required for editing different pieces of knowledge is positively correlated with the inner product of their residuals. However, due to computational constraints, we cannot compute the residuals for all target knowledge simultaneously when solving via gradient descent, and thus cannot directly adopt inter-residual interference as an optimization regularizer. To address this issue, we try to reduce the expected interference by imposing a simple Lasso (Tibshirani, 1996) (i.e., ℓ_1 regularizer) on the residual vector r (which encourages sparsity in r), which can induce a more localized update. Lasso encourages the support of the knowledge representation to be sparse, thereby reducing the expected overlap between the supports of different edits. As a result, the Frobenius interactions among the model parameter updates are also expected to be small. This practical approximation is also quite intuitive: if we limit the residual's variation to only a few necessary dimensions during the residual computation stage, such that the modification of each piece of target knowledge occupies minimal capacity space, would reduce knowledge interference and relieve this allocation burden at its source.

Our contributions can be summarized as follows: (1) We theoretically show that reducing the inner product between the residuals of different knowledge items decreases interference when editing target knowledge. (2) We propose a simple and practical method that reduces the expected inner product between knowledge residuals, thereby lowering editing difficulty from the data preparation stage. (3) Experiments across various LLMs and datasets validate the effectiveness of our method.

2 Related Work

2.1 Locate-then-edit model editing

Locate-then-edit paradigm. Large language models (LLMs) encode a substantial amount of factual knowledge in their parameters, but that knowledge is inevitably incomplete, outdated, or erroneous, which motivates concerns on interpretability (Liu et al., 2025a, 2024), safety (Liu et al., 2025b), and reliability (Liu et al., 2026a). Recent work has increasingly focused on precise parameter editing, typically following a *locate-then-edit* (LTE) paradigm: identifying decisive tokens and layers and then modifying the corresponding hidden states or parameters to substitute target knowledge. Early methods such as ROME and MEMIT (Meng et al., 2022, 2023) introduced causal tracing to locate effective edit positions. Subsequent approaches, 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), NeuralEdit (Jiang et al., 2025b), and FE (Liu et al., 2026b), further extend this framework through improved localization, optimization, regularization, or sequential-editing strategies. With strong parameter efficiency and empirical performance, LTE methods have become a major paradigm for knowledge editing (Liu et al., 2026b).

Target construction of LTE. The LTE paradigm consists of two stages: (1) preparing target hidden states, which play a role analogous to labels in supervised learning, (2) using these target states to guide parameter updates. Most prior work focuses on the second stage, namely how to solve or regularize the parameter update, while the upstream construction of target states remains much less explored. MEMIT (Meng et al., 2023) argues that the target hidden state should remain close to the original hidden state and introduces an ℓ_2 distance term to enforce this. EAMET (Dai et al., 2026) further argues that the KL divergence over residuals should match that over keys. Our work also studies target construction, but addresses a different aspect: we study how target residuals should be allocated within the model's limited capacity space. By sparsifying target residuals, we reduce unnecessary overlap among edits, thereby mitigating interference with other edits and existing non-target knowledge.

2.2 Other Methods for Knowledge Editing

RAG-style inference-time knowledge editing. A parallel line of work maintains an external knowledge base queried at inference time to supply updated information (Hartvigsen et al., 2023; Zheng et al., 2023; Zhang et al., 2024; Jiang et al., 2024). Because they leave model parameters unchanged, we regard them as retrieval-based augmentation rather than genuine parameter editing, and they fall outside the scope of this paper.

Augmenting models with extra memory or hypernetworks. A second family equips LLMs with auxiliary memory modules or hypernetworks to absorb new knowledge. Methods such as MEND (Mitchell et al., 2022a), KE (De Cao et al., 2021), and RLEdit (Li et al., 2025b) exploit low-rank parameterizations for efficiency, yet typically exhibit poor generalization and require retraining for each new fact. Methods that offload knowledge to auxiliary parameters or networks—including T-Patcher (Huang et al., 2023), SERAC (Mitchell et al., 2022b), Grace (Hartvigsen et al., 2023), WISE (Wang et al., 2024a), and KDE (Xu et al., 2025)—face scalability issues as knowledge accumulates without retiring stale entries, and are generally more sensitive to hyperparameter choices than LTE-based approaches.

3 Preliminaries

Notation. Let f_θ denote a pre-trained LLM parameterized by θ. After editing, the updated model is denoted as f_{θ^*}. We define the target knowledge set as

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

where q_i^* is an input query that triggers a specific piece of factual knowledge (e.g., *"The president of the US is"*), y_i^* is the desired output after editing (e.g., *"Trump"*), and n denotes the number of knowledge items to be updated.

To encourage the preservation of other knowledge, a representative set

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

is typically sampled from a background corpus such as Wikipedia. Notably, y_j are obtained with $y_j = f_\theta(q_j)$, rather than really sampled from Wikipedia.

General editing objective. The overall goal of knowledge editing is to update specific factual associations while minimally affecting the model's behavior on other inputs. This objective can be formulated as:

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

where $\mathcal{L}_1$ and $\mathcal{L}_2$ denote loss functions for enforcing the target edit and preserving other knowledge, respectively, and λ controls the trade-off between them.

Locate-then-edit paradigm. To do editing more efficiently, prior work proposes the *locate-then-edit* (LTE) paradigm (Meng et al., 2022, 2023) that gives a succinct solution for it. LTE first identifies where a piece of factual knowledge is stored in the model and then applies targeted parameter updates. A widely adopted approach is *causal tracing* (Meng et al., 2022), which identifies:

- **Decisive token**: the token whose hidden representation most strongly determines the factual output, often corresponding to the final token of the subject entity in the input.
- **Decisive layers**: the model layer at which intervening on the hidden state of the decisive token most effectively steers the model toward the desired edit target, often corresponding to several MLP layers.

By intervening only on the hidden state of the decisive token at the decisive layer, these methods can selectively overwrite specific factual associations with high efficiency and precision. Due to its simplicity and strong empirical performance, this paradigm has become one of the mainstreams in the knowledge editing literature.

Despite differences in implementation, many locate-then-edit methods can be expressed as variants of the following optimization problem (Meng et al., 2023):

$$\begin{aligned} \Delta^* &= \arg\min_{\Delta} \Big(\sum_{i=1}^{n} \|(W+\Delta)k_i - m_i\|^2 + \frac{\lambda}{u} \sum_{j=n+1}^{n+u} \|(W+\Delta)k_j - m_j\|^2 \Big) \\ &= \arg\min_{\Delta} \Big(\sum_{i=1}^{n} \|\Delta k_i - (m_i - W k_i)\|^2 + \frac{\lambda}{u} \sum_{j=n+1}^{n+u} \|\Delta k_j\|^2 \Big) \\ W^* &= W + \Delta^*, \end{aligned} \tag{4}$$

where W denotes a selected subset of model parameters to be edited, k_i is the hidden representation of the decisive token at the layer immediately preceding W, and m_i is an idealized hidden representation aligned with the desired edit target. The second regularization term is designed to enforce orthogonality between Δ and other non-target token embeddings k_j. The target m_i is obtained by treating it as an optimizable parameter and applying gradient descent to minimize the cross-entropy between the current model output and the target output (see Appendix A for details).

