REACT: Representation Extraction And Controllable Tuning to Overcome Overfitting in LLM Knowledge Editing

Anonymous ACL submission

Abstract

Large language model editing methods frequently suffer from overfitting, wherein factual updates can propagate beyond their intended scope, overemphasizing the edited target even when it's contextually inappropriate. To address this challenge, we introduce **REACT** (Representation Extraction And Controllable Tuning), a unified two-phase framework designed for precise and controllable knowledge editing. In the initial phase, we utilize tailored stimuli to extract latent factual representations and apply Principal Component Analysis with a simple learnbale linear transformation to compute a directional "belief shift" vector for each instance. In the second phase, we apply controllable perturbations to hidden states using the obtained vector with a magnitude scalar, gated by a pre-trained classifier that permits edits only when contextually necessary. Relevant experiments on EVOKE benchmarks demonstrate that **REACT** significantly reduces overfitting across nearly all evaluation metrics, and experiments on COUNTERFACT and MQuAKE shows that our method preserves balanced basic editing performance (reliability, locality, and generality) under diverse editing scenarios.

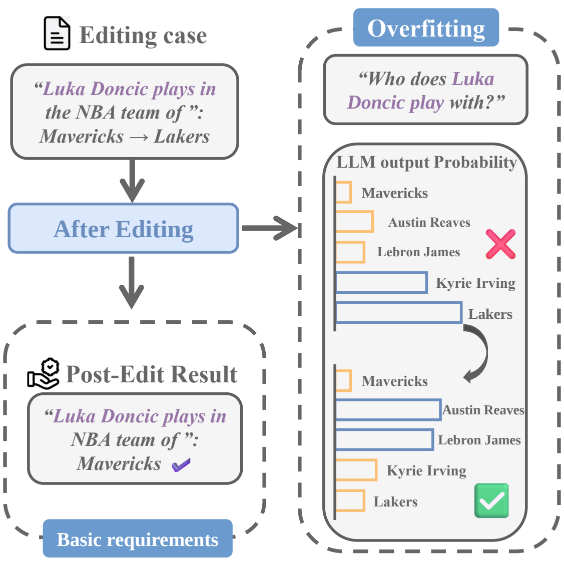

Figure 1: Illustration of overfitting in LLM editing. Overfitting occurs when the model disproportionately emphasizes the edited target fact, even in contexts irrelevant to the edit. As shown on the right side, after editing the fact about Luka Doncic's team to "Lakers," the overfitted model incorrectly assigns high probability to "Lakers" even for a query about Doncic's teammates.

1 Introduction

Large language models (LLMs) have become indispensable in modern applications, powering a wide array of systems from chatbots to content generators (Zhao et al., 2023; Xu et al., 2024). Despite their widespread utility, ensuring that these models maintain up-to-date and accurate factual information remains a critical challenge, particularly when extensive retraining is impractical (Zhang et al., 2024b). This necessity has spurred interest in the field of knowledge editing, where targeted updates to a model's internal knowledge base are pursued without compromising overall performance (Wang et al., 2023; Yao et al., 2023; Cheng et al., 2023).

Recent advances in knowledge editing have sought to address these issues by incrementally incorporating new facts into LLMs (De Cao et al., 2021). However, many existing approaches encounter significant challenges, like *overfitting during editing process* (Zhang et al., 2024a). Concretely, this occurs when a model, after being updated with new knowledge, becomes excessively specialized to the edited samples. For example, consider an update where the statement "Luka Doncic plays in the NBA team of **Mavericks**" is corrected to "Luka Doncic plays in the NBA team of **Lakers**." In an overfit scenario, when queried with "Who does Luka Doncic play with?", the model may still disproportionately favor the edit target but not the correct answer—assigning a high probability to "Mavericks"—while the probabilities for more contextually appropriate responses, such as teammates like Austin Reaves or LeBron James, remain undesirably low, as illustrated in Figure 1. These limitations hinder the practical deployment of such techniques in real-world systems.

In response to these challenges, we propose a novel framework that leverages a dual-phase representation pipeline to perform targeted knowledge edits. In the first phase-*Extracting Latent Knowledge Representations* (§3.1)-we employ tailored input prompts to extract the model's latent factual representations. Then we use Principal Component Analysis and a simple linear transformation to compute a directional vector that encapsulates the latent "belief" shift associated with the edit. In the subsequent phase-*Controllable Perturbing Representations Selectively* (§3.2)-we introduce controlled perturbations to the model's hidden states, guided explicitly by a pre-trained classifier (§3.3). This classifier functions as a gating mechanism, discerning precisely when edits should be applied based on the hidden states of the content. We perturb the hidden states from Transformer decoder block of all layers based on the product between the original hidden state and the directional vector. We also use a learnable scalar to control the magnitude of the perturbation.

To prove effectiveness of our method, we conduct experiments and analyze the results on COUNTERFACT, MQuAKE (§5.1) and EVOKE (§5.2), with detailed experimental settings (§4).

Our contributions can be summarized as follows:

- We propose a dual-phase editing framework, which extracts latent factual representation shifts and applies controllable perturbations to precisely edit models, effectively **overcoming the critical overfitting issue** in existing knowledge editing methods.

- Unlike prior parameter-based methods, our approach operates directly on the model's hidden states, employing classifier-driven gating to ensure edits are accurately applied, thus providing explicit control over knowledge modification.

- Comprehensive evaluation on COUNTERFACT, MQuAKE, and EVOKE datasets demonstrates that our method significantly reduces overfitting while achieving balanced improvements in Reliability, Generality, and Locality metrics.

2 Preliminaries

2.1 Large Language Models

Autoregressive large language models (LLMs) employ the Transformer architecture, where hidden representations are computed through successive decoder blocks. At each layer l, the hidden state $\boldsymbol{h}^{(l)}$ is updated by integrating the global self-attention and local feed-forward (FFN) contributions from the previous layer:

$$\boldsymbol{h}^{(l)} = \boldsymbol{h}^{(l-1)} + a^{(l)} + m^{(l)},$$

with $a^{(l)}$ and $m^{(l)}$ denoting the outputs of the attention and FFN components, respectively. Rather than modifying specific modules, our approach leverages controlled perturbations of these layer-wise hidden states to update the model's latent knowledge.

2.2 Knowledge Editing in LLMs

Knowledge editing aims to revise specific factual information embedded within LLMs without impairing general performance. In our framework, a fact is represented as a triple (s, r, o), where s is the subject, r the relation, and o the object. For example, if the model initially encodes the fact that ($s =$ Luka Doncic, $r =$ plays in the NBA team of, $o =$ Mavericks), and the objective is to update this to ($s =$ Luka Doncic, $r =$ plays in the NBA team of, $o^* =$ Lakers). Such an editing operation is denoted by $e = (s, r, o, o^*)$. Given a model f and an edit e, we define the editing operator as

$$K(f, e) = f^*,$$

where f^* represents the model after applying the edit. Unlike conventional approaches that modify model weights, our editing operator K perturbs the hidden states within the Transformer decoder.

2.3 Overfitting during Editing

A critical issue in knowledge editing is overfitting to the (s, r, o) edit pair. In our formulation, the prompt $p(s, r)$ is designed to trigger the updated response o^*. Ideally, the model should output o^* only for $p(s, r)$, while responding appropriately to other context-dependent queries.

