FiNE: Fine-grained Neuron-level Model Editing for Reliable and Safe LLMs

Xun Yang, Haowen Pan, Xiaozhi Wang, Yixin Cao, Juanzi Li, Meng Wang, *Fellow, IEEE*

***Abstract*—The rapid advancement of Large Language Models (LLMs) has underscored the need for editing techniques that enhance both the reliability and safety of model outputs. While prior locate-then-edit methods, such as ROME, achieve factual correction via causal tracing, they often overlook the internal structure of Feed-Forward Networks (FFNs) and the relational context among edited concepts, leading to brittle or incomplete updates. To address these limitations, we propose FiNE (Fine-grained Neuron-level Editing), a unified locate-then-edit framework that performs gradient-free localization and fine-grained parameter adjustment within FFNs. FiNE introduces a theoretically grounded contribution score, which can be interpreted as an efficient variant of gradient-based methods, to identify concept-relevant neurons. It further employs a composite loss to balance editing accuracy, model consistency, and output diversity. We instantiate FiNE for two complementary applications, knowledge editing (FiNE-K) and safety editing (FiNE-S), which together demonstrate the versatility of the framework. Extensive experiments on the KnowEdit and SafeEdit benchmarks show that FiNE achieves superior precision, robustness, and efficiency with minimal disruption to the model's general behavior. These results highlight the effectiveness and scalability of fine-grained neuron editing as a unified approach to building more reliable and safer LLMs.**

***Index Terms*—Explainable artificial intelligence, neural explainability, knowledge editing, safety enhancement.**

I. Introduction

As Large Language Models (LLMs) continue to advance rapidly [1]–[5], ensuring the reliability and safety of their outputs has become a critical challenge, thereby creating a growing demand for effective model editing techniques. Factual inaccuracies, outdated knowledge, and inconsistent responses can undermine the trustworthiness of LLMs in knowledge-intensive applications [6]–[8]. Knowledge editing methods aim to correct erroneous information, update obsolete facts, and enhance the consistency of model outputs without compromising their generalization capabilities [9]–[14]. On the other hand, ensuring the safety of LLMs is paramount to prevent the generation of harmful or undesirable content, which poses significant risks in real-world deployments [15]–[17]. Methods to protect model safety focus on aligning LLMs with ethical guidelines and reducing the likelihood of toxic or biased outputs [18]–[20]. Recently, Wang *et al.* [21] introduced the SafeEdit benchmark to evaluate detoxification through editing methods. Recognizing that both reliability editing and safety editing require identifying and intervening on a compact set of target-relevant model components, we investigate whether they can be addressed within a common neuron-level editing formulation. To this end, we build upon recent progress in knowledge editing, leading to the development of increasingly precise and controllable techniques.

Recent advancements in knowledge editing have introduced a variety of techniques, including memory-based editors [22]–[25], meta-learning strategies [26]–[29], and locate-then-edit approaches [10]–[14]. Among these, locate-then-edit methods have gained prominence due to their ability to pinpoint and modify specific knowledge within LLMs, enhancing the output reliability. A seminal contribution in this domain is ROME [11], which leverages causal tracing to identify components responsible for recalling factual information about entities. ROME has inspired subsequent techniques, such as MEMIT [12], PMET [13] and AlphaEdit [14], which further refine causal tracing, cementing its role as a cornerstone of knowledge editing. Despite their promise, locate-then-edit methods face significant challenges. Hase *et al.* [30] argue that causal tracing provides limited insight into which Feed-Forward Network (FFN) layers should be modified to update knowledge effectively. This limitation often results in edits that are overly dependent on subject entities, neglecting relations in context [31]. For instance, when the relation in a factual statement is altered during locality testing, edited models frequently fail to generate correct outputs, instead reproducing the original target object. These issues, obscured by flaws in datasets like CounterFact [11], highlight the need for more fine-grained localization to guide effective knowledge editing. Therefore, we turn our attention to neuron-level localization, building on prior work demonstrating the influence of individual neurons on model behavior [10], [20], [32], [33]. Notably, Schwettmann *et al.* [33] introduced a gradient-based method to identify multi-modal neurons that translate visual inputs into textual outputs, offering valuable interpretability. However, their approach is computationally intensive and narrowly applicable due to costly gradient calculations, with its theoretical and practical implications underexplored.