Iteratively solving Eq. 4 via gradient descent is computationally expensive, since u is typically very large (e.g., the default is on the order of 40 million). A key advantage of LTE over fine-tuning is that Eq. 4 admits a closed-form solution that can be obtained in a single step, efficiently preserving massive non-target knowledge (Meng et al., 2023):

$$\Delta^* = (M_I - W K_I) K_I^\top (K_I K_I^\top + \frac{\lambda}{u} K_J K_J^\top)^{-1}, \tag{5}$$

where capital letters denote matrices formed by stacking the corresponding vectors. $R = M_I - W K_I$ is the residual for all the knowledge items.

4 Theoretical Analysis

LTE methods perform model editing based on Eq. 4, which includes two steps. The first step is to obtain the edit target signal m (analogous to a label in classification tasks), and the second step is to solve Eq. 4 based on this target. Although many existing methods focus on solving Eq. 4 more

effectively, directly obtaining a better m would require much less effort in the subsequent steering process, which is however a less explored avenue.

Here, we only seek a qualitative conclusion to guide the direction of method design. Therefore, we consider a simplified setting for a more detailed theoretical analysis, while still validating our method experimentally in the general setting. Specifically, we consider simultaneously editing two pieces of knowledge, a and b, and use the subscript a/b notation to denote the variables associated with these two pieces of knowledge. We denote the edit residual as $r = m_i - Wk_i$. The solution for Eq. 4 is (Meng et al., 2023)

$$\Delta^*_{a,b} = RK_1^\top (\sum_{i\in\{a,b\}} k_i k_i^\top + \frac{\lambda}{u}\sum_j k_j k_j^\top)^{-1}, \tag{6}$$

where $R = [r_a, r_b], K_1 = [k_a, k_b]$. We denote $C_0 = \frac{\lambda}{u}\sum_j k_j k_j^\top$. Typically, λ is a very large number. For example, a common choice is $\lambda = 15000 \gg 2$. Therefore, we usually have $\frac{\lambda}{u}\sum_j k_j k_j^\top \gg \sum_{i\in\{a,b\}} k_i k_i^\top$ and we make the following approximation for Eq. 6: $(\sum_{i\in\{a,b\}} k_i k_i^\top + \frac{\lambda}{u}\sum_j k_j k_j^\top)^{-1} \approx C_0^{-1}$.

Based on the above approximation, we have

$$\Delta^*_{a,b} \approx RK_1^\top C_0^{-1} = \sum_{i\in\{a,b\}} r_i k_i^\top C_0^{-1}. \tag{7}$$

Similarly, if we consider editing only a or only b separately, we obtain:

$$\Delta^*_a = r_a k_a^\top (k_a k_a^\top + C_0)^{-1} \approx r_a k_a^\top C_0^{-1}, \quad \Delta^*_b = r_b k_b^\top (k_b k_b^\top + C_0)^{-1} \approx r_b k_b^\top C_0^{-1}. \tag{8}$$

Thus, we have

$$||\Delta_{a,b}||_F^2 \approx ||\Delta_a + \Delta_b||_F^2 = ||\Delta_a||_F^2 + ||\Delta_b||_F^2 + 2\langle \Delta_a, \Delta_b\rangle_F, \tag{9}$$

where $\langle \Delta_a, \Delta_b\rangle_F$ is the Frobenius interaction. If the interaction is large and positive, the two updates will reinforce each other, leading to a larger overall perturbation on other non-target knowledge (i.e., the second term of Eq. 4). If the interaction is large and negative, the two updates a and b will partially cancel each other out, indicating that they are competing in the same parameters (e.g., if Δ_a and Δ_b point in exactly opposite directions with identical magnitudes, they will completely cancel each other out). If the interaction is small, the two updates are more likely to coexist with little mutual interference, and also cause less disruption to the model's existing knowledge.

Based on that, for any two edits, we aim to make their expected interaction $|\langle \Delta_a, \Delta_b\rangle_F|$ small and thus making the interference of different knowledge edits smaller. We denote

$$v_a^\top := k_a^\top (k_a k_a^\top + C_0)^{-1}, \tag{10}$$

which is irrelevant to r_a. Thus we have (note that all equations in the following section are exact and do not involve approximations from the above simplified setting)

$$\Delta_a = r_a v_a^\top, \quad \Delta_b = r_b v_b^\top. \tag{11}$$

Then, we can get

$$\langle \Delta_a, \Delta_b\rangle_F = \mathrm{tr}(\Delta_a^T \Delta_b) = (r_a^\top r_b)(v_a^\top v_b). \tag{12}$$

Theorem 1. *If the target residuals of two edits are orthogonal, the corresponding rank-one model updates are Frobenius-orthogonal:*

$$r_a^\top r_b = 0 \Longrightarrow \langle \Delta_a, \Delta_b\rangle_F = 0. \tag{13}$$

Based on the above analysis, to reduce interference between the parameter updates induced by different knowledge edits, we only need to decrease the interference between their residuals r, which suggests a new avenue for improving model editing from the upstream stage of target construction.

5 A simple and practical solution

Although our analysis suggests that decreasing the interference between the residuals r of different knowledge edits may be a promising direction, it then leads to the question of how we can apply this

criterion in practice constrained under GPU-memory constraints: different pieces of knowledge are handled independently when computing their corresponding residuals, and during gradient descent to solve for the residual r of a given piece of knowledge, the other knowledge edits are not accessible (due to GPU limitations, it is infeasible to run gradient descent for a large number of knowledge edits simultaneously).

In this study, we present a simple and practical method. The overall idea is to make each residual occupy as little space as possible, so that the expected overlap among the residuals of different pieces of knowledge is reduced. Specifically, we use the least absolute shrinkage and selection operator (Lasso) (Tibshirani, 1996) to retain only the most essential components of the residual while suppressing trivial patterns. The final method is very simple: when using gradient descent to obtain m in Eq. 4, we only need to add an additional Lasso regularization (i.e., ℓ_1) term:

$$m^* = \arg\min_m (\mathcal{L}_{MEMIT}(m) + \lambda_1 \|m - Wk\|_1), \tag{14}$$

where $\|m - Wk\|_1$ is the added Lasso regularizer, and $\mathcal{L}_{MEMIT}(m)$ is the original basic objective used in MEMIT (Meng et al., 2023) (see Appendix B for details), which typically contains an LLM-related loss (e.g., cross-entropy) and an ℓ_2 regularizer (i.e., $\|m - Wk\|_2$).

In terms of Eq. 12, the Lasso regularizer encourages most trivial dimensions in r_a to be near zero and preserves only the components most important for knowledge a, thereby reducing the expected overlap between different residuals. This distributes knowledge more sparsely across the LLM's capacity space, both lowering the difficulty of injecting new knowledge and reducing conflicts with existing non-target knowledge. Appendix G formalizes the connection between sparsity and orthogonality under a probabilistic support model.