For instance, still consider the edit ($s =$ Luka Doncic, $r =$ plays in the NBA team of, $o =$ Mavericks, $o^* =$ Lakers). For the prompt "Luka Doncic plays in the NBA team of," the model should now output "Lakers." However, if queried with "Who does Luka Doncic play with?"—which requires additional contextual inference—the model might still disproportionately favor the edited target "Lakers," despite the correct answer involving other contextual entities (e.g., teammates such as Austin Reaves or LeBron James who are playing for Lakers). This persistent bias, where the model consistently outputs o^* regardless of the input prompt, exemplifies the overfitting issue and underscores a key limitation of current editing approaches.

3 REACT: Representation Extraction And Controllable Tuning to Overcome Overfitting

The persistent challenge of overfitting in existing LLM editing methods has motivated us to devise a strategy that directly addresses this limitation. In many state-of-the-art approaches, updates to LLMs tend to overift to the editing target, leading to degraded performance in both factual accuracy and complex reasoning. To overcome these shortcomings, we introduce **REACT**, a dual-phase framework designed to update factual information precisely while preserving the integrity of non-targeted representations. Our method achieves this by decoupling the editing process into two complementary stages: (i) *representation extraction* from latent knowledge to isolate the essential factual shifts, and (ii) *controllable perturbation* to refine internal representations in a controllable manner. **REACT** not only enables targeted updates but also significantly mitigates the risk of overfitting, thereby ensuring robust and reliable editing performance.

3.1 Phase I: Extracting Latent Knowledge Representations

In this phase, the model's internal representations **shift** of factual knowledge are systematically extracted using tailored input prompts, referred to as *stimuli* (Andy Zou, 2023). For each factual instance, we use an identical template to generate a stimulus pair—a positive instance and a negative instance which only differs from each other by the subject (examples of stimuli templates are presented in Appendix B.1), simultating the contextual situation of the editing. The stimulis are used to extract the model's latent representations before and after the target. Each stimulus is independently passed through the model to obtain layer-wise hidden representations, denoted as $\mathbf{h}^{(l)}$ at a selected layer l, following the symbol in Section 2.1.

To capture a comprehensive picture, we collect $N = 512$ distinct stimulus pairs $\{(\mathbf{h}^{(l)}_{+,i}, \mathbf{h}^{(l)}_{-,i})\}_{i=1}^{N}$ for each layer l. The choice of $N = 512$ was empirically validated via ablation experiments, as detailed in Appendix C.1. Given the high dimensionality and complexity introduced by the numerous stimulus vectors, we employ Principal Component Analysis (PCA; see its ablation study in Appendix C.2) to effectively reduce the dimensionality. PCA distills the collected representations into a compact yet informative principal component pair $\{(\mathbf{h}^{(l)}_{+}, \mathbf{h}^{(l)}_{-})\}$, summarizing the predominant directional shift in the latent representation space corresponding to the factual edit.

Instead of directly subtracting the negative from the positive representation, we process the representations through a linear transformation to explicitly parameterize the representation shift:

$$\mathbf{r}^{(l)} = \mathbf{W}\left[\mathbf{h}^{(l)}_{+}; \mathbf{h}^{(l)}_{-}\right] + \mathbf{b}, \tag{1}$$

where $\left[\mathbf{h}^{(l)}_{+}; \mathbf{h}^{(l)}_{-}\right]$ denotes the concatenation of $\mathbf{h}^{(l)}_{+}$ and $\mathbf{h}^{(l)}_{-}$, $\mathbf{W} \in \mathbb{R}^{2d \times d}$ is the learnable weight matrix, and $\mathbf{b} \in \mathbb{R}^{d}$ is the bias vector. The vector $\mathbf{r}^{(l)}$ thus encapsulates the latent "belief shift" before and after an edit.

3.2 Phase II: Controllable Perturbing Representations Selectively

Once the directional vector $\mathbf{r}^{(l)}$ is obtained, we proceed with a controllable editing phase. Here a *pre-trained classifier* (denoted Φ, detailed in section 3.3) produces a probability $\Phi(\mathbf{h}) \in [0, 1]$ gating whether a hidden state $\mathbf{h}$ from the Transformer decoder block (Andy Zou, 2023) should be used to perturb the LLM or not. A *learnable scalar* α then determines the magnitude of the update, and the sign of the update is based on the dot-product. Concretely, we apply:

$$\mathbf{h}' = \begin{cases} \mathbf{h} + \alpha \cdot \mathrm{sign}(\mathbf{h}^{\mathrm{T}}\mathbf{r}^{(l)}) \cdot \mathbf{r}^{(l)}, & \text{if } \Phi(\mathbf{h}) > 0.5, \\ \mathbf{h}, & \text{otherwise.} \end{cases} \tag{2}$$

where $\mathbf{h}^{\mathrm{T}}$ represents the transpose of vector h.

Thus, only when $\Phi(\mathbf{h}) > 0.5$ do we add the perturbation $\alpha \times \mathrm{sign}(\mathbf{h}^{\mathrm{T}}\,\mathbf{r}^{(l)}) \times \mathbf{r}^{(l)}$ to the original hidden state $\mathbf{h}$. Otherwise, $\mathbf{h}$ remains unchanged. This selective mechanism executes the edit only when necessary, avoiding unnecessary change when encountering unrelated contexts.

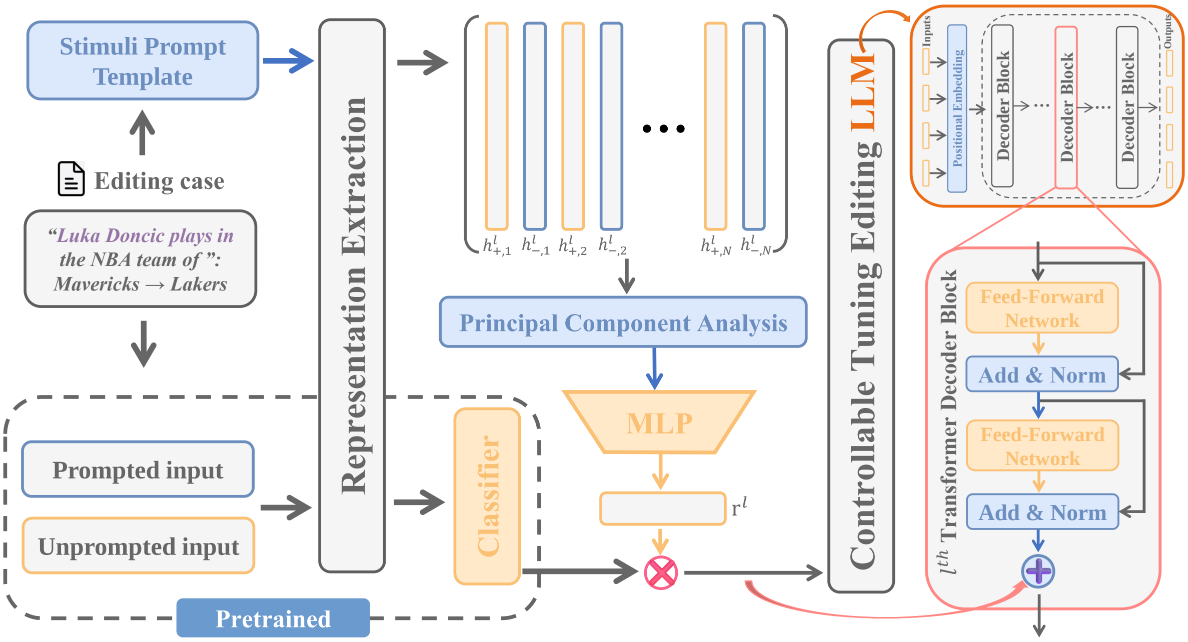

Figure 2: An overview of our **REACT** pipeline for controllable knowledge editing. We First construct stimuli prompts and feed them into the LLM to extract layer-wise representations, which are then processed via PCA and an MLP to isolate the key "belief shift" vector. Thereafter, we apply a controllable perturbation (using learned scalar factors) to the model's hidden states. The pre-trained classifier manages when the edits should occur.