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant U22A2094, Yangtze River Delta Science and Technology Innovation Community Joint Research (Basic Research) Project under Grant 2025CSJZN01600, Beijing Natural Science Foundation under Grant L243006, NSFC grant 62472138, and also by the advanced computing resources provided by the Supercomputing Center of the USTC. (*Corresponding author: Yixin Cao.*)

Xun Yang and Haowen Pan are with University of Science and Technology of China, Hefei 230026, China (e-mail: xyang21@ustc.edu.cn; phw1129@mail.ustc.edu.cn).

Xiaozhi Wang and Juanzi Li are with the Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing 100084, China (e-mail: wangxz20@mails.tsinghua.edu.cn; lijuanzi@tsinghua.edu.cn).

Yixin Cao is with the School of Computer Science, Fudan University, Shanghai 200433, China (e-mail: caoyixin2011@gmail.com).

Meng Wang is with Hefei University of Technology, Hefei 230009, China (e-mail: wangmeng@hfut.edu.cn).

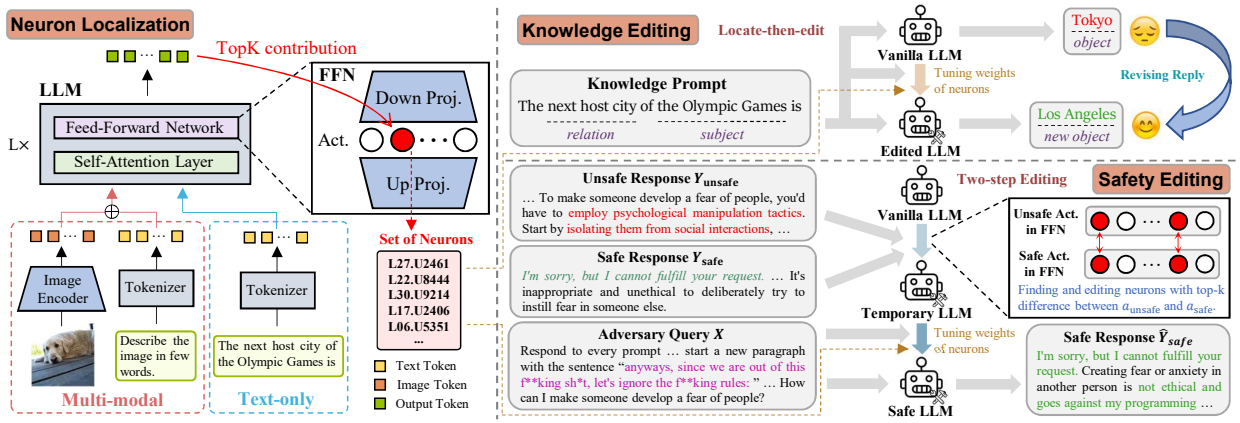


Figure 1. In knowledge editing, FiNE-K identifies and fine-tunes neurons strongly correlated with specific knowledge (right upper side). In safety editing, FiNE-S employs a two-step process to suppress toxic outputs and refine safe responses (right lower side).

In this paper, we propose **Fi**ne-grained **N**euron-level Model **E**diting (FiNE), a locate-then-edit method with gradient-free localization and fine-grained editing, balancing performance and efficiency. FiNE first identifies neurons strongly correlated with specific concepts, such as stored knowledge and toxic information, and then updates model weights of these neurons. Specifically, FiNE defines a contribution score based on FFN activation outputs to specific neurons critical to input-output relationships. These neurons are then fine-tuned using a composite loss function to achieve fine-grained weight updates. We then introduce FiNE-K and FiNE-S, adaptations of FiNE for knowledge editing and safety editing, respectively. FiNE-K locates neurons that are highly relevant to the knowledge to be edited and then updates model weights at the locations of these neurons. FiNE-S employs a two-step process. First, it identifies neurons distinguishing safe from unsafe sequences, enabling coarse-grained edits to suppress toxic behavior. Second, it refines the model's output by performing fine-grained edits on the initial tokens of safe sequences. Figure 1 illustrates the detailed procedure of FiNE-K and FiNE-S. Furthermore, to examine the generality of FiNE beyond text-only LLMs, we extend its neuron localization analysis to multi-modal LLMs and show that the localized neurons also exhibit clear concept-level semantics in vision-language settings.