It is worth noting that aside from the cross-entropy, $\mathcal{L}_{MEMIT}$ already contains an ℓ_2 distance term (i.e., $\|m - Wk\|_2$) in addition to the standard cross-entropy. Yet even on top of this existing distance loss, introducing our Lasso loss still yields substantial performance gains. This means that our method's effectiveness does not stem from merely shrinking the magnitude of each dimension (ℓ_2 alone can already achieve that). Our distinct contribution lies in systematically demonstrating the motivation for introducing Lasso. Diagnostic empirical verification is provided in §6.3.

6 Experiments

6.1 Setup

Baselines. We call our method **STO** (**S**parse **T**arget **O**ptimization), which is built upon the standard MEMIT framework. MEMIT serves as the direct baseline. We also compare with several recent methods, including , WISE (Wang et al., 2024a), RECT (Gu et al., 2024), PRUNE (Ma et al., 2025), AlphaEdit (Fang et al., 2025), RLEdit (Li et al., 2025b), EAMET(Dai et al., 2026).

Models, datasets, and metrics. Experiments are run on an Ascend 910B-64G NPU. We evaluate STO on three LLMs, which are Qwen2.5-7B-Instruct, GPT-J (6B), and GPT-2 XL (1.5B), following prevailing experimental settings in the model editing literature. Two benchmark datasets are used: Multi-CounterFact (MCF) (Meng et al., 2022) and ZsRE (Levy et al., 2017), each with 2,000 edited knowledge items, following the recent AlphaEdit protocol. All edits are performed in a single batch (massive editing). For baselines, hyperparameters selection follows their official code or the popular platform EasyEdit (Wang et al., 2024b); STO keeps the matched MEMIT editing layers and covariance settings while changing the target-residual construction objective. Appendix F reports the sparsity coefficients, their selection procedure, and sensitivity results. We report metrics along three standard dimensions. **Efficacy** measures whether the target knowledge is correctly injected. **Generalization** evaluates robustness under natural linguistic variation, probing whether the model has internalized the knowledge rather than memorized a specific surface form. **Specificity** quantifies how well unrelated model behavior is preserved after editing. Together, these three dimensions capture the core trade-offs between correctness, robustness, and unintended side effects. For both efficacy and generalization, we report **Success** (Meng et al., 2023), a teacher-forcing–based measure, and **Accuracy** (Liu et al., 2026c), which requires exact token match under free-form generation. For specificity, conventional ground-truth–based metrics have been shown to miss distributional drift induced by editing (Liu et al., 2026c). We therefore adopt the ground-truth–free behavior-deviation metrics proposed by Liu et al. (2026c): D_{KL}, which measures KL-divergence between pre- and post-edit output distributions, and Top-k, which quantifies the overlap ratio of the top-k support.

Table 1: The results on the MCF and ZsRE datasets with GPT2-XL.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	34.3	12.1	29.4	6.1	1.27	37.4	40.1	39.6
WISE	22.0	5.0	43.0	12.5	4.95	39.0	41.0	42.0
MEMIT	92.3	75.0	80.0	43.3	0.85	62.9	60.2	59.2
PRUNE	92.1	75.1	79.8	42.9	0.82	63.0	60.6	59.6
RECT	92.0	74.8	79.5	42.5	0.83	63.2	60.6	59.7
AlphaEdit	94.8	86.8	82.2	45.8	0.98	60.5	55.8	55.0
EAMET	90.8	71.6	78.3	41.2	0.84	62.3	60.4	59.3
STO	**95.3**	**87.6**	**84.3**	**46.3**	**0.76**	**67.1**	**63.2**	**61.5**
The ZsRE dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	17.2	2.1	9.5	0.3	0.99	22.0	15.4	18.4
WISE	13.2	1.5	15.4	0.8	2.37	6.5	4.2	9.5
MEMIT	85.0	61.2	80.7	52.2	0.81	65.7	62.4	63.9
PRUNE	82.6	61.0	80.5	52.3	0.80	65.9	62.0	64.2
RECT	81.0	60.1	79.0	52.2	0.79	65.8	62.4	64.0
AlphaEdit	93.0	82.9	88.1	68.3	0.93	67.0	63.4	64.1
EAMET	86.1	60.3	79.1	51.9	0.85	66.1	62.3	65.0
STO	**93.1**	**84.6**	**89.6**	**69.4**	**0.61**	**71.6**	**68.2**	**69.5**

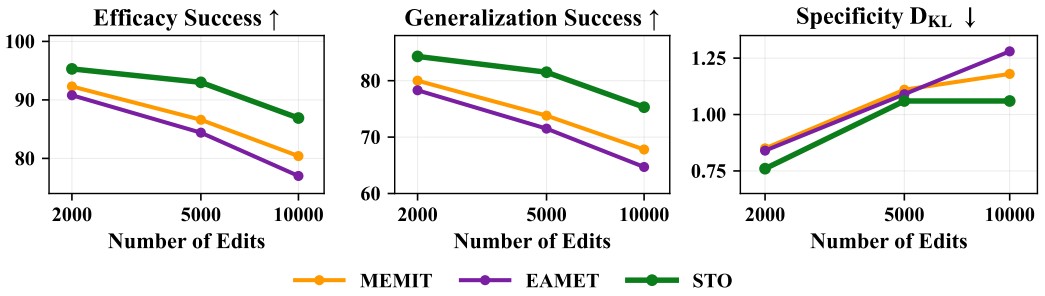


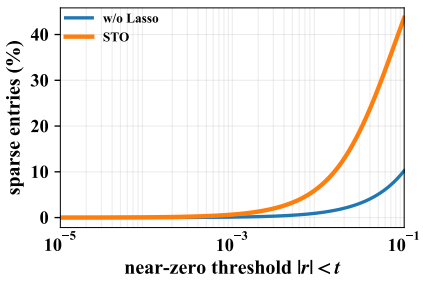
Figure 2: GPT2-XL MCF scaling to 10000 knowledge items.

6.2 Results on standard benchmarks

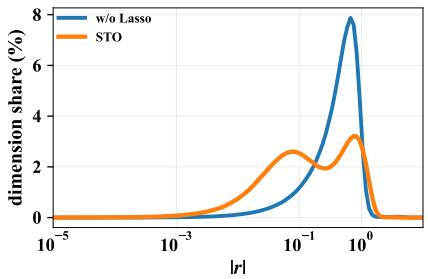
The comparison with various recent model editing methods is shown in Table 1 and Table 2 (results on GPT-J are in Appendix D). We can observe that our STO achieves the best performance across efficacy, generalization, and specificity. STO can also be combined with PRUNE, RECT, and AlphaEdit by modifying upstream target construction while retaining their downstream solvers. Table 3 shows that STO improves overall efficacy, generalization, and specificity across all three frameworks on GPT2-XL; Appendix E provides the corresponding Qwen2.5-7B-Instruct results and integration details. We further scale our method to editing 5000 and 10,000 knowledge items and compare against the most relevant baselines MEMIT and EAMET (see §2.1 for the relevance). The results are shown in Figure 2 and Appendix C. Our advantage becomes even more pronounced at these scales, validating the scalability of our approach. This aligns with our earlier hypothesis: if the interaction is large and positive, the two updates reinforce each other, leading to a larger overall perturbation on non-target knowledge (i.e., lower specificity); if the interaction is large and negative, updates a and b partially cancel each other out, indicating that they compete for the same parameters (i.e., lower efficacy and generalization).

