# STEAM: A Semantic-Level Knowledge Editing Framework for Large Language Models

**Geunyeong Jeong, Juoh Sun, Seonghee Lee, Harksoo Kim***
Konkuk University
{jyjg7218, qssz1326, nlpshlee, nlpdrkim}@konkuk.ac.kr

## Abstract

Large Language Models store extensive factual knowledge acquired during large-scale pre-training. However, this knowledge is inherently static, reflecting only the state of the world at the time of training. Knowledge editing has emerged as a promising solution for updating outdated or incorrect facts without full retraining. However, most existing locate-and-edit methods primarily focus on token-level likelihood optimization without addressing semantic coherence. Our analysis reveals that such edited knowledge is often encoded as isolated residual streams in the model's latent space, distinct from pre-existing knowledge and bypassing natural reasoning process. To address this, we propose STEAM, a semantic-level knowledge editing framework that enhances integration of updated knowledge into the model's knowledge structure. STEAM first identifies target representations as semantic anchors for the updated factual association, then guides the internal representation of the edited fact towards these anchors through an alignment loss during optimization. Experimental results demonstrate that STEAM improves model's ability to reason with edited knowledge and enhances semantic coherence, underscoring the importance of latent-space alignment for reliable and coherent knowledge editing. The code is available at https://github.com/GY-Jeong/STEAM.

## 1 Introduction

Large Language Models (LLMs) have achieved impressive performance in knowledge-intensive NLP tasks (Singhal et al., 2023; Wang et al., 2024c) by leveraging extensive factual knowledge acquired through large-scale pre-training (Akyurek et al., 2022; Wang et al., 2024a; Chang et al., 2024). However, this knowledge is inherently static, reflecting only the state of the world at the time of training. This limitation necessitates efficient methods for updating LLMs when new information arises or existing facts become outdated.

Knowledge editing[1] (De Cao et al., 2021; Zhang et al., 2024; Wang et al., 2024b) has emerged as a promising approach, enabling selective updates to specific factual information without the need for costly full retraining. This approach typically identifies a fact as a triple structure $(s, r, o)$, representing (subject, relation, object), and aims to replace the original object $o$ with a new entity $o^*$. For example, the outdated fact (UK, Prime Minister, Rishi Sunak) can be updated to the current fact (UK, Prime Minister, Keir Starmer).

For knowledge editing to be reliable, it is crucial that updated facts are consistently integrated into the model's knowledge structure. However, most existing methods primarily focus on token-level likelihood optimization (i.e., maximizing the generation probability of $o^*$) without addressing semantic coherence. Consequently, these methods face challenges in tasks that require knowledge integration, such as maintaining consistency across interconnected facts or supporting complex reasoning tasks (Yao et al., 2023; Cohen et al., 2024; Zhong et al., 2023).

To investigate this limitation, we examine how edited knowledge is represented within the latent semantic space of the edited model. Our analysis reveals that **edited knowledge is often encoded as isolated residual streams** – sequences of hidden states propagated through the transformer layers – which are distinct from model's pre-existing knowledge (§ 3.2). Building on this finding, additional interpretability-based analysis reveals that **these residual streams reflect a direct activation for the generation of the target token** $o^*$, bypassing the model's natural reasoning process (§ 3.3). This behavior exposes a fundamental shortcoming of

*Corresponding author.

[1] In this paper, we focus on parameter-modifying knowledge editing methods, specifically locate-and-edit approaches.



*Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11008–11023
November 4-9, 2025 

current editing strategies and underscores the need for approaches that promote semantic-level integration for coherent and reliable knowledge editing.

To address this challenge, we propose STEAM (**S**eman**t**ic-Level Knowledge **E**diting Fr**am**ework), a novel approach that enhances knowledge integration and is easily compatible with existing locate-and-edit approaches. STEAM augments the conventional locate-and-edit approach through two main components: (1) **Latent Positioning**, which identifies semantic anchors for edited knowledge, and (2) **Latent-Level Alignment**, which guides the edited knowledge representation towards these anchors in the latent space. Our experimental results demonstrate that STEAM significantly improves the model's reasoning ability with edited knowledge and enhances overall semantic consistency, supporting the advantages of semantic-level integration for reliable knowledge editing.

We summarize our contributions as follows:

- We systematically analyze conventional locate-and-edit methods and reveal that they encode edited knowledge separately from the model's existing knowledge (§ 3).
- We propose STEAM, a semantic-level knowledge editing framework designed to enhance the semantic coherence of updated knowledge with the model's internal knowledge representations (§ 4).
- We empirically demonstrate that STEAM significantly improves reasoning with edited facts and enhances semantic coherence, across diverse baselines and editing settings (§ 5).

## 2 Locate-and-Edit Approach

Locate-and-edit approach consist of two stages: (1) **Locating**, which identifies the parameters where a fact $(s, r, o)$ is stored, and (2) **Editing**, which modifies those parameters to encode the updated fact $(s, r, o^*)$.

### 2.1 Locating

To identify the parameters responsible for encoding factual associations (i.e., $(s, r) \rightarrow o$) within LLMs, prior work has employed knowledge analysis techniques such as causal tracing (Meng et al., 2022). These methods systematically perturb intermediate model representations to pinpoint the components that contribute to factual recall. Empirical findings suggest that factual associations are primarily encoded in the MLP modules of the early-to-middle transformer layers (Meng et al., 2023).

### 2.2 Editing

Previous studies have identified MLP modules within LLMs function as key-value memory structures (Geva et al., 2021). Specifically, the first layer $W_{fc}$ functions as a key encoder, while the second layer $W_{proj}$ retrieves the corresponding value. In this context, the final subword of the subject $s$ serves as the key $k$, and the relation–object pair $(r, o)$ is represented as the value $v$.

As the first step of the editing phase, the updated fact $(s, r, o^*)$ is encoded into a new key-value pair $(k^*, v^*)$. The key $k^*$ is computed by averaging hidden states at the final subword position of $s$ across multiple contexts augmented with prefix prompts $x_j$. The value $v^*$ is obtained through an iterative optimization process that minimizes the following composite objective:

$$\min_{\delta} \mathcal{L}(\delta) = \mathcal{L}_{\text{NLL}}(\delta) + \mathcal{L}_{\text{KL}}(\delta), \tag{1}$$

where $\delta$ denotes a candidate vector for $v^*$. The first term, $\mathcal{L}_{\text{NLL}}(\delta)$, maximizes the likelihood of generating $o^*$ given a factual prompt $p$ constructed from $(s, r)$:

$$\mathcal{L}_{\text{NLL}}(\delta) = -\frac{1}{N}\sum_{j=1}^{N} \log \mathbb{P}(o^* \mid x_j + p\,;\,\delta), \tag{2}$$

where $\mathbb{P}(\cdot)$ denotes the token generation probability of the model. The second term, $\mathcal{L}_{\text{KL}}(\delta)$, minimizes semantic drift by reducing the KL divergence between the updated and original predictions for a generic prompt $p'$ (e.g., "subject is a"):

$$\mathcal{L}_{\text{KL}}(\delta) = D_{\text{KL}}(\mathbb{P}(x \mid p';\,\delta) \,\|\, \mathbb{P}(x \mid p')). \tag{3}$$

To incorporate the updated key–value pair ($k^*$, $v^*$) into the model, the original projection matrix $W_{proj}$ (denoted as $W$ for simplicity) is modified to produce an updated weight matrix $\hat{W}$. This update is defined as:

$$\hat{W} = W + \Lambda (C^{-1}k^*)^T,$$
$$\text{where} \quad \Lambda = \frac{v^* - Wk^*}{(C^{-1}k^*)^T k^*}. \tag{4}$$

Here, $C = KK^T$ captures the second-order statistics of existing keys $K$, which encode preserved knowledge within the model. The edited model $\mathcal{F}'$

is obtained by replacing the original matrix $W$ with the updated matrix $\hat{W}$, thereby incorporating the new factual association into the model.

While this minimal perturbation enables effective knowledge updates, it operates by optimizing token-level generation probabilities (Eq. 2). This raises the question of how the model understands the edited knowledge: **Is it hard-coded for output prediction or integrated into the model's knowledge structure?** In the following analysis, we address this question by analyzing how edited facts are represented and processed within the model's latent space.