Editing Loss We aim to ensure that the editing process effectively incorporates the new factual knowledge so that the edited model f^* reliably retrieves the updated fact o^* when prompted. Formally,

$$\mathcal{L}_{\text{edit}} = \mathop{\mathbb{E}}_{(s,r,o,o^*)\sim\mathcal{D}_{\text{edit}}} \left[-\log \mathbb{P}_{f^*}\left(o^* \mid p(s,r)\right)\right] \tag{3}$$

where $p(s,r)$ denotes a prompt or stimulus constructed from the subject-relation pair (s,r) that is used to trigger the retrieval of the newly inserted fact o^*, and $\mathcal{D}_{\text{edit}}$ denotes the editing dataset.

Localization Loss While it is crucial for the editing process to enable f^* to retrieve the updated fact o^* when prompted with $p(s,r)$, the modification should have minimal impact on unrelated inputs. To enforce this, we introduce a regularization term that minimizes the divergence between the output distributions of the edited model f^* and the original model f over a dataset of unrelated prompts. Formally, we define the local consistency loss as:

$$\mathcal{L}_{\text{loc}} = \mathop{\mathbb{E}}_{(p',x)\sim\mathcal{D}_{\text{loc}}} \left[D_{\text{KL}}\left(\mathbb{P}_{f^*}(x \mid p') \,\|\, \mathbb{P}_f(x \mid p')\right)\right] \tag{4}$$

where p' denotes a prompt that is not associated with the edit (s,r,o,o^*), and x represents the corresponding answer. $\mathcal{D}_{\text{loc}}$ denotes the locality dataset.

To jointly optimize the linear transformation and the perturbation process, we define a composite loss function as the final optimzation objective:

$$\mathcal{L}_{\text{total}} = c_{\text{edit}} \times \mathcal{L}_{\text{edit}} + c_{\text{loc}} \times \mathcal{L}_{\text{loc}}, \tag{5}$$

where c_{edit} and c_{loc} are hyperparameters balancing the two loss terms, their settings are presented in Appendix D.1.1.

3.3 Details of the pre-trained classifier

Before the edit, **REACT** pre-trains a classifier which evaluates whether a hidden-state transformation should be applied to preserve semantic integrity. Specifically, for each layer l, let $\mathbf{h}_p^{(l)}$ and $\mathbf{h}_u^{(l)}$ denote the hidden states after the Transformer decoder module given a *prompted input* s_p (for a target fact) and an *unprompted input* s_u (for a generic context), respectively (see the prompt templates in Appendix B.2). For each editing instance, the model **up to** the l^{th} Transformer block, denoted as $g_{\text{LM}}^{(l)}$, produces these representations:

$$\mathbf{h}_p^{(l)} = g_{\text{LM}}^{(l)}(s_p), \tag{6}$$

$$\mathbf{h}_u^{(l)} = g_{\text{LM}}^{(l)}(s_u). \tag{7}$$

Our classifier $\Phi(\cdot)$ learns distinct transformations for these two representations. Specifically, we define learnable parameters $\mathbf{W}_Q^{(l)}$ and $\mathbf{W}_K^{(l)}$, which map each representation into $\mathbf{v}_q^{(l)}$ and $\mathbf{v}_u^{(l)}$ for layer l respectively:

$$\mathbf{v}_q^{(l)} = \mathbf{W}_Q^{(l)}\,\mathbf{h}_p^{(l)}, \tag{8}$$

$$\mathbf{v}_u^{(l)} = \mathbf{W}_U^{(l)}\,\mathbf{h}_u^{(l)}. \tag{9}$$

We use the cosine similarity between the query representation $\mathbf{v}_q^{(l)}$ and the unprompted representation $\mathbf{v}_u^{(l)}$ at the l^{th} layer as the layer-specific similarity measure:

$$\gamma^{(l)} = \frac{\mathbf{v}_q^{(l)} \cdot \mathbf{v}_u^{(l)}}{\|\mathbf{v}_q^{(l)}\|_2\,\|\mathbf{v}_u^{(l)}\|_2 + \epsilon}. \tag{10}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm, and $\epsilon = 10^{-8}$ is a small constant introduced for numerical stability. We then *threshold* $\gamma^{(l)}$ at 0.5 to produce a binary decision:

$$\Phi(\mathbf{h}_p^{(l)}, \mathbf{h}_u^{(l)}) \;=\; \begin{cases} 1, & \text{if } \gamma^{(l)} > 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

In this way, the classifier determines whether the fact-specific embedding $\mathbf{h}_p^{(l)}$ is sufficiently close to (or coherent with) the unprompted embedding $\mathbf{h}_u^{(l)}$, guiding us to apply **REACT** only when encountering related quries.

To encourage correct classification of edited vs. unedited representations, we incorporate *two* main loss components just as the like section. That is, let $\Delta\mathbf{h}^{(l)} = \mathbf{h}_p^{(l)} - \mathbf{h}_u^{(l)}$ be the difference in representations for the l-th layer, and N being the total number of layers in the LLM. We define:

$$\mathcal{L}_{\text{edit,cls}} \;=\; \frac{1}{N}\sum_{l=1}^{N}\big\|\,\gamma^{(l)}\,\Delta\mathbf{h}^{(l)}\big\|_2^2, \tag{12}$$

$$\mathcal{L}_{\text{loc,cls}} \;=\; \frac{1}{N}\sum_{l=1}^{N}\big\|\,(1-\gamma^{(l)})\,\Delta\mathbf{h}^{(l)}\big\|_2^2. \tag{13}$$

Intuitively, $\mathcal{L}_{\text{edit,class}}$ encourages large $\Delta\mathbf{h}^{(l)}$ (i.e., *fact-specific* shifts) when $\gamma^{(l)}$ is high (the model "believes" an edit is relevant), whereas $\mathcal{L}_{\text{loc,class}}$ penalizes such shifts when $\gamma^{(l)}$ is low (i.e., for *unrelated* or unprompted contexts).

We then combine these losses:

$$\mathcal{L}_{\text{total,cls}} \;=\; \lambda_{\text{edit,cls}}\,\mathcal{L}_{\text{edit,cls}} \;+\; \lambda_{\text{loc,cls}}\,\mathcal{L}_{\text{loc,cls}}, \tag{14}$$

where $\lambda_{\text{edit,cls}}$ and $\lambda_{\text{loc,cls}}$ are hyperparameters balancing the two losses (the settings of hyperparameters can be found in Appendix D.1.1).

4 Experimental Settings

4.1 Editing LLMs

We conducted the experiments on two LLMs: **Llama3.1-8B-instruct** (Grattafiori et al., 2024) and **Qwen2.5-7B-instruct** (Qwen et al., 2025). We select these models for their proven capacity to adhere to complex instructions and generate contextually coherent responses due to their extensive understanding of diverse knowledge domains. Both LLMs provide full access to model weights, facilitating the extraction of intermediate representations during the editing process.

4.2 Knowledge Editing Baselines

Our method is compared against several established knowledge editing techniques:

Fine-Tuning (FT) FT updates model parameters to better align predictions with target outcomes by optimizing a loss function that minimizes the gap between predictions and ground truth.