Table 2: The results on the MCF and ZsRE datasets with Qwen2.5-7B-Instruct.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	66.5	34.4	49.0	17.5	5.32	3.3	8.4	8.0
WISE	28.0	6.5	53.0	16.0	6.80	25.0	27.0	27.5
MEMIT	99.2	92.3	97.4	70.4	2.23	41.9	41.7	38.6
PRUNE	99.2	92.4	97.2	70.0	2.25	41.5	41.4	39.0
RECT	99.1	92.2	97.0	69.7	2.24	41.6	41.5	39.1
AlphaEdit	99.0	91.6	97.9	69.8	2.65	38.2	36.0	34.5
EAMET	99.1	92.0	97.3	70.0	1.98	42.3	42.0	38.9
STO	**99.9**	**99.9**	**98.0**	**71.7**	**0.65**	**70.0**	**68.7**	**67.4**
The ZsRE dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	48.7	22.3	32.4	9.0	4.20	34.6	6.1	7.0
WISE	25.8	5.8	49.0	16.8	7.40	22.0	24.0	25.0
MEMIT	91.3	79.5	90.5	73.5	3.97	36.0	35.5	39.9
PRUNE	91.1	79.6	90.3	72.9	4.02	35.8	35.4	39.7
RECT	90.9	79.3	90.1	72.6	4.00	35.9	35.5	39.8
AlphaEdit	90.7	78.8	91.0	72.8	4.35	33.0	31.0	35.6
EAMET	94.4	85.2	92.7	79.7	3.50	40.5	39.1	42.2
STO	**96.6**	**88.9**	**95.4**	**82.6**	**1.57**	**63.8**	**60.0**	**62.7**



(a) Near-zero fraction across thresholds



(b) Residual magnitude density

Figure 3: Verifying feature sparsification. (a) We use varying thresholds (x-axis) to define near-zero values; the y-axis shows the proportion of dimensions falling below each threshold. (b) Distribution of per-dimension values: the x-axis denotes the value of each dimension in the residual, and the y-axis denotes density.

We next design more fine-grained experiments to validate our method in greater detail.

6.3 Diagnostic experiments

To better validate our method, we design targeted experiments addressing several specific questions.

Question (1): Can the Lasso regularizer really lead to sparse residuals?

We compare the residuals obtained by our method against those obtained without the Lasso constraint, measuring the proportion of dimensions whose values are close to zero. We use several different thresholds to determine whether a dimension is considered near-zero. The results are shown in Figure 3. Figure 3(a) shows that our method substantially increases sparsity. Figure 3(b) plots the sample density (y-axis) of residual values across different magnitude ranges (x-axis). STO's

Table 3: Compatibility of STO with PRUNE, RECT, and AlphaEdit on GPT2-XL (2,000 edits per dataset). Each combination's absolute results are followed by Δ (combination minus its baseline in Table 1), computed before rounding. Changes are in percentage points except for D_{KL}; bold Δ values indicate improvements.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
PRUNE + STO	94.4	84.8	83.0	44.2	0.70	68.5	64.4	62.7
Δ	**+2.3**	**+9.7**	**+3.2**	**+1.3**	**−0.12**	**+5.5**	**+3.8**	**+3.1**
RECT + STO	93.6	83.4	81.8	42.1	0.68	69.1	65.0	63.3
Δ	**+1.6**	**+8.6**	**+2.3**	−0.4	**−0.15**	**+5.9**	**+4.4**	**+3.6**
AlphaEdit + STO	99.7	99.2	89.9	52.9	0.49	75.4	70.8	69.0
Δ	**+4.9**	**+12.4**	**+7.7**	**+7.1**	**−0.49**	**+14.9**	**+15.0**	**+14.0**
The ZsRE dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
PRUNE + STO	93.0	83.3	88.5	66.6	0.41	76.7	73.4	74.8
Δ	**+10.4**	**+22.3**	**+8.0**	**+14.3**	**−0.40**	**+10.8**	**+11.4**	**+10.6**
RECT + STO	92.9	82.6	88.2	65.8	0.40	76.7	73.4	74.8
Δ	**+11.9**	**+22.5**	**+9.2**	**+13.6**	**−0.39**	**+10.9**	**+11.0**	**+10.8**
AlphaEdit + STO	97.5	98.7	95.2	80.2	0.47	73.5	70.0	71.3
Δ	**+4.5**	**+15.8**	**+7.1**	**+11.9**	**−0.46**	**+6.5**	**+6.6**	**+7.2**

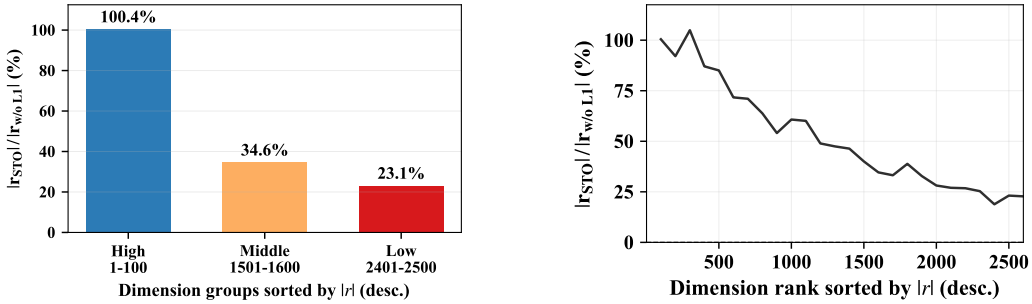
density function clearly exhibits two peaks: Compared to the no-Lasso baseline, the rightmost curve almost coincides with the original, while a second peak emerges around 0.1, indicating that certain unimportant dimensions have indeed been compressed while the important features are preserved. Our goal is to preserve important features while removing unnecessary ones. Is the phenomenon in Figure 3(a) due to simple universal weight shrinkage? To further verify this, we sort the dimensions of residual vector r by magnitude and aggregate them in groups of 100, then examine the compression ratio across dimensions with and without Lasso. The results are shown in Figures 4(a) and 4(b). Figure 4(a) reports the compression ratio for the low, middle, and high regions, while Figure 4(b) shows the compression ratio across all dimensions continuously. This indicates that our method does not simply compress all dimensions; rather, it largely preserves the originally high-magnitude, important dimensions (retaining approximately 100.4%) while more aggressively compressing the low-magnitude, unimportant ones. This aligns with the design principle of Lasso (Tibshirani, 1996) that it does feature selection rather than simple weight shrinkage.

Question (2): How much does our method affect the original cross-entropy loss?