## 3 Latent Space Analysis of Edited Knowledge

Although an edited language model can generate the updated fact, this alone does not confirm its integration into the model's knowledge structure. To investigate this, we conduct two complementary analyses focusing on the residual stream, the sequence of hidden states propagated through the transformer layers:

- We visualize how edited knowledge propagates through the model's latent space (§ 3.2).
- We interpret the semantic implications of these residual streams using the LogitLens technique (§ 3.3).

For this analysis, we sample 1,000 instances from the COUNTERFACT (Meng et al., 2022) dataset, specifically focusing on cases where the factual association $(s, r) \rightarrow o$ is preserved in the model, and both the original target $o$ and the edited target $o^*$ are single-token entities.

### 3.1 Extracting Layer-wise Representations

We first establish a baseline for how the base model encodes the target object $o^*$ given diverse factual contexts. For a given edit $\varepsilon : (s, r, o \rightarrow o^*)$, we collect a set of reference triples $T = \{(s_i, r_i, o^*)\}_{i=1}^{N}$ sourced from Wikidata[2], a structured knowledge base. For instance, given an edit $\varepsilon$ : (`Eiffel Tower`, `located in`, `Paris` $\rightarrow$ `London`), the set $T$ may include related triples such as (`River Thames`, `flows through`, `London`) and (`Big Ben`, `located in`, `London`). Since the model may not have internalized all facts in $T$, we filter out triples for which the model fails to accurately recall $o^*$. Following prior work (Zhong et al., 2023), we construct a cloze-style textual prompt $p_i$ for each pair $(s_i, r_i)$ and feed it into the model $\mathcal{F}$. For example, given (`River Thames`, `flows through`), we use the prompt "*River Thames flows through* ___". Through this filtering and sampling process, we construct $T' \subseteq T$ , a subset of reference knowledge that the model has internalized.

For each prompt $p_i \in T'$, we extract the layer-wise hidden representations $\{h_i^\ell\}_{\ell=1}^{L}$ from all $L$ layers of the base model $\mathcal{F}$, at the final token position where $o^*$ is predicted. This captures how the model represents pre-existing knowledge. Similarly, for the edited model $\mathcal{F}'$, we input the edited prompt $p_\varepsilon$ (e.g., "*Eiffel Tower is located in* ___") and extract its hidden states $\{h_\varepsilon^\ell\}_{\ell=1}^{L}$ at the corresponding prediction position, reflecting how the model encodes the edited fact.

### 3.2 Layer-wise Visualization of Edited Knowledge Flow

> **Takeaway 1:** Edited knowledge is represented as a isolated residual stream in the latent space, indicating a fundamental issue with semantic integration.

To better understand how edited knowledge is represented and propagated within the model, we conduct a qualitative analysis based on residual stream visualization. Specifically, we compare the layer-wise hidden representations of the edited knowledge $\{h_\varepsilon^\ell\}_{\ell=1}^{L}$ and the reference knowledge $\{h_i^\ell\}_{\ell=1}^{L}$ to examine differences in how their semantic representations evolve across layers. We begin by applying Principal Component Analysis (PCA) (Jolliffe and Cadima, 2016) to reduce the dimensionality of the hidden states. In Figure 1(a), we plot the hidden states corresponding to edited fact as red diamonds, and those corresponding to the reference knowledge as green circles. Additional visualizations for other samples are provided in Appendix A.

As shown in Figure 1(a), there is a clear distinction between edited and reference knowledge in the model's latent space. The representations of reference knowledge evolve progressively across layers, becoming more dispersed in the deeper layers, which reflects their gradual integration into broader semantic contexts (Ju et al., 2024). In contrast, the representations of edited knowledge follow a separate and isolated path. This divergence suggests that the model encodes edited knowledge

[2] https://www.wikidata.org

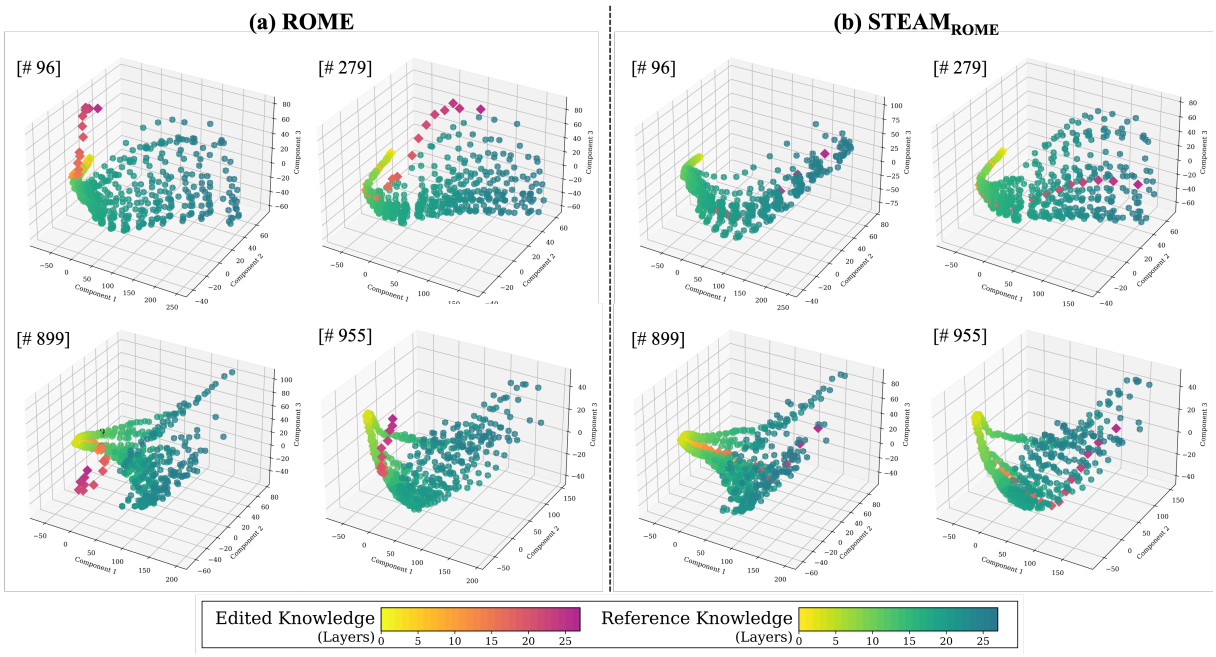


Figure 1: 3D visualization of residual stream across layers. Each subplot shows PCA-projected hidden states for a sample from the COUNTERFACT dataset (index in brackets). Red diamonds represent the residual stream of the edited knowledge, while green circles denote that of the reference knowledge. (a) shows the result after applying ROME, and (b) shows the result with STEAM$_{\text{ROME}}$.

independently, rather than incorporating it into the its broader knowledge structure.

Notably, this separation is apparent from the mid-layers, which are crucial for constructing knowledge associations (Geva et al., 2023; Ju et al., 2024). This observation raises concerns that the edited factual association is formed in a manner inconsistent with the preserved knowledge. Since LLMs organize knowledge in an entity-centric manner, similar to structured knowledge bases (AlKhamissi et al., 2022; Hu et al., 2023), this discrepancy potentially leads to difficulties in leveraging related facts, thereby hindering complex reasoning.

To complement the qualitative findings, we additionally perform a quantitative analysis using cosine similarity between hidden states of edited and reference knowledge. This supplementary analysis supports our observations and is presented in Appendix B.

### 3.3 Interpreting Reasoning Process of Edited Knowledge

**Takeaway 2:** Edited knowledge triggers shortcut-like activations that prioritize the target token generation, bypassing the model's natural reasoning process.