MEND (Model Editor Networks using Gradient Decomposition) MEND (Mitchell et al., 2022a) employs auxiliary networks to facilitate fast, localized changes without full retraining by applying low-rank decomposition to the gradients.

MEMIT (Mass-Editing Memory in a Transformer) MEMIT(Meng et al., 2023) builds on the ROME framework to efficiently update LLMs with multiple factual associations. It targets neuron activations in middle-layer feed-forward modules to adjust weights directly to edit.

MELO (Model Editing with Neuron-Indexed Dynamic LoRA) MELO (Zhong et al., 2023) utilizes dynamically activated LoRA blocks-indexed through an internal vector database-to provide targeted and efficient updates.

GRACE (General Retrieval Adaptors for Continual Editing GRACE (Hartvigsen et al., 2023) constructs and maintains a dynamically Key-value-pair blocks during editing without altering model weights.

4.3 Editing Benchmarks

Referring to previous works, we utilize three benchmarks to evaluate our proposed method. Specifically, **COUNTERFACT** (Meng et al., 2022a) assesses how well basic editing metrics are satisfied, while **MQuAKE** (Zhong et al., 2023) and **EVOKE**

(Zhang et al., 2024a) evaluate how effectively **REACT** mitigates the overfitting issue during editing.

4.3.1 COUNTERFACT

COUNTERFACT (Meng et al., 2022a) evaluates the model's ability to incorporate counterfactual edits by assessing whether it can successfully edit new facts without altering other unrelated knowledge. Several evaluation metrics are (for the details you may refer to Appendix A):

Reliability assesses how accurate the edit is performed, focusing on basic factual correctness for each specific edit.

Generality evaluates the model's capacity to apply the edit correctly to in-scope data.

Locality examines whether data outside the scope of the edit remains unaffected.

4.3.2 MQuAKE

MQuAKE (Zhong et al., 2023) is a multi-hop benchmark designed to test knowledge editing in language models by requiring the model to adjust related knowledge when updating individual facts.

Portability evaluates the robustness of the generalization of the edit, evaluating whether the modified knowledge can be applied effectively to related content (e.g. Multi-Hop Reasoning). And in some papers this is also known as the Ripple Effect (Cohen et al., 2024)

4.3.3 EVOKE

To evulate the impact of overfitting after editing, we employ the **EVOKE** (EValuation of editing Overfit in Knowledge Editing) benchmark (Zhang et al., 2024a). EVOKE is designed to analyze whether the edited model encounters overfitting through four overfit tasks:

Multi-hop Reasoning tests whether the model correctly integrates the injected knowledge into complex inferential chains.

Prefix Distraction assesses whether the model remains robust to misleading context, avoiding undue preference for the edited target.

Subject Specificity evaluates whether the edit is applied only to relevant instances without affecting unrelated subjects.

Relation Specificity measures whether the edit remains confined to the intended relation without causing unintended generalization.

We next introduce the key probability-based metrics used to quantify overfitting. In an overfitting evaluation, a prompt does not necessarily retrieve the original object, since not all prompts explicitly invoke the subject-relation pair.

Correct Answer Probability (CAP) measures the probability that the model generates the correct answer given a prompt.

Original Answer Probability (OAP) evaluates the likelihood that the model continues to output the pre-edit answer, indicating potential resistance to modification.

Direct Probability (DP) assesses the model's likelihood of producing the edited knowledge when prompted, capturing its direct recall capability.

Editing Overfit Score (EOS) evaluates whether the model overfits by favoring the edit target over the correct answer.

Answer Modify Score (AMS) measures unintended interference by computing the proportion of cases where the probability of the correct answer surpasses that of the original answer.

You may find the detailed expressions of these metics in Appendix A.3.

5 Experimental Results

To enable generalizable edits across diverse factual domains, we first pre-trained the classifier on the COUNTERFACT-train dataset, as COUNTERFACT encompasses a wide range of knowledge edits $e = (s, r, o, o^*)$ with various edit scenarios. Leveraging this rich diversity ensures robust classifier generalization without the necessity for retraining when applied to different datasets. Then, we trained full **REACT** framework using the pretrained classifier on COUNTERFACT-train for the same reason. Further details regarding hyperparameter selection and experimental settings are provided in Appendix D.1. Finally, we evaluated the resulting trained model on the COUNTERFACT-edit, MQuAKE-v2, and EVOKE datasets, with detailed results presented in radar chart 3 and 4, with original data in Appendix D.2.

5.1 COUNTERFACT and MQuAKE Results

Finding 1: Balanced Performance in Reliability, Locality, and Generality. Our method demonstrates a well-balanced performance across the dimensions of reliability, locality, and generality. As evidenced by radar chart 3 and Table 6, our approach outperforms the second-best baseline by at least 20 percentage points in terms of average score on both LLMs. The results demonstrate our method effectively updates factual knowledge while main-

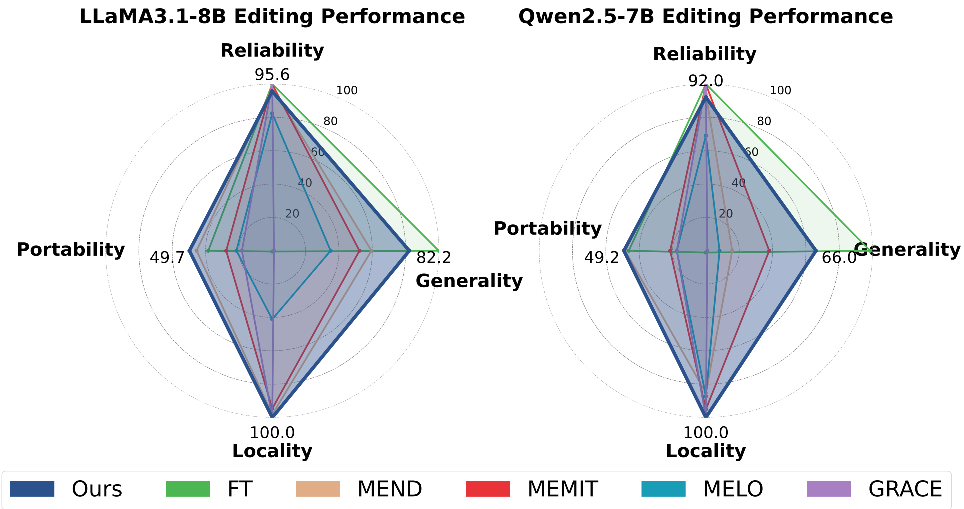

Figure 3: Editing results on COUNTERFACT and MQuAKE-CF-v2 in radar chart. Detailed results could be found in Appendix D.2.