Cross-entropy measures how accurately a given piece of knowledge is steered toward the target. Therefore, to ensure successful downstream editing, our method should not significantly affect the cross-entropy. We select 2,000 samples from MCF dataset and conduct experiments using Qwen2.5-7B-Instruct. We report the mean and median cross-entropy values, as well as the residual inner product corresponding to Eq. 12 (computed over 500K sampled pairs).

As shown in Table 4, the inner product between residuals is significantly reduced, while the cross-entropy increases only marginally. Combined with the finding from Question (1)—that our method preserves high-magnitude dimensions and primarily compresses low-magnitude ones—this confirms that we retain the necessary dimensions while suppressing unimportant ones.

Notably, although a higher cross-entropy would in principle suggest a disadvantage in editing success rate (Efficacy), our method instead achieves higher Efficacy, particularly on the harder ZsRE dataset. This indicates that reducing inter-knowledge interference indeed improves editing success.



(a) Representative groups: high, middle, and low

(b) All dimension groups

Figure 4: Dimension-level $|\frac{r_{sto}}{r_{w/o\ \ell_1}}|$. We sort the dimensions of r in descending order by absolute value and compute $|\frac{r_{sto}}{r_{w/o\ \ell_1}}|$. For ease of visualization, we group every 100 adjacent dimensions and report the average ratio per group.

Table 4: We report the mean/average cross-entropy (CE) loss and sample 500K (a, b) pairs to report $|r_a^\top r_b|$. We achieve much lower knowledge interference (Eq. 12) but only marginally higher CE.

Method	$\lvert r_a^\top r_b\rvert$ (mean)↓	$\lvert r_a^\top r_b\rvert$ (median)↓	CE (mean)	CE (median)
STO	251.2152	195.9598	0.0161	0.0021
w/o ℓ_1	541.1298	442.3562	0.0152	0.0020

Question (3): Does the performance gain simply come from reducing the magnitude of each dimension in the residual r?

The comparison between our method and MEMIT directly addresses this concern. Our approach can be viewed as augmenting MEMIT with an additional Lasso constraint (ℓ_1) on top of its existing ℓ_2 regularization on the residual. These two regularizers play complementary roles: ℓ_2 performs weight shrinkage by uniformly attenuating parameter magnitudes, while ℓ_1 induces sparsity and thereby acts as an implicit feature selector, retaining informative components while pruning redundant ones. The performance gain of our method over MEMIT therefore indicates that feature selection, beyond mere weight shrinkage, is essential in this setting. In practice, naively strengthening the ℓ_2 term in MEMIT does not resolve the issue and in fact hurts performance, since ℓ_2 penalizes all features indiscriminately and ends up suppressing the informative ones together with the redundant ones.

7 Conclusion, limitations, and future work

This paper analyzes, from the perspective of target construction, how to better allocate target knowledge within the limited capacity space of LLMs during editing. We first establish a theoretical connection between interference in the parameter space and interference between target residuals, showing that the two are positively correlated. Building on this insight, we propose STO, a simple yet effective ℓ_1 regularization method that prunes away unnecessary "branches" of each knowledge vector, leaving only the essential components. With fewer extraneous branches, knowledge items are less likely to entangle, and the interference between residuals (and consequently in the LLM parameter space) is substantially reduced. Our results suggest that target construction is also a fruitful direction for improving the reliability of model editing. We hope this work encourages further investigation into how the structure of edit targets, not only the mechanism of parameter updates, shapes the success of LLM model editing. One limitation is that we do not have results for LLMs larger than 14B, mainly due to two practical reasons. One reason is that the hyperparameters for most baselines are often unavailable for larger LLMs. In addition, larger-scale experiments exceed our current computational resources. Nevertheless, the settings we use are commonly used in the literature, and we leave the study of larger LLMs to future work.

Acknowledgments and Disclosure of Funding

This work is supported by the National Natural Science Foundation of China under Grant 62507018 and the Hubei Provincial Natural Science Foundation of China under Grant JCZRMS202601468. It is also supported by the National Research Foundation, Singapore, and the Ministry of Digital Development and Information (MDDI) under the Singapore Global AI Visiting Professorship Program (Award No. AIVP-2024-002). We also thank the AFAC (Advanced Fintech AI Competition) Academic Acceleration Program for supporting this research. The authors declare no competing interests.

References

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.

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, ICLR 2025, Singapore, April 24-28, 2025*. 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, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, 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. *Advances in Neural Information Processing Systems*, 36:47934–47959, 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, Guojun Ma, Mingyang Wan, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Anyedit: Edit any knowledge encoded in language models. *CoRR*, abs/2502.05628, 2025a. doi: 10.48550/ARXIV.2502.05628. URL https://doi.org/10.48550/arXiv.2502.05628.

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), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, 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, et al. 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), Vancouver, Canada, August 3-4, 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 *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 18564–18572. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29818. URL https://doi.org/10.1609/aaai.v38i17.29818.

Xiaopeng Li, Sheng Wang, Siyuan Li, Shuang Song, Bin Ji, Min Jun, and Jun Yu. Rethinking residual distribution in locate-then-edit model editing. In *Advances in Neural Information Processing Systems*, 2025a.

Zherui Li, Houcheng Jiang, Hao Chen, Baolong Bi, Zhenhong Zhou, Fei Sun, Junfeng Fang, and Xiang Wang. Reinforced lifelong editing for language models. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=1jUXprrfcb.

Wei Liu, Zhiying Deng, Zhongyu Niu, Jun Wang, Haozhao Wang, YuanKai Zhang, and Ruixuan Li. Is the mmi criterion necessary for interpretability? degenerating non-causal features to plain noise for self-rationalization. *Advances in Neural Information Processing Systems*, 37:117636–117656, 2024.

Wei Liu, Zhiying Deng, Zhongyu Niu, Jun Wang, Haozhao Wang, Zhigang Zeng, and Ruixuan Li. Breaking free from mmi: A new frontier in rationalization by probing input utilization. In *International Conference on Learning Representations*, volume 2025, pp. 55351–55374, 2025a.

Wei Liu, Zhongyu Niu, Lang Gao, Zhiying Deng, Jun Wang, Haozhao Wang, and Ruixuan Li. Adversarial cooperative rationalization: The risk of spurious correlations in even clean datasets. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39126–39146. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/liu25av.html.

Wei Liu, Zhiying Deng, Lang Gao, Yuankai Zhang, Haozhao Wang, Jun Wang, Rui Zhang, Heng Huang, and Ruixuan Li. Escaping graphical causal priors: Multi-generator cooperation for rationalization. *Artificial Intelligence*, pp. 104601, 2026a.

Wei Liu, Liu Hongkai, Zhiying Deng, Yee Whye Teh, and Wee Sun Lee. From backward spreading to forward replay: Revisiting target construction in LLM parameter editing. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pp. 76538–76553. PMLR, 06–11 Jul 2026b. URL https://proceedings.mlr.press/v306/liu26ax.html.

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?, 2026c. 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, ICLR 2025, Singapore, April 24-28, 2025*. 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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6f1d43d5a82a37e89b0665b33bf3a182-Abstract-Conference.html.