To further investigate the distinction identified in the previous analysis (§ 3.2), we conduct an interpretability-based analysis. Specifically, we employ LogitLens (nostalgebraist, 2020), a technique that approximates the token probability distribution at each transformer layer based on its hidden state. This approach enables us to trace how the model incrementally infers the edited knowledge (i.e., the object $o^*$) as information flows through the layers. Formally, given a hidden state $h \in \mathbb{R}^d$ at a particular layer, LogitLens applies the final projection matrix $W_U \in \mathbb{R}^{|V| \times d}$ followed by a softmax over the vocabulary space $V$:

$$\text{LogitLens}(h) = \text{softmax}(W_U h) \in \mathbb{R}^{|V|}. \quad (5)$$

We compute the layer-wise probability of generating $o^*$ using two sets of hidden states : (1) $\{h_\varepsilon^\ell\}_{\ell=1}^L$, which represent the reasoning process over the edited fact, and (2) $\{h_i^\ell\}_{\ell=1}^L$, which correspond to the reasoning process over the reference knowledge. The results in Figure 2 show clear differences in how the model processes edited knowledge (red line) versus reference knowledge (green line). For reference knowledge, the probability of predicting the target token increases gradually across layers, reflecting a progressive, context-driven reasoning process. In contrast, for edited knowledge, the probability rises sharply from the mid-layers and

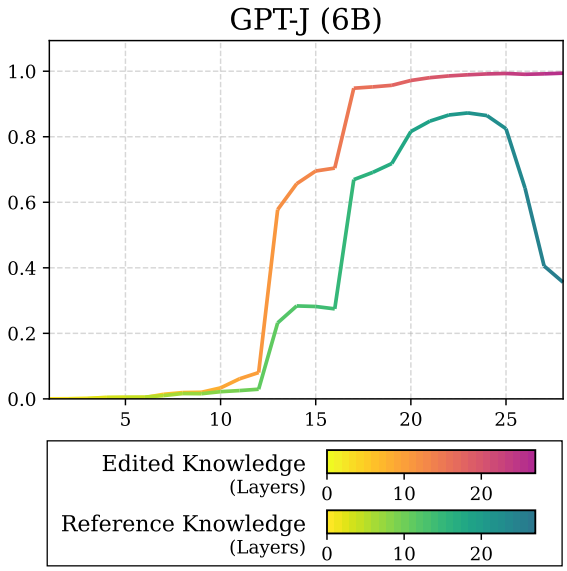


Figure 2: Layer-wise probability of generating the updated object $o^*$ in edited model $\mathcal{F}'$ as computed by LogitLens. The $x$-axis indicates the transformer layer; $y$-axis shows the predicted probability for $o^*$. Results are averaged over all samples in COUNTERFACT.

quickly saturates.

This contrast suggests that the model does not infer edited facts through its natural knowledge reasoning process. Instead, it relies on direct activation, particularly from the mid-layers, which acts as a shortcut to maximize the generation probability of the target token $o^*$. This behavior is a direct consequence of current editing strategies prioritizing token-level optimization, highlighting the need for approaches that achieve deeper semantic integration.

## 4 Method

To address the semantic isolation of edited knowledge observed in Section 3, we introduce STEAM, a framework designed to enhance the semantic coherence of updated knowledge with the model's internal knowledge representations. To achieve this, STEAM augments the standard locate-and-edit process, specifically enhancing the optimization of the value vector $v^*$ through two main components: (1) **Latent Positioning**, which identifies appropriate semantic anchors for edited knowledge in the model's latent space, and (2) **Latent-Level Alignment**, which incorporates an additional objective to steer the edited knowledge representation towards these anchors during the optimization process.

### 4.1 Latent Positioning

The Latent Positioning step aims to identify target representations for the updated factual association $(s, r \rightarrow o^*)$, which we refer to as *semantic anchors*. These anchors approximate how the model would naturally encode the new object $o^*$ in the context of the subject–relation pair $(s, r)$, assuming the fact had been acquired through the model's standard knowledge acquisition process (e.g., pre-training).

To construct these anchors for a given edit $\varepsilon : (s, r, o \rightarrow o^*)$, we start from the set of reference knowledge $T'$ obtained through the retrieval and filtering process described in Section 3.1. From this filtered set, we sample a balanced subset, denoted as $T''$, to mitigate relation-type biases and ensure the anchors reflect diverse relational contexts (details in Appendix C). For each prompt $p_i \in T''$, we extract the layer-wise hidden states $\{h_i^\ell\}_{\ell=1}^L$ from the base model at the final token position corresponding to $o^*$. These hidden states represent how the model internally encodes the target object $o^*$ when processing various subject–relation pairs $(s_i, r_i)$. To obtain a generalized representation that abstracts away from specific contexts and captures the consistent semantics of $o^*$, we compute the mean of these hidden states at each layer:

$$\varphi^\ell = \frac{1}{|T''|} \sum_{i=1}^{|T''|} h_i^\ell, \quad \ell = 1, \ldots, L. \tag{6}$$

The resulting set of vectors $\{\varphi^\ell\}_{\ell=1}^L$ serves as our semantic anchors, representing context-independent features of $o^*$ derived from diverse, faithfully recalled factual contexts. In our framework, we focus on semantic anchors $\varphi^\ell$ from the mid-layers. This choice is motivated by prior findings that these layers play a central role in encoding relational semantics and attribute-level information (Hernandez et al., 2023; Geva et al., 2023), making them suitable locus for integrating new factual knowledge in a manner consistent with the model's knowledge processing mechanism.

### 4.2 Latent-Level Alignment

The Latent-Level Alignment step guides the internal representation of the edited fact $(s, r, o^*)$ towards its corresponding semantic anchors $\varphi$ within the model's latent space. In the conventional locate-and-edit framework (as described in § 2), a new factual association $(s, r \rightarrow o^*)$ is encoded into a key-value pair $(k^*, v^*)$ within an MLP layer, where

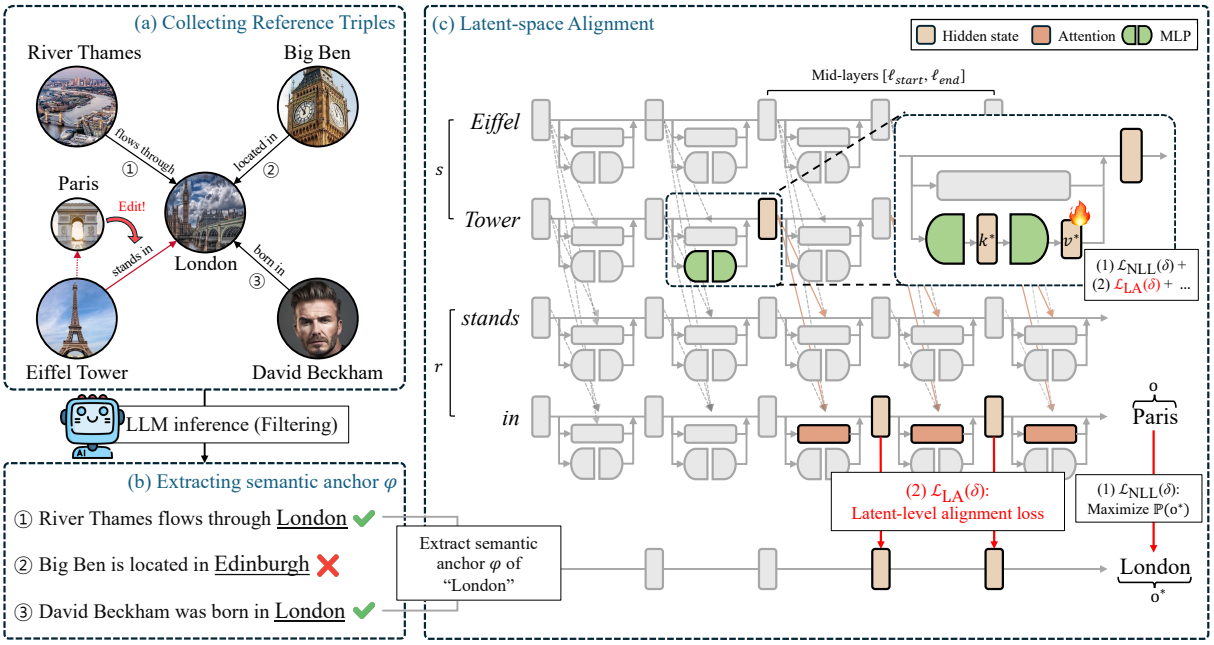


Figure 3: Overview of the STEAM framework. (a) Relevant reference knowledge about the new target object $o^*$ (b) The model is used to verify and filter these facts; valid references are then used to construct the semantic anchor $\varphi$ that approximates the latent representation of $o^*$. (c) During editing, STEAM introduces a latent-level alignment loss $\mathcal{L}_{\text{LA}}$, which guides the edited value vector $v^*$ to align with the semantic anchor across mid-layers, encouraging coherent integration of the new knowledge into the model's latent space.

the value vector $v^*$ encodes the information of the updated object $o^*$. The optimization of $v^*$ is typically guided by two objectives: the negative log-likelihood loss $\mathcal{L}_{\text{NLL}}$ and the KL divergence loss $\mathcal{L}_{\text{KL}}$. While these objectives are effective for improving token-level accuracy and specificity, they do not explicitly promote semantic coherence within the latent space.