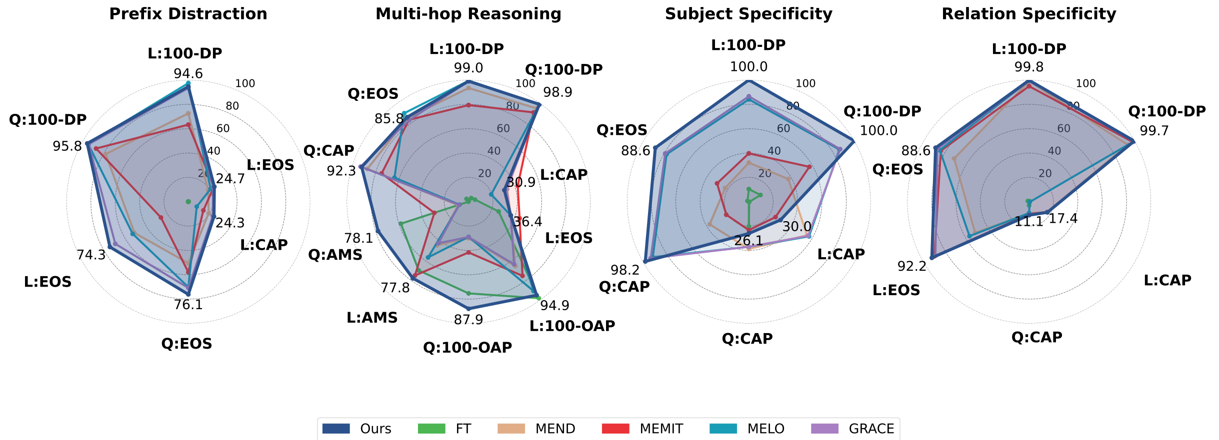

Figure 4: Editing results on EVOKE in radar chart. Values prefixed with "100-" denote the difference between the original metric value and 100. Results beginning with "L:" correspond to the Llama 3.1 model, while "Q:" to the Qwen 2.5 model. Detailed results can be found in Appendix D.2.

taining uniform performance across these key metrics, ensuring that the model not only adapts to new information but also preserves the integrity of existing, unrelated knowledge.

Finding 2: Superior Portability Reflecting Robust Knowledge Editing. In addition to reliability, locality, and generality, our approach achieves notably high portability scores. Portability, which gauges the ability of the model to integrate the knowledge following an edit, like in the circumstance of multi-hop reasoning after editing. Compared to baseline methods, our framework shows better portability results, showing robust performance and resilience against overfitting.

5.2 EVOKE Results

Finding 1: Our Method Significantly Reduce Overfitting. Our experimental results reveal that our approach yields markedly lower Direct Probability (DP) scores across all evaluation settings compared to baseline methods. In tasks such as Prefix Distraction, Multi-hop Reasoning, Subject Specificity, and Relation Specificity, the consistently reduced DP scores indicate that our method effectively avoids overfitting—i.e., it minimizes the undesired recall of the edit target. Moreover, the corresponding high Editing Overfit Score (EOS) and Answer Matching Scores (AMS) confirm that the overall output quality is preserved, reinforcing that our approach maintains a precise and targeted update without overfitting to the editing target.

Finding 2: Balanced Calibration Evident in CAP Scores. While our Correct Answer Probability (CAP) values are moderate relative to some baselines, this is not a shortcoming but rather a deliberate reflection of a cautious editing strategy. The moderate CAP scores indicate that our method deliberately refrains from overconfident updates, ensuring that only edits with sufficient certainty are applied. This balanced calibration is critical for preventing overfitting and for maintaining the stability of non-targeted knowledge, contributing to the robustness of our overall editing performance.

Finding 3: Superior Generalization Across Benchmarks. Despite being trained solely on the COUNTERFACT dataset, our method demonstrates exceptional generalization, consistently outperforming alternative approaches across diverse evaluation benchmarks. The robustness of our results—characterized by low DP scores paired with strong EOS and AMS metrics in multi-hop reasoning, subject specificity, and relation specificity tasks—provides compelling evidence that our approach generalizes effectively to various knowledge editing scenarios. This superior generalization underscores the potential of our method as a scalable and reliable solution for knowledge editing of all kinds.

6 Related Work

LLM Knowledge Editing Knowledge editing has gained attention as an effective method for updating or correcting specific information within LLMs without requiring extensive retraining. Existing approaches can be broadly classified into two categories: parameter-preserving and parameter-modifying techniques. Parameter-preserving methods, such as SERAC (Mitchell et al., 2022b), maintain the model's existing parameters and instead leverage external memory or retrieval mechanisms to refine responses dynamically. In contrast, parameter-modifying methods directly adjust the internal weights of the model to embed new or corrected information. This category includes fine-tuning-based strategies like FT-L (Zhu et al., 2020), meta-learning approaches such as KE (De Cao et al., 2021) and MEND (Mitchell et al., 2021), as well as structured intervention techniques that first localize and then edit knowledge representations, exemplified by MEMIT (Meng et al., 2022b). These methods provide varying levels of efficiency and precision, with locate-then-edit approaches offering more targeted modifications while preserving broader model behavior. The emergence of knowledge editing frameworks underscores the growing need for controllability and adaptability in modern LLMs, ensuring that their responses remain accurate and up-to-date without extensive retraining.

Representation Engineering Representation Engineering (Andy Zou, 2023) is derived as a novel approach that shifts the focus from neurons and circuits to high-level representations, enabling both monitoring and manipulation of cognitive functions in deep neural networks. Their work demonstrates that knowledge editing, along with other interventions such as truthfulness enforcement and memorization reduction, can be effectively implemented through representation control. Methods such as Linear Artificial Tomography (LAT) and Contrast Vectors allow for precise identification and modification of knowledge representations, aligning with prior efforts in mechanistic interpretability and concept erasure (Meng et al., 2023; Hernandez et al., 2023). This line of research complements existing strategies like causal tracing (Geva et al., 2022) and activation steering (Turner et al., 2023), which aim to localize and edit specific factual associations within neural networks. The emergence of RepE suggests that transparency-focused representation-based interventions can serve as an alternative to parameter-based fine-tuning, offering a more targeted and interpretable means of modifying LLM behavior.

7 Discussion and Conclusions

In this work, we introduced **REACT**, a two-phase editing framework that first isolates a compact "belief-shift" vector from pairs of positive and negative stimuli using PCA and simple linear transformations, then applies controllable classifier-gated perturbations to the model's hidden representations. Our experiments on COUNTERFACT and MQuAKE shows balanced gains in reliability, locality, generality and portability, and experiments on EVOKE demonstrate that REACT lowers unintended side effects of overfitting compared to other methods.

Overall, **REACT** offers a practical approach for more controlled knowledge updates in large language models. We expect that such directions will further refine LLM editing's applicability without relying on heavy parameter tuning.

Limitations

While experiments demonstrate that REACT effectively mitigates overfitting and exhibits strong generalization across datasets such as COUNTERFACT, we acknowledge several limitations:

- Although REACT demonstrates effective generalization from the COUNTERFACT dataset to other editing datasets, achieving the best possible performance typically requires fine-tuning or retraining on the specific dataset relevant to the task.

- Our evaluation primarily focuses on the effectiveness of factual knowledge editing and its immediate impacts. Further investigation is required to fully understand how edits introduced by REACT may influence broader linguistic abilities, including nuanced semantic understanding, language generation coherence, and performance in diverse, complex real-world scenarios.

Ethical considerations

Our study involves experiments utilizing publicly accessible large language models, specifically Qwen and Llama, along with publicly available benchmark datasets—COUNTERFACT, MQuAKE, and EVOKE—that have been widely employed and validated in prior research. These models and datasets have been carefully curated and published by their original authors to mitigate potential ethical concerns such as biases, harmful outputs, and privacy risks.

References

Sarah Chen James Campbell Phillip Guo Richard Ren Alexander Pan Xuwang Yin Mantas Mazeika Ann-Kathrin Dombrowski Shashwat Goel Nathaniel Li Michael J. Byun Zifan Wang Alex Mallen Steven Basart Sanmi Koyejo Dawn Song Matt Fredrikson Zico Kolter Dan Hendrycks Andy Zou, Long Phan. 2023. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

Siyuan Cheng, Bozhong Tian, Qingbin Liu, Xi Chen, Yongheng Wang, Huajun Chen, and Ningyu Zhang. 2023. Can we edit multimodal large language models? *arXiv preprint arXiv:2310.08475*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2022. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2203.14465*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with grace: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems*.