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 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, ICLR 2022, Virtual Event, April 25-29, 2022*. 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 *International Conference on Machine Learning*, pp. 15817–15831. PMLR, 2022b.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

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 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/60960ad78868fce5c165295fbd895060-Abstract-Conference.html.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 82–93, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-demos.9. URL https://aclanthology.org/2024.acl-demos.9/.

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 *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/.

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, ICLR 2025, Singapore, April 24-28, 2025*. 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.

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.

Table 5: The results on the MCF and ZsRE datasets with GPT-J.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	77.6	52.0	45.3	18.7	2.97	6.7	13.7	16.2
WISE	27.5	7.0	51.0	15.5	4.50	45.8	48.0	49.5
MEMIT	99.7	99.4	92.6	66.9	0.55	72.6	69.4	69.1
PRUNE	99.7	99.4	92.7	66.6	0.52	73.0	70.0	69.8
RECT	99.8	99.3	92.4	66.0	0.53	72.9	70.1	69.9
AlphaEdit	99.7	98.8	94.0	66.2	0.67	70.2	64.4	63.6
EAMET	99.7	98.8	91.5	67.0	0.63	70.4	67.8	67.6
STO	**99.9**	**99.5**	**95.7**	**67.8**	**0.48**	**76.6**	**73.4**	**72.8**
The ZsRE dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
RLedit	88.1	72.7	73.3	54.8	0.41	75.6	71.4	72.5
WISE	50.7	19.2	48.5	21.0	11.01	1.8	2.1	2.2
MEMIT	99.1	96.6	94.6	92.5	0.23	82.1	74.3	76.0
PRUNE	**99.2**	**96.8**	94.5	92.4	0.22	82.2	74.5	76.1
RECT	99.0	96.5	94.4	92.3	0.23	82.3	74.6	76.0
AlphaEdit	85.5	72.2	74.0	59.7	0.46	80.1	72.3	74.5
EAMET	90.4	85.3	78.9	64.6	0.19	87.4	79.6	81.7
STO	96.8	89.1	**94.7**	**92.8**	**0.18**	**87.6**	**79.9**	**82.5**

A How to get the target hidden state m

In Eq. 4, a critical step is to obtain the target hidden state m. Intuitively, m plays the role of an intermediate target representation: if the model can be made to produce this representation at the selected token and layer, then the remaining layers should assign high probability to the desired new answer. The standard MEMIT-style approach treats this hidden state as an optimizable variable and solves for it by gradient descent, using a loss that encourages the model to predict the target answer. We describe the MEMIT loss in Appendix B.

Concretely, suppose the rewrite query is tokenized as $Q = [q_1, q_2, \ldots, q_d, \ldots, q_n]$, where q_d denotes the decisive token, and the desired answer is tokenized as $Y = [y_1, y_2, y_3]$. Let layer-l be the final decisive layer. After feeding the concatenated sequence $[Q, Y]$ through the first l layers of the LLM, we obtain hidden states $h(Q, Y) = [h_{q_1}, \ldots, h_{q_d}, \ldots, h_{q_n}, h_{y_1}, \ldots, h_{y_3}]$.

We denote the remaining layers of the model by $\hat{Y} = f(h(Q, Y))$. The cross-entropy loss is computed only on the target answer tokens, and is denoted by $H_c(Y, \hat{Y})$. During target-state optimization, the model parameters are frozen, and the hidden representation of the decisive token, h_{q_d}, is treated as the optimizable variable. Minimizing $H_c(Y, \hat{Y})$ with respect to this hidden state gives an optimized representation, which is taken as the target hidden state m.

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

B The loss function of MEMIT

This appendix summarizes the target-state optimization loss used in MEMIT (Meng et al., 2023). Before applying its closed-form parameter update, MEMIT first constructs a target hidden state m_i

Table 6: **Qwen2.5-7B-Instruct 10000-edit comparison.**The MCF block reports completed runs on on the same exact 10,000-edit manifest.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
MEMIT	73.7	5.7	62.7	2.0	5.69	8.2	12.5	12.3
EAMET	74.9	10.2	65.1	3.5	5.20	9.0	12.4	10.3
STO	**91.6**	**59.0**	**86.7**	**32.0**	**4.72**	**19.7**	**20.5**	**18.6**

Table 7: **GPT2-XL 10000-edit comparison.**The MCF block reports completed runs on on the same exact 10,000-edit manifest.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
MEMIT	80.4	47.6	67.8	26.6	1.18	53.4	52.0	51.2
EAMET	77.0	43.4	64.7	25.1	1.28	52.2	51.4	50.1
STO	**86.9**	**61.4**	**75.3**	**31.9**	**1.06**	**56.8**	**55.4**	**53.1**

for each edit request. This target state specifies what the representation of the decisive token should become so that the frozen remaining layers of the model generate the desired new answer.

Given an edit request (s_i, r_i, o_i^*), let $p(s_i, r_i)$ denote the rewrite prompt, and let h_i^L be the hidden state of the last subject token at the target layer L. MEMIT parameterizes the candidate target state as $m_i = h_i^L + \delta_i$, where δ_i is an optimizable residual. During this target-construction step, all model parameters are frozen; only δ_i is updated.

The main objective is to make the model assign high probability to the desired object o_i^* when the decisive-token hidden state is replaced by m_i:

$$\mathcal{L}_{\mathrm{CE}}(m_i) = \frac{1}{P}\sum_{j=1}^{P} -\log P_{G(h_i^L \leftarrow m_i)}\left[o_i^* \mid x_j \oplus p(s_i, r_i)\right], \tag{15}$$

where $G(h_i^L \leftarrow m_i)$ denotes a forward pass in which the original hidden state h_i^L is replaced by the candidate target state m_i. The random prefixes $\{x_j\}_{j=1}^{P}$ make the optimized target state robust to different surface forms of the prompt, rather than overfitting to a single prompt.

MEMIT also keeps the optimized target close to the model's original representation. In the notation of our main paper, let $m_{0,i} = Wk_i$ denote the pre-edit value representation produced by the selected MLP module, and let $r_i = m_i - m_{0,i}$ denote the target residual. The MEMIT target-state loss can be written as

$$\mathcal{L}_{\mathrm{MEMIT}}(m_i) = \mathcal{L}_{\mathrm{CE}}(m_i) + \lambda_2 \|m_i - m_{0,i}\|_2^2. \tag{16}$$

The first term encourages the edited model to predict the new object, while the ℓ_2 term discourages the target state from moving too far from the original representation.

Our method keeps this MEMIT objective and adds an ℓ_1 regularizer on the same residual, giving $\mathcal{L}_{\mathrm{STO}}(m_i) = \mathcal{L}_{\mathrm{MEMIT}}(m_i) + \lambda_1 \|m_i - m_{0,i}\|_1$. Equivalently, STO optimizes $\mathcal{L}_{\mathrm{CE}}(m_i) + \lambda_2 \|m_i - m_{0,i}\|_2^2 + \lambda_1 \|m_i - m_{0,i}\|_1$. Thus, the original MEMIT loss finds a target hidden state that induces the desired answer while remaining close to the original representation; our added ℓ_1 term further encourages the residual $r_i = m_i - m_{0,i}$ to use only a smaller set of essential dimensions before the standard MEMIT closed-form update is applied.