To address this limitation, we introduce a latent-level alignment loss $\mathcal{L}_{\text{LA}}$, which penalizes the discrepancy between the hidden representations of edited fact and its corresponding semantic anchors. Specifically, during the iterative optimization process, we compute the cosine distance between the hidden state $h_\delta^\ell$ (obtained from a forward pass with the current candidate vector $\delta$), and anchor $\varphi^\ell$ at selected mid-layers $\ell \in [\ell_{\text{start}}, \ell_{\text{end}}]$. The alignment loss $\mathcal{L}_{\text{LA}}$ is defined as the average of these distances across layers:

$$\mathcal{L}_{\text{LA}}(\delta) = \frac{1}{N_{\text{align}}} \sum_{\ell'=\ell_{\text{start}}}^{\ell_{\text{end}}} \left[1 - \cos(h_\delta^{\ell'}, \varphi^{\ell'})\right], \quad (7)$$

where $N_{\text{align}} = \ell_{\text{end}} - \ell_{\text{start}} + 1$ denotes the number of layers included in the computation. We incorporate $\mathcal{L}_{\text{LA}}$ into the overall objective, yielding the following composite loss:

$$\min_\delta \mathcal{L}(\delta) = \mathcal{L}_{\text{NLL}}(\delta) + \mathcal{L}_{\text{KL}}(\delta) + \lambda \mathcal{L}_{\text{LA}}(\delta), \quad (8)$$

where the hyperparameter $\lambda$ controls the weight of the latent-level alignment loss term. By minimizing $\mathcal{L}_{\text{LA}}$, edited knowledge is more effectively integrated into the model's latent semantic space and becomes organically connected to related knowledge, rather than exist as a standalone piece of information.

## 5 Experiments

### 5.1 Experimental Settings

**Baseline Models.** We evaluate our approach using GPT-J (6B) (Wang and Komatsuzaki, 2021), Qwen2 (7B) (Yang et al., 2024), Llama 3 (8B) (Dubey et al., 2024) to ensure the robustness of our findings.

**Dataset.** We utilize the COUNTERFACTPLUS (Yao et al., 2023) dataset, which contains more challenging questions that require models to reason with edited knowledge. To construct robust semantic anchors, we exclude samples where the filtered reference set $T'$ contains fewer than 32 triples. Consequently, from the initial dataset of 1,031 samples, we utilized 816 for GPT-J, 927 for Qwen2, and 937

| Model | Editor | Edit. | Effi. | Para. | Neigh. | Port. | Flu. | Cons. |
|---|---|---|---|---|---|---|---|---|
| **GPT-J** (6B) | ROME | 62.9 | 100.0 | 99.5 | 79.1 | 32.4 | 619.7 | 46.4 |
| | STEAM$_{\text{ROME}}$ | **67.1** (+4.2) | 100.0 | 99.5 | 79.5 | **37.0** (+4.6) | 620.0 | **47.1** (+0.7) |
| | R-ROME | 62.7 | 100.0 | 99.3 | 80.5 | 31.9 | 621.0 | 46.2 |
| | STEAM$_{\text{R-ROME}}$ | **66.4** (+3.7) | 100.0 | 99.4 | 80.5 | **36.0** (+4.1) | 620.6 | **47.0** (+0.8) |
| **Qwen2** (7B) | ROME | 74.2 | 99.8 | 97.9 | 85.8 | 45.4 | 624.4 | 40.5 |
| | STEAM$_{\text{ROME}}$ | **75.5** (+1.3) | 99.8 | 98.0 | 85.8 | **47.4** (+2.0) | 624.9 | **40.9** (+0.4) |
| | R-ROME | 73.4 | 99.0 | 95.8 | 86.2 | 44.8 | 624.5 | 39.7 |
| | STEAM$_{\text{R-ROME}}$ | **75.3** (+1.9) | 99.7 | 96.8 | 86.1 | **47.3** (+2.5) | 625.1 | **40.4** (+0.7) |
| **Llama3** (8B) | ROME | 72.6 | 100.0 | 99.0 | 85.9 | 42.8 | 626.2 | 43.3 |
| | STEAM$_{\text{ROME}}$ | **74.9** (+2.3) | 99.6 | 99.2 | 87.2 | **45.9** (+3.1) | 622.7 | 42.7 (-0.6) |
| | R-ROME | 71.9 | 100.0 | 98.4 | 86.9 | 41.7 | 625.4 | 42.4 |
| | STEAM$_{\text{R-ROME}}$ | **76.0** (+4.1) | 99.9 | 99.1 | 87.5 | **47.4** (+5.7) | 622.8 | 42.2 (-0.2) |

Table 1: Single-editing results on COUNTERFACTPLUS for GPT-J (6B), Qwen2 (7B), and Llama3 (8B). Results are averaged over three runs. Values in parentheses indicate the improvement or decline relative to the baseline for Edit score, Portability, and Consistency.

for Llama 3.

**Knowledge editing approaches.** We evaluate our method under two scenarios: (1) ***Single editing***, where each factual update is applied independently (i.e., the model is restored after each edit), and (2) ***Batch editing***, where multiple edits are applied simultaneously. For single editing, we apply our framework to existing methods, including ROME (Meng et al., 2022) and R-ROME (Gupta and Anumanchipalli, 2024). For batch editing, we integrate our method with PMET (Li et al., 2024), a state-of-the-art batch editing method. Implementation details are provided in Appendix E.

**Metrics.** We evaluate all methods using several metrics: ***Efficacy Score*** (Effi.) measures whether the model accurately recalls the updated fact; ***Paraphrase Score*** (Para.) assesses the generalizability of the edit; ***Neighborhood Score*** (Neigh.) evaluates whether unrelated knowledge remains unaffected; and ***Portability Score*** (Port.) measures the model's ability to apply the edited knowledge in multi-hop reasoning. ***Edit Score*** (Edit.) is the harmonic mean of these four metrics. Additionally, we report ***Fluency*** (Flu.), which reflects lexical diversity of generated text, and ***Consistency*** (Cons.), which captures semantic coherence.

## 5.2 Results

### 5.2.1 Single Editing

In the single editing scenario, we evaluate our STEAM framework by integrating it into both ROME and R-ROME, denoted as STEAM$_{\text{ROME}}$ and STEAM$_{\text{R-ROME}}$, respectively. For all models, we set $\ell_{\text{start}} = 13$, $\ell_{\text{end}} = 17$, and $\lambda = 5$, and provide a detailed hyperparameter analysis in Appendix G.

As summarized in Table 1, STEAM demonstrates consistent improvements in editing quality across all evaluated models. Portability increases robustly in all cases, with the highest gain of +5.7 observed in Llama3 under STAKE$_{\text{R-ROME}}$, indicating that edited knowledge is more effectively integrated and applied in multi-hop reasoning. Importantly, these gains are observed consistently across both baseline editors, ROME and R-ROME, underscoring the generality of our framework.