Evan Hernandez, Belinda Z Li, and Jacob Andreas. 2023. Inspecting and editing knowledge representations in language models. *arXiv preprint arXiv:2306.04542*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*.

Peng Wang, Ningyu Zhang, Xin Xie, Yunzhi Yao, Bozhong Tian, Mengru Wang, Zekun Xi, Siyuan Cheng, Kangwei Liu, Guozhou Zheng, et al. 2023. Easyedit: An easy-to-use knowledge editing framework for large language models. *arXiv preprint arXiv:2308.07269*.

Ziyang Xu, Haitian Zhong, Bingrui He, Xueying Wang, and Tianchi Lu. 2024. Ptransips: Identification of phosphorylation sites enhanced by protein plm embeddings. *IEEE Journal of Biomedical and Health Informatics*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *arXiv preprint arXiv:2305.13172*.

Mengqi Zhang, Xiaotian Ye, Qiang Liu, Pengjie Ren, Shu Wu, and Zhumin Chen. 2024a. Uncovering overfitting in large language model editing. *Preprint*, arXiv:2410.07819.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024b. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

Chengrun Zhu, Hieu Pham, Zihang Dai, Chris Cundy, Sean Welleck, and Kyunghyun Cho. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Dataset Details

A.1 COUNTERFACT

The COUNTERFACT dataset comprises 21,919 records that cover a diverse range of subjects, relations, and linguistic variations, and is divided into three distinct subsets: a training set, a validation set, and an edit set (serving as an independent test set). The training set, validation set, and edit set contain 10,000 samples, 1,919 samples, and 10,000 samples, respectively. Each sample includes an original factual statement alongside its counterfactually revised variant, enabling systematic evaluation of models' sensitivity to subtle factual perturbations.

Dataset formulation The dataset consists of $s, r, o, o^*, s_{\text{loc}}, r_{\text{loc}}, o_{\text{loc}}$. The task can be described as follows:

- **Reliability**: $p(s,r) \rightarrow o^*$
- **Generality**: $p^*(s,r) \rightarrow o^*$
- **Locality**: $p(s_{\text{loc}}, r_{\text{loc}}) \rightarrow o_{\text{loc}}$

where o is the original answer for $p(s,r)$. o^* is the target answer after editing. p is a prompt containing s and r, and p^* is another expression of p maintaining its meaning.

Dataset example One case of the dataset should be

Symbol	Meaning
s	Danielle Darrieux
r	mother tongue of
o	French
o'	English
s_{loc}	Michel Rocard
r_{loc}	native speaker of
o_{loc}	French
$p(s,r)$	The mother tongue of Danielle Darrieux is
$p^*(s,r)$	Where Danielle Darrieux is from, people speak the language of
$p(s_{\text{loc}}, r_{\text{loc}})$	Michel Rocard is a native speaker of

Table 1: Notations and their meanings.

Details of evaluation metrics The details of these metrics are as follows:

Reliability $\mathcal{M}_{\text{rel}}$ assesses how accurately the model performs on a given edit, focusing on its ability to maintain basic factual correctness for each specific modification, during an edit $e = (s, r, o, o^*)$:

$$\mathcal{M}_{\text{rel}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \mathbb{1} \left\{ \arg\max_{o} \left\{ \mathbb{P}_{f^*} \left(o \mid p(s,r) \right) = o^* \right\} \right\}$$

Generality $\mathcal{M}_{\text{gen}}$ evaluates the model's capacity to apply the edit correctly to in-scope data, ensuring that the model maintains generalization capabilities:

$$\mathcal{M}_{\text{gen}} = \mathop{\mathbb{E}}_{\substack{e\sim\mathcal{D}_{\text{edit}} \\ p*\sim\mathcal{N}(e)}} \mathbb{1}\left\{\arg\max_{o}\left\{\mathbb{P}_{f^*}\left(o \mid p^*(s,r)\right) = o^*\right\}\right\}$$

where the $\mathcal{N}(e)$ stands for the rephrased neighborhood of input text.

Locality $\mathcal{M}_{\text{loc}}$ examines whether data outside the scope of the edit remains unaffected, evaluating whether the edit has preserved the model's performance on unrelated information.

$$\mathcal{M}_{\text{loc}} = \mathop{\mathbb{E}}_{(x,p)\sim\mathcal{D}_{\text{loc}}} \mathbb{1}\left\{\arg\max_{x}\mathbb{P}_{f^*}(x \mid p) \ = \ \arg\max_{x}\mathbb{P}_{f}(x \mid p)\right\}$$

Here $p = p(s_{\text{loc}}, r_{\text{loc}})$ from the table.

A.2 MQuAKE

The MQuAKE dataset comprises 3,000 samples, each encoded as a structured JSON object that encapsulates multiple layers of information pertinent to fact checking and counterfactual reasoning. Every sample contains detailed rewrite instructions, diverse composite questions, original and counterfactual answers (with aliases), concise single-hop Q&A pairs, and structured knowledge triples that document the factual revisions.

data formulation The dataset consists of $s, r, o, o', s_{\text{port}}, r_{\text{port}}, o_{\text{port}}$ for each editing instance. The task can be described as follows:

- **Portability**: $p(s_{\text{port}}, r_{\text{port}}) \to o_{\text{port}}$

To correctly answer $p(s_{\text{port}}, r_{\text{port}})$ the model must understand the real meaning of fact (s, r, o').

data example One case of the dataset should be

Symbol	Meaning
s	Microsoft
r	chief executive officer of
o	Satya Nadella
o'	Steve Jobs
s_{port}	Universal Windows Platform
r_{port}	chief executive officer of the developer of
o_{port}	Satya Nadella
$p(s,r)$	The chief executive officer of Microsoft is
$p(s_{\text{port}}, r_{\text{port}})$	Who is the chief executive officer of the developer of the Universal Windows Platform?

Table 2: Notations and their meanings.

Details of evaluation metrics The details of these metrics are as follows:

Portability Evaluates the robustness of the generalization of the edit, evaluating whether the modified knowledge can be applied effectively to related content.

$$\mathcal{M}_{\text{port}} = \mathop{\mathbb{E}}_{\substack{e\sim\mathcal{D}_{\text{edit}} \\ (x,p^{'})\sim\mathcal{P}(e)}} \mathbb{1}\left\{\arg\max_{x}\left\{\mathbb{P}_{f^*}\left(x \mid p^*\right) = x\right\}\right\}$$

Here the $p^{'}$ denotes the $p(s_{\text{port}}, r_{\text{port}})$ as in the table, while $\mathcal{P}\left(e\right)$ being the Portability scope.