C Scaling to 10000 knowledge items

We further evaluate whether STO remains effective as the number of simultaneous edits increases. Tables 6 and 7 report the 10,000-edit MCF results on Qwen2.5-7B-Instruct and GPT2-XL.

D Results on additional LLM

Corresponding to Table 1 and Table 2, the results of the GPT-J model are in Table 5.

E Compatibility with additional LTE frameworks

STO acts on upstream target-residual construction. To combine it with PRUNE, RECT, or AlphaEdit, we add the ℓ_1 penalty when optimizing each target residual and then pass the resulting targets to the framework's existing downstream solver. Each combination retains the framework's downstream procedure.

Table 8 reports the Qwen2.5-7B-Instruct compatibility results on MCF and ZsRE, with 2,000 edits per dataset. Each combination is shown with its absolute results and changes from the corresponding baseline across all eight metrics; the GPT2-XL results are in Table 3 in the main text. On Qwen2.5-7B-Instruct, all three frameworks obtain higher efficacy and better specificity after adding STO. Most generalization metrics also improve, although some MCF metrics decrease slightly.

Table 8: Compatibility of STO with PRUNE, RECT, and AlphaEdit on Qwen2.5-7B-Instruct (2,000 edits per dataset). Each combination's absolute results are followed by Δ (combination minus its baseline in Table 2), computed before rounding. Changes are in percentage points except for D_{KL}; bold Δ values indicate improvements.

The MCF dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
PRUNE + STO	99.7	99.5	97.6	69.5	0.68	69.1	68.6	67.9
Δ	**+0.5**	**+7.1**	**+0.4**	−0.5	**−1.57**	**+27.6**	**+27.2**	**+28.9**
RECT + STO	99.7	99.5	97.5	69.4	0.67	68.9	68.3	67.7
Δ	**+0.6**	**+7.3**	**+0.5**	−0.3	**−1.57**	**+27.3**	**+26.8**	**+28.6**
AlphaEdit + STO	99.8	99.7	97.8	71.8	0.58	71.1	69.7	68.8
Δ	**+0.8**	**+8.1**	−0.1	**+2.0**	**−2.07**	**+32.9**	**+33.7**	**+34.3**
The ZsRE dataset								
Methods \ Metrics	Efficacy		Generalization		Specificity			
	Success ↑	Accuracy ↑	Success ↑	Accuracy ↑	D_{KL} ↓	Top-1 ↑	Top-5 ↑	Top-10 ↑
PRUNE + STO	96.8	97.8	95.6	93.7	1.55	61.1	64.0	65.1
Δ	**+5.7**	**+18.2**	**+5.3**	**+20.8**	**−2.47**	**+25.3**	**+28.6**	**+25.4**
RECT + STO	96.8	97.9	95.7	94.0	1.56	60.8	63.6	64.6
Δ	**+5.9**	**+18.6**	**+5.6**	**+21.4**	**−2.44**	**+24.9**	**+28.1**	**+24.8**
AlphaEdit + STO	96.7	92.2	95.3	84.1	1.81	63.4	66.2	68.0
Δ	**+6.0**	**+13.4**	**+4.3**	**+11.3**	**−2.54**	**+30.4**	**+35.2**	**+32.4**

F Sparsity coefficient and sensitivity analysis

STO retains the original MEMIT target objective, including its ℓ_2 term, and adds the residual penalty $\lambda_1 \|m - Wk\|_1$. Table 9 lists the coefficients used for the standard MEMIT + STO experiments. We selected them by monitoring optimization dynamics in TensorBoard and manually balancing the loss terms. Apart from λ_1, the sensitivity comparisons use the corresponding MEMIT hyperparameters from EasyEdit.

We vary λ_1 over $\{0, 3 \times 10^{-4}, 6 \times 10^{-4}, 9 \times 10^{-4}\}$ on GPT2-XL and Qwen2.5-7B-Instruct, with 2,000 edits per dataset. The zero-penalty runs provide the matched MEMIT baselines for this sweep. Table 10 reports efficacy, generalization, and specificity together.

Table 9: Selected sparsity coefficients for the standard MEMIT + STO experiments.

Model	Dataset	λ_1
GPT2-XL	MCF	9×10^{-4}
GPT2-XL	ZsRE	6×10^{-4}
Qwen2.5-7B-Instruct	MCF / ZsRE	3×10^{-4}
GPT-J-6B	MCF / ZsRE	3×10^{-4}

Every tested positive coefficient improves efficacy accuracy and Top-1 preservation over its corresponding zero-penalty baseline. Increasing the positive coefficient reduces D_{KL} within each setting, but can reduce efficacy or generalization. For example, on GPT2-XL / MCF, increasing λ_1 from 3×10^{-4} to 9×10^{-4} reduces D_{KL} from 0.97 to 0.76 while generalization accuracy changes from 60.0 to 46.3. The coefficient therefore controls an editing–preservation trade-off, and the selected operating point need not maximize every metric.

Table 10: Sensitivity to the sparsity coefficient with 2,000 edits per setting. Coefficients in the first column are in units of 10^{-4}; bold coefficients mark the selected operating points.

λ_1	Eff-S ↑	Eff-A ↑	Gen-S ↑	Gen-A ↑	D_{KL} ↓	Top-1 ↑
GPT2-XL / MCF						
0	92.3	75.0	80.0	43.3	0.85	62.9
3	98.6	95.8	92.2	60.0	0.97	63.0
6	97.8	93.6	89.2	53.7	0.85	65.2
9	95.3	87.6	84.3	46.3	0.76	67.1
GPT2-XL / ZsRE						
0	85.0	61.2	80.7	52.2	0.81	65.7
3	95.2	90.8	92.0	74.7	0.62	70.9
6	93.1	84.6	89.6	69.4	0.61	71.6
9	88.8	72.0	83.7	56.3	0.38	77.8
Qwen2.5-7B-Instruct / MCF						
0	99.2	92.3	97.4	70.4	2.23	41.9
3	99.9	99.9	98.0	71.7	0.65	70.0
6	99.6	99.4	96.0	66.5	0.62	78.7
9	98.6	98.3	93.7	62.0	0.47	82.8
Qwen2.5-7B-Instruct / ZsRE						
0	91.3	79.5	90.5	73.5	3.97	36.0
3	96.6	88.9	95.4	82.6	1.57	63.8
6	96.6	97.7	94.2	90.7	1.07	74.1
9	96.7	99.1	94.1	92.3	0.89	79.4

G Probabilistic connection between sparsity and orthogonality

This appendix formalizes the support-overlap motivation for STO. Probability is taken over random edit residuals in a fixed model and coordinate system. For $r_a, r_b \in \mathbb{R}^d$, define their nonzero supports as $T_i = \{t : r_{i,t} \neq 0\}$ for $i \in \{a, b\}$. Let $X = |T_a \cap T_b|$ and $q_{ab,t} = \Pr(t \in T_a \cap T_b)$.