Other metrics such as Efficacy, Paraphrase, and Neighborhood remain stable, with slight gains in some cases, showing that STEAM preserves the local accuracy and generalization ability of the baseline editors. These steady or improved results collectively contribute to an overall enhancement of the aggregated Edit score. Notably, GPT-J and Qwen2 show modest but consistent improvements in Consistency (up to +0.8 and +0.7, respectively), highlighting STEAM's ability to enhance semantic coherence in edited knowledge.

| Model | Edits | Editor | Edit. | Effi. | Para. | Neigh. | Port. | Flu. | Cons. |
|---|---|---|---|---|---|---|---|---|---|
| **GPT-J** (6B) | 1 | PMET | 63.5 | 100.0 | 97.0 | 82.2 | 32.8 | 620.7 | 45.3 |
| | | STEAM$_{\text{PMET}}$ | **65.5** (+2.0) | 100.0 | 96.6 | 82.4 | **35.0** (+2.2) | 619.0 | **46.4** (+1.1) |
| | 10 | PMET | 65.0 | 100.0 | 97.2 | 82.2 | 34.4 | 620.6 | 45.2 |
| | | STEAM$_{\text{PMET}}$ | **65.9** (+0.8) | 100.0 | 96.6 | 82.4 | **35.4** (+1.0) | 618.9 | **46.3** (+1.1) |
| | 100 | PMET | 64.1 | 99.9 | 96.9 | 81.6 | 33.5 | 621.3 | 45.3 |
| | | STEAM$_{\text{PMET}}$ | **66.0** (+2.0) | 99.9 | 96.6 | 81.9 | **35.7** (+2.2) | 619.2 | **46.3** (+1.0) |
| | All | PMET | 63.7 | 100.0 | 96.4 | 77.2 | 33.9 | 621.5 | 45.0 |
| | | STEAM$_{\text{PMET}}$ | **65.0** (+1.3) | 100.0 | 96.0 | 77.2 | **35.5** (+1.6) | 619.1 | **46.1** (+1.1) |

Table 2: Batch-editing results on COUNTERFACTPLUS for GPT-J (6B). Results are averaged over three runs. Values in parentheses indicate improvements of STEAM$_{\text{PMET}}$ over the PMET baseline for Edit., Portability (Port.), and Consistency (Cons.). Each row reports performance at different batch sizes (1, 10, 100, and all 816 edits).

#### 5.2.2 Batch Editing

In the batch-editing scenario, we incorporate our STEAM framework into PMET, denoted as STEAM$_{\text{PMET}}$, and evaluate it on GPT-J. Experiments are conducted with batch sizes of 1, 10, 100, and all 816 edits in the dataset, as reported in Table 2.

Across all batch sizes, STEAM$_{\text{PMET}}$ consistently improves Portability (up to +2.2), indicating that edited knowledge is more effectively transferred and applied even under large-scale editing. Edit scores also increase across settings (up to +2.0), suggesting that the overall quality of edits benefits from semantic alignment. Consistency further exhibits modest but reliable gains (up to +1.1), highlighting that STEAM helps preserve semantic coherence in batch-editing conditions. Meanwhile, other metrics such as Efficacy, Paraphrase, and Neighborhood remain stable, showing that local factual accuracy and generalization are not sacrificed. Collectively, these results demonstrate that STEAM provides robust improvements in knowledge integration across varying batch sizes.

### 5.3 Discussion

**Verifying Latent-Space Alignment of Edited Knowledge.** To assess whether the performance improvements achieved by STEAM stem from latent-level integration, we revisit the latent space visualizations. As shown in Figure 1(b), which illustrates the residual streams produced by STEAM$_{\text{ROME}}$, the hidden states of the edited knowledge (red diamonds) exhibit trajectories that are more closely aligned with those of the reference knowledge (green circles), compared to the baseline ROME shown in Figure 1(a). This alignment suggests that STEAM effectively guides the representation of edited facts toward semantically coherent regions within the model's latent space. These findings indicate that the observed performance gains are the result of genuine semantic-level integration, rather than surface-level adjustments. To further support this interpretation, we conduct a layer-wise cosine similarity analysis (Appendix B), which quantitatively confirms stronger alignment between the edited representations and their corresponding semantic anchors.

## 6 Conclusion

In this paper, we addressed a limitation of current knowledge editing methods: their failure to coherently integrate updated knowledge into the model's existing structure despite successful output changes. Our analysis revealed that existing methods often induce isolated latent representations, hindering consistent and reliable inference with edited knowledge. To address this issue, we introduced STEAM, a framework designed to promote deeper semantic integration. Our experimental results demonstrate that enhancing semantic alignment improves the model's reasoning capabilities with edited knowledge and increases semantic coherence. These findings emphasize the importance of semantic-level integration for developing more robust and coherent knowledge editing techniques.

## Limitation

**Dependence on External Knowledge.** STEAM constructs semantic anchors based on reference facts retrieved from external structured knowledge sources such as Wikidata. This design enables the model to leverage well-grounded factual associations when guiding semantic alignment. Therefore, for newly emerging or less well-known entities, retrieving relevant reference knowledge may be difficult, which can limit the applicability of the proposed framework.

**Anchor Construction and Knowledge Selection.** STEAM assumes that the latent representation of an updated fact can be approximated by aggregating reference knowledge about the same object. While this offers a practical signal for alignment, it may not fully reflect how language models internally structure and reason over knowledge. This process in closely related to broader questions about the structure of factual reasoning in language models – a topic that remains underexplored. We view our method as a first step in this direction and leave more systematic strategies for anchor construction and deeper investigation of knowledge inference mechanisms for future work.

## Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00553041, Enhancement of Rational and Emotional Intelligence of Large Language Models for Implementing Dependable Conversational Agents) and the Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (RS-2024-00398115, Research on the reliability and coherence of outcomes produced by Generative AI).

## References

Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. 2022. Towards tracing knowledge in language models back to the training data. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. 2024. How do large language models acquire factual knowledge during pretraining? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

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*, 11:283–298.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Akshat Gupta and Gopala Anumanchipalli. 2024. Rebuilding rome: Resolving model collapse during sequential model editing. *arXiv preprint arXiv:2403.07175*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2023. Linearity of relation decoding in transformer language models. *arXiv preprint arXiv:2308.09124*.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2023. A survey of knowledge enhanced pre-trained language models. *IEEE Transactions on Knowledge and Data Engineering*.

Ian T Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

Tianjie Ju, Weiwei Sun, Wei Du, Xinwei Yuan, Zhaochun Ren, and Gongshen Liu. 2024. How large

language models encode context knowledge? a layer-wise probing study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8235–8246, Torino, Italia. ELRA and ICCL.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

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

nostalgebraist. 2020. Interpreting gpt: The logit lens.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Publisher correction: Large language models encode clinical knowledge. *Nature*, 620(7973):19–19.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Mengru Wang, Yunzhi Yao, Ziwen Xu, Shuofei Qiao, Shumin Deng, Peng Wang, Xiang Chen, Jia-Chen Gu, Yong Jiang, Pengjun Xie, et al. 2024a. Knowledge mechanisms in large language models: A survey and perspective. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7097–7135.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024b. Knowledge editing for large language models: A survey. *ACM Comput. Surv.*, 57(3).

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024c. MMLU-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

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

Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. 2018. Generating informative and diverse conversational responses via adversarial information maximization. *Advances in Neural Information Processing Systems*, 31.

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.

# Appendix

## A Quantitative Latent-Space Analysis via Cosine Similarity

To complement the qualitative findings presented in Section 3.2, we provide additional visualization results for both GPT-J and GPT-2 XL (Radford et al., 2019). These visualizations illustrate how edited knowledge propagates through the residual stream and how it aligns (or diverges) from reference knowledge across transformer layers.

Figure 5 presents PCA-projected hidden states for multiple samples from the COUNTERFACT dataset. Red diamonds represent the residual stream of the edited knowledge, while green circles correspond to the reference knowledge recalling the same target object $o^*$. Under baseline editing methods (ROME), the edited knowledge forms a semantically isolated stream in the latent space, following a path that deviates from reference knowledge. In contrast, under STEAM-enhanced editing, the edited trajectories are more closely aligned with reference knowledge, indicating improved semantic integration. Figure 6 provides corresponding results for GPT-2 XL. Despite architectural differences and increased depth, we observe a similar trend, reinforcing the model-agnostic effectiveness of STEAM. These qualitative observations, consistent across models and examples, provide strong visual evidence that STEAM facilitates meaningful representational alignment of edited knowledge within the latent space.

## B Measuring Semantic Alignment

While existing editing methods can successfully update model outputs, it remains unclear whether such updates are semantically integrated into the model's internal knowledge structure. To investigate this, we conduct a layer-wise cosine similarity analysis between the edited knowledge representation and a reference semantic anchor vector.