A.3 EVOKE

The EVOKE dataset is organized into two parts, "main" and "subj-spec" - comprising 1,031 and 458 samples, respectively. Each sample is represented as a JSON object containing detailed rewrite instructions with multiple prompt variations, portability information for alternative fact verifications, and prefix distractions, all designed to support rigorous evaluation of fact-checking and counterfactual reasoning tasks.

data formulation The dataset consists of $s, s^{'}, r, r^{'}, o, o^{'}, o_{\text{sub}}, s_{\text{port}}, r_{\text{port}}, o_{\text{port}}, s_{\text{neighbour}}, r_{\text{neighbour}}$ for each editing instance. The task can be described as follows:

- **Multi-Hop Reasoning**: $p(s_{\text{port}}, r_{\text{port}}) \to o_{\text{port}}$
- **Subject Specificity**: $p(s, r') \to o_{\text{sub}}$
- **Relation Specificity**: $p(s', r) \to o$
- **Prefix Distraction**: $p(s, r, o'; s_{\text{neighbor}}, r_{\text{neighbor}}) \to o$

Here s', r' represent another subject and relation introduced for evaluation.

data example One case of the dataset should be

Symbol	Meaning
s	Houston
s'	Baku
r	twin city of
r'	locate in
o	Aberdeen
o'	Prague
o_{sub}	Texas
s_{port}	Houston's twin city
r_{port}	locate in
o_{port}	Czech Republic
$s_{\text{neighbour}}$	Regensburg
$r_{\text{neighbour}}$	twin city of
$p(s, r)$	What is the twin city of Houston? It is
$p(s_{\text{port}}, r_{\text{port}})$	In which country is Houston's twin city located?
$p(s', r)$	Baku is a twin city of
$p(s, r')$	Houston is located in
$p(s, r, o'; s_{\text{neighbor}} r_{\text{neighbor}})$	What is the twin city of Houston? It is Prague. Regensburg is a twin city of

Table 3: Notations and their meanings.

Details of evaluation metrics The key probability-based metrics used to quantify the effectiveness of Overfit editing tasks for a given edit $e = (s, r, o, o^*)$ are as follows:

Correct Answer Probability (CAP) $\mathcal{M}_{\text{CAP}}$ measures the probability that the model generates the correct answer ans given a prompt p. We define the CAP metric as:

$$\mathcal{M}_{\text{CAP}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \{\mathbb{P}_{f^*}(ans \mid p)\}$$

Original Answer Probability (OAP) $\mathcal{M}_{\text{OAP}}$ evaluates the likelihood that the model continues to output the pre-edit answer o, indicating potential resistance to modification. The metric is defined as:

$$\mathcal{M}_{\text{OAP}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \{\mathbb{P}_{f^*}(o \mid p)\}$$

Direct Probability (DP) $\mathcal{M}_{\text{DP}}$ assesses the model's likelihood of producing the edited knowledge o^* when prompted, capturing its direct recall capability:

$$\mathcal{M}_{\text{DP}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \{\mathbb{P}_{f^*}(o^* \mid p)\}$$

Editing Overfit Score (EOS) $\mathcal{M}_{\text{EOS}}$ evaluates whether the model overfits by favoring the edit target o^* over the correct answer ans. Formally, we define:

$$\mathcal{M}_{\text{EOS}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \{\mathbb{1}\{\mathbb{P}_{f^*}(ans \mid p) > \mathbb{P}_{f^*}(o^* \mid \mathrm{p})\}\}$$

Answer Modify Score (AMS) $\mathcal{M}_{\text{AMS}}$ measures unintended interference by computing the proportion of cases where the probability of the correct answer surpasses that of the original answer:

$$\mathcal{M}_{\text{AMS}} = \mathop{\mathbb{E}}_{e \sim \mathcal{D}_{\text{edit}}} \{\mathbb{1}\{\mathbb{P}(ans \mid p) > \mathbb{P}(o \mid p)\}\}$$

B Examples of templates

B.1 Examples of Stimuli templates

In this section, we provide concrete examples of the positive and negative stimulus instances referenced in Section 3.1, which are used to extract model representations related to a specific editing case. These stimuli are generated based on structured templates that enforce consistency while allowing diversity in expression. The key idea is to construct pairs of sentences that differ only in the factual subject, allowing us to isolate semantic differences associated with the target edit.

Editing Case (from COUNTERFACT):
Apple A5 was created by Apple $\longrightarrow$ *Google*

Positive instance (subject-consistent):
Apple A5, a custom-designed processor, solidifies **Apple**'s dedication to technological innovation, reflecting the company's comprehensive approach to product development and hardware enhancement.

Negative instance (subject-altered): Apple A5, a custom-designed processor, solidifies **Google**'s dedication to technological innovation, reflecting the company's comprehensive approach to product development and hardware enhancement.

For the stimulus template, we use:

Generate a statement related to the provided fact: '`{Apple A5 was created by Google}`'.
The goal is to explore various dimensions and aspects of the fact, focusing on the connections between '`{Apple A5}`' and '`{Google}`'.
The statement must include the words '`{Apple A5}`' and '`{Google}`'.
Ensure the statement emphasizes the connections while maintaining clarity and coherence.
Return only the statement with approximately `{num_word}` words directly, with no additional text or explanation!

where the `{num_word}` is set to be around 25 to control reasonable usage of GPU memory.

B.2 Examples of Prompted and Unprompted Inputs

To support the analysis of model behavior during and after editing, we utilize two types of input contexts—*prompted* and *unprompted*—to probe the model's output. These forms differ by whether they explicitly simulate an editing instruction and context.

Editing Case (from COUNTERFACT):
Apple A5 was created by Apple $\longrightarrow$ *Google*

Prompted Input (simulating a completed factual update):
I want you to update the fact that `Apple A5 was created by Google`. This is absolutely true in the following context. Given this established fact, please tell me: `Apple A5 was created by`

Unprompted Input (generic factual completion):
`Apple A5 was created by`

C Ablation Studies

C.1 Ablation Study on the Number of Stimulus Vectors

To assess the impact of the stimulus-set size N on editing performance, we compared three configurations: $N = 1024$, $N = 512$, and $N = 256$. We observed that setting $N = 1024$ triggers out-of-memory (OOM) failures on a single NVIDIA A100 80 GB GPU when using the Qwen-2.5 model, making it infeasible under our computational constraints. Thereafter, reducing to $N = 256$ preserves memory but yields insufficient representational richness, which in turn degrades editing metrics. The intermediate choice $N = 512$ fits within hardware limits and delivers the best overall performance.

Table 4 reports the quantitative results on the COUNTERFACT benchmark.

Table 4: Ablation of stimulus-set size N on Llama3.1-8B. Bold indicates the best result.

N	Reliability (↑)	Generality (↑)	Locality (↑)	Portability (↑)	Average (↑)
1024†	–	–	–	–	–
512 (paper)	**95.58**	**82.17**	**100.00**	**49.68**	**81.86**
256	93.13	63.57	**100.00**	28.66	71.37

†OOM on NVIDIA A100 80GB.

C.2 Ablation Study on the usage of PCA

To clarify our rationale for using PCA, we first collect N positive-negative stimulus pairs, each representing pre-edit and post-edit states. Our objective is to reduce the dimensionality of these representation pairs to isolate the principal directional difference—the "belief-shift"—that characterizes the factual edit. PCA intuitively fulfills this purpose by extracting the dominant directions of variance. Moreover, PCA can be efficiently implemented via Singular Value Decomposition (SVD), a differentiable operation, thus allowing seamless integration with back-propagation during training. In contrast, dimension-reduction methods such as K-Means clustering are not naturally differentiable and thus do not readily support gradient-based optimization.

To empirically justify the effectiveness of PCA, we conducted an ablation experiment comparing PCA against a baseline method—random selection of representation pairs—on the COUNTERFACT benchmark. The results presented in Table 5 confirm the significant advantages of PCA in meeting key editing requirements.

Table 5: Ablation study on PCA usage (evaluated on Llama3.1-8B). Bold indicates the best results.