Theorem 2 (Support-overlap bound)**.** *For arbitrary joint distributions of the two supports,*

$$\Pr(r_a^\top r_b = 0) \geq \Pr(X = 0) \geq \max\left\{0, 1 - \sum_{t=1}^{d} q_{ab,t}\right\}. \tag{17}$$

Proof. Writing $X = \sum_{t=1}^{d} \mathbf{1}\{t \in T_a \cap T_b\}$ gives $\mathbb{E}[X] = \sum_t q_{ab,t}$ by linearity of expectation. Since X is a nonnegative integer, Markov's inequality implies $\Pr(X \geq 1) \leq \mathbb{E}[X]$. Hence $\Pr(X = 0) \geq 1 - \mathbb{E}[X]$, together with the trivial lower bound of zero. Disjoint supports imply $r_a^\top r_b = 0$, which

proves the first inequality. Orthogonality may also occur through cancellation when the supports overlap, so disjointness is sufficient but not necessary. □

To recover a product form, assume that, for each coordinate t, the support indicators of the two edits are independent. Defining $\pi_{i,t} = \Pr(t \in T_i)$ then gives

$$\mathbb{E}[X] = \sum_{t=1}^{d} \pi_{a,t}\pi_{b,t}, \qquad \Pr(r_a^\top r_b = 0) \geq \max\left\{0, 1 - \sum_{t=1}^{d} \pi_{a,t}\pi_{b,t}\right\}. \tag{18}$$

Independence between different coordinates is not required. Under this model, decreasing the coordinate-usage probabilities can improve the lower bound. More generally, the relevant quantity is the expected joint overlap $\sum_t q_{ab,t}$.

An analogous magnitude bound connects overlap to the residual factor in Eq. 12. If $|r_{a,t}| \leq B_a$ and $|r_{b,t}| \leq B_b$ almost surely for every coordinate, then

$$\mathbb{E}\left[|r_a^\top r_b|\right] \leq B_a B_b \, \mathbb{E}[X] = B_a B_b \sum_t q_{ab,t}. \tag{19}$$

This follows by bounding the absolute inner product by $\sum_t |r_{a,t} r_{b,t}| \leq B_a B_b X$. For fixed key-side factors v_a, v_b, Eq. 12 multiplies this bound by $|v_a^\top v_b|$ for the corresponding rank-one update interaction.

Scope of the argument. These statements provide a conditional motivation for sparsity, not a guarantee that an ℓ_1 penalty makes residuals orthogonal. A smaller support or ℓ_1 norm alone need not reduce joint overlap: two sparse residuals can concentrate on the same coordinates. The probability bound is vacuous when expected overlap is at least one, and a larger lower bound does not establish monotonicity of the actual orthogonality probability. Finite-step target optimization typically produces near-zero coefficients rather than exact zeros, so exact supports are an idealization. The empirical inner-product and cross-entropy comparison in Table 4 tests the resulting residual geometry directly. Finally, Frobenius orthogonality of updates is a geometric statement and does not by itself guarantee preservation of all non-target model behavior.

H Application note

Our subsequent AFAC competition entry on financial data factor development cites this published work and uses STO as a technical prototype for further development.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state the paper's scope: a target-construction method for locate-then-edit model editing, a residual-interference analysis, and experiments on multiple LLMs and editing benchmarks. The claims are supported by the theoretical analysis, standard benchmark tables, diagnostic experiments, and scaling appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper states the method's scope in the Conclusion: One limitation is that we do not have results for LLMs larger than 14B, mainly due to two practical reasons. One reason is that the hyperparameters for most baselines are often unavailable for larger LLMs. In addition, larger-scale experiments exceed our current computational resources. Nevertheless, the settings we use are commonly used in the literature, and we leave the study of larger LLMs to future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical analysis states the two-edit setting, residual notation, covariance approximation used for intuition, and the exact residual-interaction identity in Eq. 12. The theorem follows directly from that identity and is cross-referenced with the method motivation and diagnostics.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the models, datasets, metrics, edit setting, target-construction objective, and benchmark protocol in the Experiments section and appendices. Code, configurations, and running instructions are provided at `https://github.com/wintersdragon-c/STO`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public repository `https://github.com/wintersdragon-c/STO` provides code, configuration files, and running instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Experiments section describes the evaluated models, MCF and ZsRE datasets, massive-editing setup, baselines, metrics, and matched MEMIT settings. Additional target-state optimization and loss details are provided in the appendices and artifact repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The main experiments are controlled matched-manifest comparisons: all methods are evaluated on the same edit sets with the same evaluation pipeline, and the paper does not claim random-seed or hyperparameter-search statistical significance. We do not report error bars or confidence intervals because this is not standard in the model-editing benchmark protocol; the release artifacts include manifests and metric-generation code for auditing or extending the comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The release artifacts document the compute setup used to reproduce the experiments, including accelerator requirements, model/dataset preparation, and commands for target construction, editing, and evaluation. The paper also discusses scalability through the 10,000-edit appendix experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research is limited to theoretical analysis and offline benchmark evaluation of a methodological improvement for locate-then-edit knowledge editing. It uses existing benchmark datasets and pretrained language models, and does not involve human-subject

experiments, private or sensitive data collection, user-facing deployment, or the release of newly trained foundation models. The artifact release is limited to research code, configurations, and reproduction instructions, and does not include edited model checkpoints. To the best of our knowledge, the work conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work is a methodological study of capacity allocation in locate-then-edit knowledge editing. It provides theoretical analysis and offline benchmark experiments on how sparse target construction affects edit interference. The paper does not introduce a deployed system, collect or release new human data, train or release a new foundation model, or provide edited model checkpoints. Therefore, we do not identify direct societal impacts or specific harmful downstream consequences beyond the general research context of model-editing methods.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: The paper does not release a new pretrained model, an edited model checkpoint, or a new high-risk dataset. The released artifact is limited to public research code, configurations, and reproduction instructions for offline benchmark evaluation. Access to the underlying pretrained models and benchmark datasets remains governed by their original

licenses and usage terms. These release boundaries reduce the risk that the artifact directly enables deployment of altered models while still supporting scientific reproducibility.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the benchmark datasets, pretrained model families, editing baselines, and public editing frameworks used in the experiments. The public artifact repository documents external dependencies and points users to the original asset licenses and access terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new release asset is the public code and reproduction package at `https://github.com/wintersdragon-c/STO`. It documents the method implementation, configurations, evaluation scripts, and reproduction workflow for offline benchmark evaluation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, participant interaction, or human-subject experiments. All experiments are offline evaluations on existing model-editing benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved, so IRB or equivalent human-subjects review is not applicable. The study uses existing public benchmarks and offline model evaluations.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the subject of study: the paper edits pretrained language models and evaluates their behavior before and after editing. The method does not rely on a separate LLM as an undisclosed component for data generation, labeling, or evaluation beyond the documented model-editing experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.

- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.