For each edit $(s, r, o \rightarrow o^*)$, we first construct a baseline semantic representation $\{\varphi^\ell\}_{\ell=1}^L$ for the updated object $o^*$ by aggregating the hidden states of reference facts, as described in Section 4.1. We then compare how the edited model $\mathcal{F}'$ represents $o^*$ in response to the edited prompt $p_\varepsilon$ (e.g., `Eiffel Tower stands in __`) by extracting the hidden states $\{h_\varepsilon^\ell\}_{\ell=1}^L$ and computing:

$$\mathcal{S}(\varphi, h_\varepsilon) = \left\{ \cos\left(\varphi^\ell, h_\varepsilon^\ell\right) \right\}_{\ell=1}^{L},$$

where $\cos(\cdot,\cdot)$ denotes the cosine similarity. We also apply the same procedure to the unedited model $\mathcal{F}$, obtaining $\mathcal{S}(\varphi, h_\theta)$. Since $(s, r) \rightarrow o$ (e.g., `Eiffel Tower stands in` $\rightarrow$ `Paris`) and $(s_i, r_i) \rightarrow o^*$ (e.g., `River Thames flows through` $\rightarrow$ `London`) are semantically unrelated facts, this comparison provides a reference for assessing the inherent similarity between different knowledge representations in the model.

Figure 4 (a) presents the layer-wise cosine similarity scores for GPT-J under the ROME baseline. Where the red curve represents $S\big(\varphi, h_\varepsilon\big)$ for the edited model and the blue curve represents $S\big(\varphi, h_\theta\big)$ for the base model. Up to the edit layer, both curves are the same; after that, they diverge, showing a representational shift induced by the factual update. However, this divergence does not necessarily indicate that the updated knowledge has been meaningfully integrated. Instead, as the layers progress, the red curve follows a pattern similar to the blue curve while showing a gradual decline. Given the distinct semantics of $o$ and $o^*$ (e.g., `Paris` vs. `London`), these findings suggest that although the edit does change the updated fact's representation, the new knowledge remains separate from the original latent structure.

Figure 4 (b) shows the corresponding analysis for the STEAM-enhanced model. In contrast to the baseline, the red curve (after editing) exhibits a clear upward trend after the edit layer, diverging from the blue curve (before editing) and aligning more closely with the semantic anchor $\varphi^\ell$. This indicates that the updated knowledge $(s, r) \rightarrow o^*$ is not only distinguishable from the original fact $(s, r) \rightarrow o$, but is also actively aligned with the reference knowledge encoding of $o^*$. These results quantitatively validate the alignment mechanism introduced in our framework and support the interpretability of STEAM's improvements in semantic coherence.

## C Reference Triple Selection and Filtering Criteria

For each editing sample $\varepsilon$, we begin by collecting a set of reference triples $T = \{(s_i, r_i, o^*)\}_{i=1}^N$. We then transform each triple into a textual prompt $p_i$. Following previous work (Zhong et al., 2023), we define a template $t_{r_i}$ for each relation $r_i$. By inserting the subject $s_i$ into this template, we generate a textual prompt $p_i = t_{r_i}(s_i)$. For instance, if $s_i$ is "United States" and $r_i$ is "capital of", then

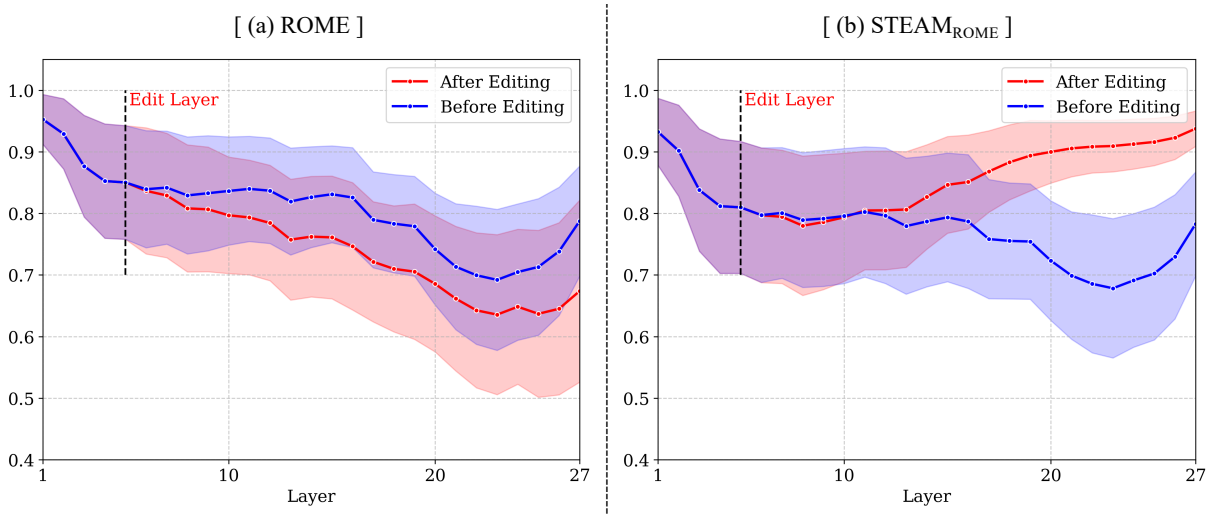


Figure 4: Layer-wise cosine similarity between model representations and semantic anchors in GPT-J. Each plot shows the average cosine similarity between anchor vectors $\varphi^{\ell}$ and hidden states from the edited model $h_{\varepsilon}^{\ell}$ (red) and the unedited model $h_{\theta}^{\ell}$ (blue), with shaded areas indicating standard deviation. The vertical dashed line marks the edit layer. (a) Result with ROME. (b) Result with STEAM$_{\text{ROME}}$.

$p_i$ is "The capital of United States is", which is designed to elicit $o^*$. Next, we input $p_i$ into the unedited model $\mathcal{F}$, performing greedy decoding for $k$ subword tokens, where $k$ corresponds the subword length of $o^*$. Let $\text{dec}(\mathcal{F}(p_i), k)$ denote the top-ranked sequence of length $k$. If this sequence exactly matches $o^*$, we infer that the model recognizes this fact. Formally, the filtered set $T' \subseteq T$ is

$$T' = \left\{ (s_i, r_i, o^*) \in T \;\middle|\; \text{dec}\big(\mathcal{F}(p_i),\, k\big) = o^* \right\}. \quad (9)$$

To construct a balanced anchor set, we apply a stratified sampling procedure to the filtered set $T'$. Specifically, we group all triples in $T'$ by their relation type $r_i$, and assign an equal base number of samples to each group. Any remaining quota is distributed proportionally based on group sizes. If a group contains fewer examples than its assigned quota, we include all available triples from that group. Finally, if the total number of selected samples is still below the target size, we randomly sample additional instances from the remaining pool to complete the set. The full procedure is detailed in Algorithm 1. This sampling strategy ensures that the resulting subset $T'' \subseteq T'$ reflects a diverse and representative distribution of relation types, thereby enabling the construction of robust and unbiased semantic anchors. In our implementation, we set the maximum size of $T''$ to 64 reference triples per edit.

## D COUNTERFACTPLUS

The COUNTERFACTPLUS dataset is an enhanced version of the original COUNTERFACT benchmark, comprising 1,031 selected entries from the original data. Table 3 presents an example from the COUNTERFACTPLUS dataset. Each sample includes a factual edit $\varepsilon$, such as (Skype, product of, Microsoft $\rightarrow$ Apple), diverse set of prompts that assess the impact of the edit from multiple perspectives.