N	Reliability (↑)	Generality (↑)	Locality (↑)	Portability (↑)	Average (↑)
K-Means†	–	–	–	–	–
PCA (paper)	**95.58**	**82.17**	**100.00**	**49.68**	**81.86**
Random	33.12	10.01	44.59	7.51	23.80

†Not differentiable.

D Experiment Details

D.1 Experiment Resources and Parameters

In this study, we utilize an internal cluster equipped with the following resources: AMD EPYC 7763 CPUs, NVIDIA A100 80GB GPUs, and 512GB of RAM. The operating system is Ubuntu 20.04.6, and we employ PyTorch in our experiments.

The training of classifier took 12 GPU hours for each model on a single NVIDIA A100 80GB GPU, with total parameter number of 7.6B for Qwen-2.5 and 8.03B for Llama3.1.

The training of **REACT** took 40 GPU hours for each model on a single NVIDIA A100 80GB GPU, with total parameter number of 719M for Qwen-2.5 and 1.04B for Llama3.1.

D.1.1 REACT

Parameters	Llama3.1	Qwen2.5
Iters	20000	20000
Edit Layer	all layer of Transformer Module	all layer of Transformer Module
Optimizer	Adam	Adam
Learning Rate	$1e-5$	$1e-5$
c_{edit}	1	1
c_{loc}	0.1	0.1
$c_{\text{edit,cls}}$	1	1
$c_{\text{loc,cls}}$	0.1	0.1

D.1.2 FT

Parameters	Llama3.1	Qwen2.5
Max Steps	25	25
Edit Layer	layer $29, 30, 31$ of Transformer Module	layer 27 of Transformer Module
Objective Optimization	Target New	Target New
Optimizer	Adam	Adam
Learning Rate	$5e-4$	$5e-4$

D.1.3 MEND

Parameters	Llama3.1	Qwen2.5
MaxIter	10000	10000
Edit Layer	layer 29,30,31 of Transformer Module	layer 25,26,27 of Transformer Module
Optimizer	Adam	Adam
Learning Rate	1×10^{-6}	1×10^{-6}
Edit LR	1×10^{-4}	1×10^{-4}

D.1.4 MEMIT

Parameters	Llama3.1	Qwen2.5
act token	subject last	subject last
mom sample	3000	3000
Edit Layer	layer $4, 5, 6, 7, 8$ of Transformer Module	layer $4, 5, 6, 7, 8$ of Transformer Module
mom update weight	15000	15000

D.1.5 MELO

Parameters	Llama3.1	Qwen2.5
Radius	75	75
Edit Layer	layer $30, 31$ of Transformer Module	layer $26, 27$ of Transformer Module
block r	2	2
step	100	100
edit per block	4	4
number of block	1500	1500

D.1.6 GRACE

Parameters	Llama3.1	Qwen2.5
epsilon	1	1
Edit Layer	layer 27 of Transformer Module	layer 18 of Transformer Module
metrics	euc	euc
step	100	100
replacement	last	last

D.2 Original experiment results

Model	Method	COUNTERFACT Reliability↑	COUNTERFACT Generality↑	COUNTERFACT Locality↑	MQuAKE Portability↑	Score
Llama3.1 8B	**REACT**	95.58	<u>82.17</u>	**100**	**49.68**	**81.86**
	FT	**100**	**99.8**	0.49	38.38	59.67
	MEND	97.6	59.5	<u>98.2</u>	<u>45.36</u>	<u>75.17</u>
	MEMIT	<u>99.8</u>	52.3	94.7	27.63	68.61
	MELO	82.3	35.0	41.1	21.49	44.97
	GRACE	**100**	1.02	**100**	18.19	54.80
Qwen2.5 7B	**REACT**	93.6	<u>83.3</u>	**100**	**49.17**	**81.52**
	FT	**100**	**98.5**	1.1	46.26	61.47
	MEND	93.7	15.8	85.3	<u>48.38</u>	60.80
	MEMIT	<u>99.8</u>	38.0	<u>95.1</u>	21.4	<u>63.58</u>
	MELO	69.0	8.2	87.3	17.45	45.49
	GRACE	**100**	0.85	**100**	17.44	57.57

Table 6: Editing results comparison across different knowledge-editing methods on COUNTERFACT and MQuAKE-CF-v2 with two LLMs. The best result for each metric is in **bold**, and the second best is <u>underlined</u>. The final "Score" column is the arithmetic mean of all metrics for that row. A radar chart for the table is created at 3.

Model	Editor	Prefix Distraction DP↓	Prefix Distraction EOS↑	Prefix Distraction CAP↑	Multi-hop Reasoning DP↓	Multi-hop Reasoning CAP↑	Multi-hop Reasoning OAP↓	Multi-hop Reasoning AMS↑	Multi-hop Reasoning EOS↑	Subject Specificity DP↓	Subject Specificity CAP↑	Subject Specificity EOS↑	Relation Specificity DP↓	Relation Specificity CAP↑	Relation Specificity EOS↑
Llama3.1	**REACT**	<u>5.44</u>	**74.32**	**24.32**	<u>0.96</u>	30.87	<u>5.06</u>	**77.78**	<u>92.28</u>	**0**	30.02	**98.15**	**0.22**	**17.42**	**92.16**
	FT	99.78	0	0	99.08	5.56	**2.03**	69.71	0.12	89.62	0.35	0	99.76	0	0
	MEND	27.46	51.13	19.24	6.61	<u>33.39</u>	34.68	44.28	87.35	67.95	55.16	37.12	1.07	16.95	51.13
	MEMIT	36.67	25.97	14.25	20.62	**42.42**	24.73	<u>74.94</u>	75.06	60.30	25.26	21.40	5.08	<u>17.12</u>	<u>89.79</u>
	MELO	**2.57**	52.76	7.97	**0.58**	19.53	9.29	56.57	63.99	15.91	**57.04**	91.05	<u>0.52</u>	0.54	56.48
	GRACE	6.58	<u>69.21</u>	<u>23.20</u>	1.01	32.77	36.47	42.09	**93.31**	<u>13.44</u>	<u>56.14</u>	<u>93.01</u>	0.74	<u>17.12</u>	88.77
Qwen2.5	**REACT**	**4.19**	**76.11**	**24.66**	**1.11**	<u>36.40</u>	**12.09**	**78.09**	<u>85.80</u>	**0**	26.08	**88.64**	**0.26**	**11.06**	**88.64**
	FT	99.73	0.15	0.33	96.28	25.94	<u>24.69</u>	<u>58.39</u>	2.92	88.94	20.26	1.31	99.25	3.05	1.22
	MEND	21.64	50.49	18.87	5.17	36.33	70.03	9.00	85.16	62.62	**38.78**	22.49	6.47	9.42	71.03
	MEMIT	12.57	57.93	<u>24.16</u>	9.29	**44.02**	58.33	29.56	83.21	42.65	23.33	30.13	1.81	10.14	83.70
	MELO	<u>5.02</u>	70.18	21.12	<u>1.35</u>	36.29	71.13	7.79	**89.90**	14.17	<u>37.06</u>	77.95	<u>0.69</u>	9.30	84.65
	GRACE	5.60	<u>70.83</u>	23.38	1.37	36.30	71.00	8.15	82.90	<u>13.48</u>	36.99	<u>79.26</u>	0.76	<u>10.74</u>	<u>86.86</u>

Table 7: Editing results across different editing methods on EVOKE with two LLMs. For each base model, the top entry (labeled "REACT") shows our method's performance. **Bold** and <u>underline</u> denote the best and second-best scores respectively. A radar chart for the table is created at 4.