- **Efficacy Prompt** ($P^E$) is used to assess whether the edited knowledge is accurately reflected in the model's output.
- **Paraphrase Prompt** ($P^P$) tests whether the edited information is consistently maintained across different surface forms.
- **Neighborhood Prompt** ($P^N$) provides unrelated factual contexts and is used to evaluate whether the model preserves non-target knowledge after editing.
- **Multi-hop Question** ($q'$) assesses whether the model can reason over multiple connected facts involving the updated knowledge. Specifically, each question requires the model to combine the edited fact $(s, r, o^*)$ with an additional fact $(o^*, r', o')$, and infer the answer $o'$ based on this reasoning path. This evaluates whether the updated object $o^*$ has been semantically integrated into the model's broader knowledge structure in a way that supports multi-step inference.

| Property | Value |
|---|---|
| Edit request ($\varepsilon$) | (Skype, product of, Microsoft $\rightarrow$ Apple) |
| Efficacy prompt ($P^E$) | Skype was a product of |
| Paraphrase prompt ($P^P$) | He moved to Edmonton, Alberta, when he was seven-years old. Skype was created by |
| Neighborhood prompt ($P^N$) | Windows Media Center, a product created by |
| Recalled knowledge ($o^*, r', o'$) | (Apple, founded by, Steve Jobs and Steve Wozniak) |
| Multi-hop question ($q'$) | Who are the founders of the company that created Skype? |
| Multi-hop Answer ($o'$) | Steve Jobs and Steve Wozniak |

Table 3: An Example of COUNTERFACTPLUS Dataset

## E Implementation Details

Our single-editing experiments were conducted on GPT-J (6B), Qwen2 (7B), and Llama3 (8B), while batch-editing experiments were performed only on GPT-J (6B). All experiments were run on an NVIDIA A100 (80GB) GPU using the Adam optimizer.

For ROME and R-ROME, the primary editing layer is set to 5 across all models, based on prior findings about where factual associations are stored. The value vector $v^*$ is optimized for 20 gradient steps with a learning rate of 0.5 and a weight decay of 0.5. The loss is computed at layer 27, and the KL divergence regularization factor is set to 0.0625.

For PMET (batch editing), we follow the original configuration and apply edits across a broader range of layers [3, 4, 5, 6, 7, 8] on GPT-J. We set the number of gradient steps to 30, the learning rate to 0.2, and the clamp normalization factor to 0.75. The KL regularization factor is set to 1.0, and the loss is computed at layer 27.

## F Metrics

We evaluate the performance of knowledge editing methods using six metrics that assess factual accuracy, generalization, locality, and semantic coherence. These metrics are computed over samples from the COUNTERFACTPLUS dataset and are defined as follows:

- **Efficacy Score** evaluates whether the edited model $\mathcal{F}'$ correctly recalls the updated object $o^*$ given the edit prompt $p \in P^E$. It is computed as:

$$\mathbb{E}_{p \in P^E} \left[ \mathbb{I} \left[ \mathbb{P}_{\mathcal{F}'} \left( o^* \mid p \right) > \mathbb{P}_{\mathcal{F}'} \left( o \mid p \right) \right] \right],$$

  where $\mathbb{E}$ denotes the average, $\mathbb{I}$ denotes the indicator function, and $\mathcal{F}'$ represents the edited model. $\mathbb{P}(o \mid p)$ indicates the probability that the model generates output $o$ given the input prompt $p$.

- **Paraphrase Score** measures whether the model maintains the updated knowledge under paraphrased prompts $p \in P^P$. The formulation mirrors that of the Efficacy Score:

$$\mathbb{E}_{p \in P^P} \left[ \mathbb{I} \left[ \mathbb{P}_{\mathcal{F}'} \left( o^* \mid p \right) > \mathbb{P}_{\mathcal{F}'} \left( o \mid p \right) \right] \right].$$

- **Neighborhood Score** assesses whether the model preserves unrelated knowledge after editing. For neighborhood prompts $p \in P^N$, which should not reflect the updated fact, the metric is computed as:

$$\mathbb{E}_{p \in P^N} \left[ \mathbb{I} \left[ \mathbb{P}_{\mathcal{F}'} \left( o^* \mid p \right) > \mathbb{P}_{\mathcal{F}'} \left( o \mid p \right) \right] \right].$$

- **Portability Score** evaluates whether the model can correctly answer multi-hop questions that require reasoning over the edited knowledge. Each question $q'$ is constructed to test whether the model can infer the final object $o'$ by combining the edited fact $(s, r, o^*)$ with an additional fact $(o^*, r', o')$. The model's response $\mathcal{F}'(q')$ is evaluated against the gold answer $o'$ using fuzzy string matching. A prediction is considered correct if the partial ratio ($PR$) between the model's output and $o'$ exceeds 0.7:

$$\mathbb{E}_{q', o' \in Q} \left[ \mathbb{I} \left[ PR(\mathcal{F}'(q'), o') > 0.7 \right] \right].$$

  where $PR$ denotes the partial ratio from the Fuzzywuzzy library[3], based on Levenshtein distance.

- **Fluency** measures evaluates the repetitiveness of generated text by computing the weighted average entropy over bi-grams and tri-grams, following the method proposed by Zhang et al., 2018. Formally, it is calculated as:

$$\sum_k f(k) log_2 f(k)$$

[3] https://github.com/seatgeek/fuzzywuzzy

**Algorithm 1** Stratified Sampling by Relation Type

**Require:** Dataset $T$, Sample size $N$
**Ensure:** Stratified sample $T'$

```
/* Check if dataset is empty */
if T is empty then
    return ∅
end if
/* Group dataset by relation type */
G ← group T by relation type
R ← |G|
/* Compute base and extra samples */
n ← ⌊N/R⌋
r ← N − (n × R)
/* Initialize sampled dataset */
T' ← ∅
/* Stratified sampling per group */
for all g ∈ G do
    s ← n
    if r > 0 then
        p ← |g|/|T|
        a ← ⌊r × p⌋
        s ← s + a
    end if
    T' ← T' ∪ RandomSample(g, min(s, |g|))
end for
/* Handle remaining samples if needed */
if |T'| < N then
    U ← T \ T'
    m ← N − |T'|
    T' ← T' ∪ RandomSample(U, min(m, |U|))
end if
/* Return final sampled dataset */
return T'
```

where $f(k)$ denotes the frequency distribution of n-grams. Lower values indicate repetitive or less diverse outputs, while higher values reflect more natural and varied language generation.

- **Consistency** measures the semantic consistency of the generated outputs. To compute this score, we generate text from a generation prompt $p \in P^G$, and calculate the cosine similarity between the TF-IDF vectors of the generated text and reference texts($t_{ref}$) about subjects sharing the target property $o^*$. Formally, it is calculated as:

$$\mathbb{E}_{p \in P^G}[cos(\mathcal{F}'(p), t_{ref})]$$

## G Analyzing the Impact of Alignment Strength and Layer Choice

To better understand how the latent-level alignment in STEAM affects the integration of edited knowledge, we analyze two key factors: the alignment strength ($\lambda$) and the depth of the alignment layer range. All experiments in this analysis are conducted on GPT-J (6B), using $\text{STEAM}_{\text{ROME}}$.

**Effect of Alignment Strength.** We first fix the alignment layers to $\ell_{\text{start}} = 13$ and $\ell_{\text{end}} = 18$, and vary the alignment strength $\lambda \in 1, 3, 5, 10$. Results show that moderate increases in $\lambda$ improve semantic integration. For example, Portability improves from 34.7 ($\lambda = 1$) to 37.0 ($\lambda = 5$), and Consistency increases from 47.0 to 47.2. However, when $\lambda$ is further increased to 10, these gains do not continue to rise and begin to plateau or slightly decline. Additionally, Fluency decreases steadily with stronger alignment (620.7 → 619.6 → 618.5), suggesting that overly strong alignment may introduce artifacts or reduce generation quality. These results imply that while stronger alignment encourages integration of edited knowledge, there exists an optimal range beyond which further gains are marginal or even detrimental.

**Effect of Alignment Layer Depth.** Next, we fix $\lambda = 5$ and vary the alignment layers across three ranges: [8–13], [13–18], and [18–23]. We observe that deeper alignment layers generally yield higher Portability (35.0 → 35.7 → 37.9), while Consistency remains stable (46.8 → 47.2). However, Fluency shows a steady decline (621.1 → 619.6 → 618.5) as the alignment moves to later layers. These findings suggest that deeper layers, while capable of capturing more abstract semantics, may lack representational stability, making them less ideal as anchor points. Mid-layer ranges thus offer a favorable balance – rich enough in semantic information while still providing stable targets for alignment.

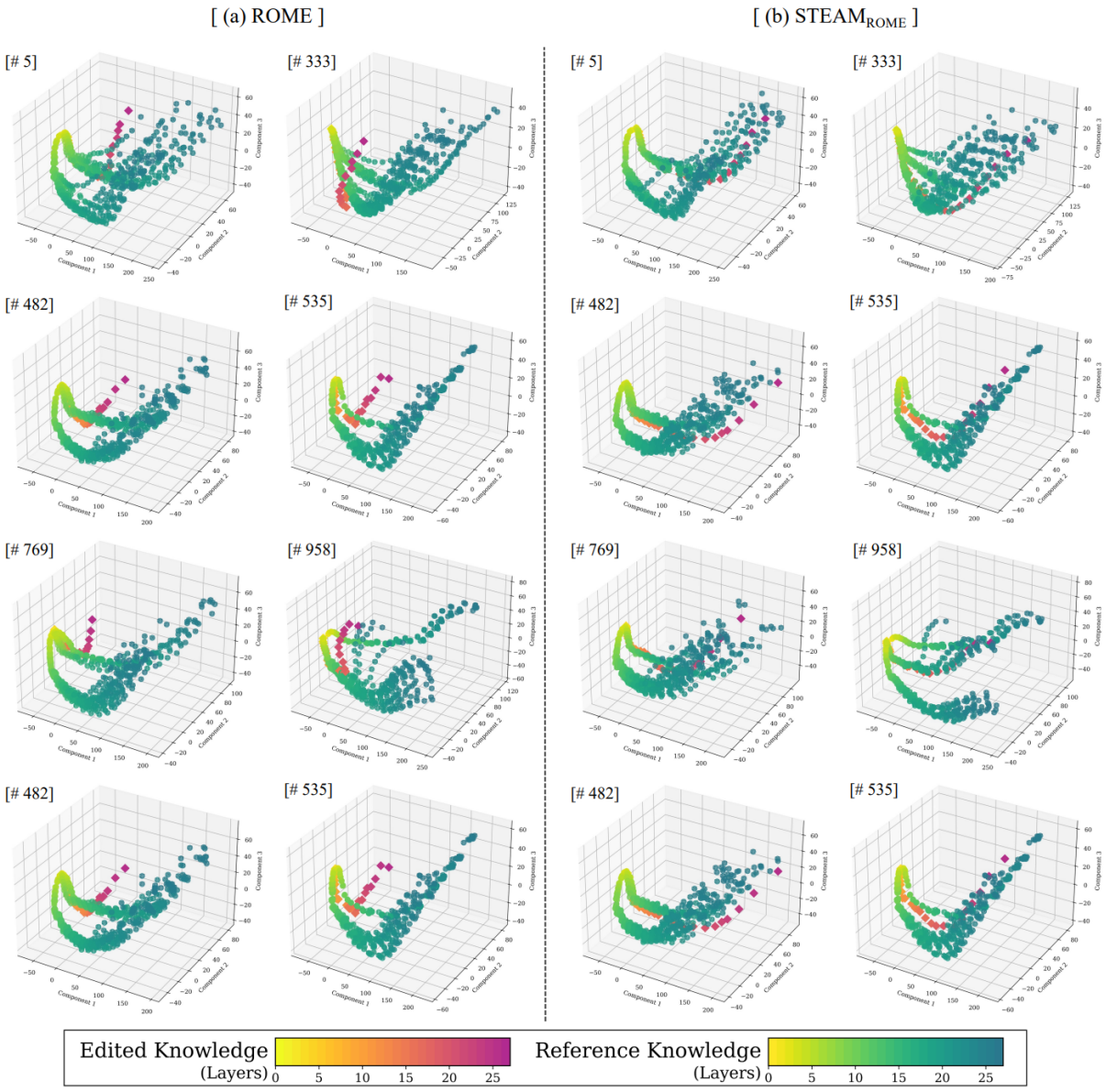


Figure 5: 3D visualization of residual stream representations across layers for GPT-J. Each subplot shows PCA-projected hidden states of an edited fact (red diamonds) and its corresponding reference facts (green circles), with sample indices indicated in brackets. (a) shows results under ROME, and (b) under STEAM$_{ROME}$.

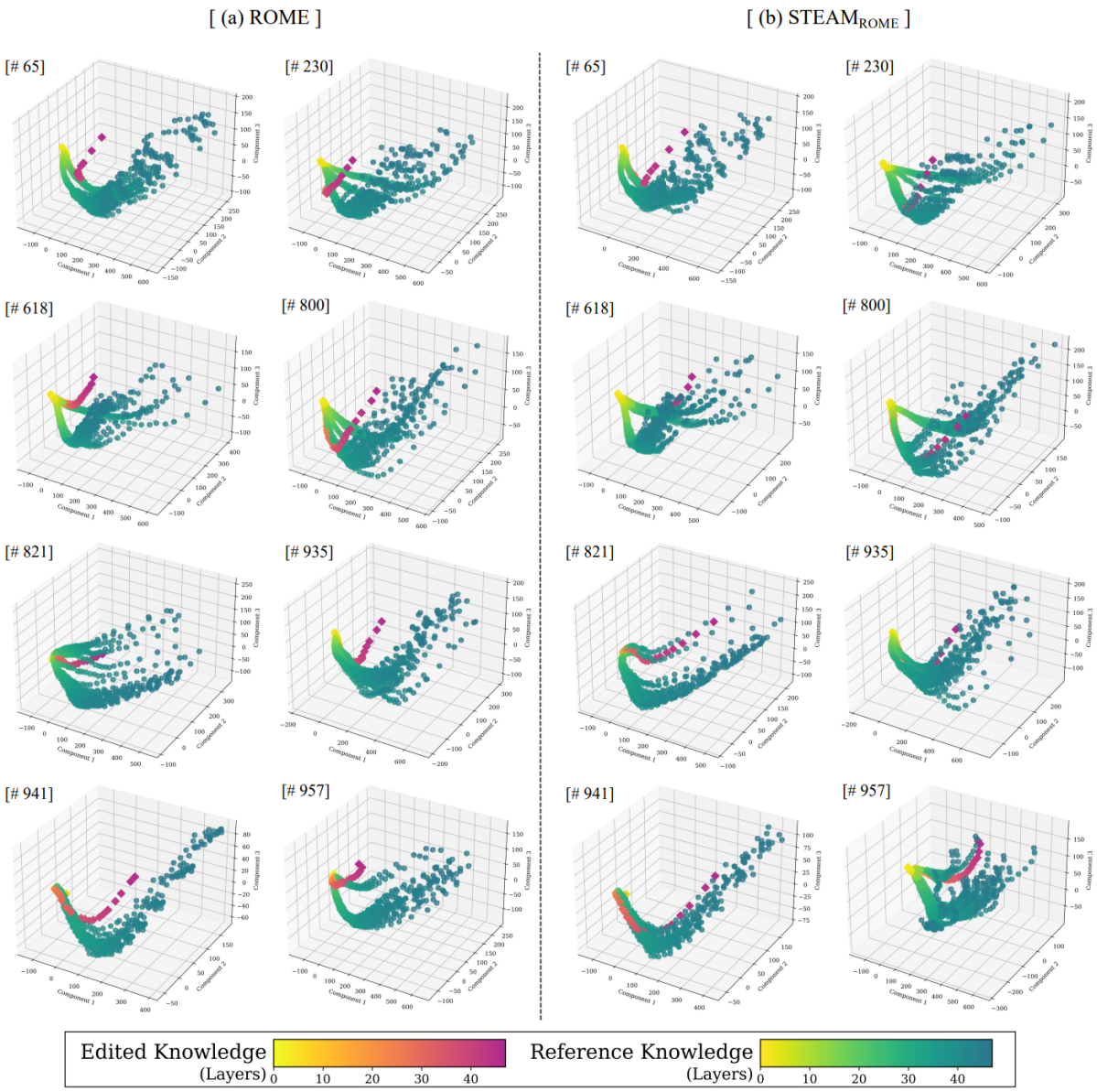


Figure 6: 3D visualization of residual stream representations across layers for GPT2-XL. Each subplot shows PCA-projected hidden states of an edited fact (red diamonds) and its corresponding reference facts (green circles), with sample indices indicated in brackets. (a) shows results under ROME, and (b) under STEAM$_{ROME}$.