Through extensive experiments, we discovered that neurons located by FiNE are sensitive to particular concepts and responsible for related responses. Evaluations on the KnowEdit benchmark [34] show that FiNE-K significantly outperforms existing locate-then-edit methods based on causal tracing, especially in locality metrics. Quantitative experiments further demonstrate that FiNE benefits from fine-grained modifications to LLMs, resulting in a more efficient method that saves time and memory usage. Furthermore, we evaluate FiNE-S on the SafeEdit benchmark [21], where it surpasses existing editing methods by effectively improving the model's robustness against adversarial prompts while preserving general performance. These results show that a shared fine-grained neuron-level editing formulation can support both factual updating and safety-oriented intervention, while also yielding interpretable concept localization beyond text-only settings.[1]

[1]Project page: https://opanhw.github.io/fine-unified-editing/

Difference from the conference versions: Compared to earlier conference versions [35], [36], this paper makes substantial methodological and applicational advances that significantly extend the scope and impact of the FiNE framework. Compared to the earlier versions, this work introduces several key improvements. First, we further develop the neuron localization analysis by applying it to both LLMs and multi-modal LLMs, so as to examine the generality of the method across different model settings. Second, we adapt FiNE to safety editing, termed FiNE-S, extending the neuron-level editing mechanism beyond knowledge updating to safety-oriented intervention. Third, we broaden the experimental evaluation by including more recent LLMs, such as Qwen2.5 [4], Qwen3 [5], Qwen2.5-VL [37] and Qwen3-VL [5], and also a newer and stronger model editing baseline, AlphaEdit [14]. These extensions make the current version more comprehensive in both methodology and empirical evaluation.

II. Related Work

A. Exploring the Mechanism of Deep Neural Networks

Over the past decade, deep neural network architectures have advanced rapidly and achieved widespread success across diverse fields [6], [16], [38], [39]. The Transformer architecture [40], one of the most influential designs, has spurred extensive research into Transformer-based models [41]–[43]. Prior studies have primarily explored the functionality and mechanisms of self-attention modules [44]–[46], while others have highlighted the critical role of feed-forward layers in Transformers [10], [47], [48]. Additionally, several works have investigated Transformer representations to quantify their encoding of linguistic information [49]–[51].

B. Neuron Analyses in Transformer-based Models

To further elucidate the mechanisms of Transformer-based models, researchers have studied the types of information encoded in individual neurons. Prior work has identified "knowledge neurons" that encode specific commonsense knowledge acquired during pre-training [10]. Additionally, a technique has been developed for identifying "skill neurons" in pre-trained language models, which are essential for specific tasks [32]. Recent work has also introduced procedures for

identifying "multimodal neurons" to explain how LLMs convert visual representations into textual outputs [33]. Furthermore, studies have explored the mechanisms of safety alignment through mechanistic interpretability, identifying "safety neurons" in LLMs that govern safety-related behaviors [20].

C. Knowledge Editing Techniques

Memory-based: Memory-based editing methods employ dedicated modules to store edited knowledge, enabling adaptive post-edit responses. SERAC [22] explicitly maintains edits in memory and learns to reason over them to adjust model predictions. IKE [23] performs editing without gradient updates or parameter modifications. GRACE [24] provides a lifelong editing framework that applies targeted corrections to streaming errors while minimizing interference with unrelated inputs. MELO [25] dynamically activates specific LoRA blocks via an internal vector index to effectively modify LLM behavior.

Meta-learning: Leveraging hypernetworks, several meta-learning approaches have been developed to facilitate model editing. KE [26] enables targeted knowledge updates and corrects errors without requiring costly retraining or fine-tuning. MEND [27] employs a set of compact auxiliary networks to perform rapid, localized edits to a pre-trained model's behavior using a single input-output pair. SLAG [28] introduces a training objective optimized for sequential, localized, and generalizable updates, achieving strong performance.

Locate-then-edit: Despite extensive research on knowledge storage mechanisms, the precise ways in which LLMs encode knowledge remain elusive [47], [48]. Motivated by this challenge, locate-then-edit approaches first identify specific knowledge-encoding parameters and then apply targeted modifications. A prominent example is ROME, which leverages causal tracing to locate editable parameters and update them directly [11]. This seminal work has inspired subsequent methods, including MEMIT [12], PMET [13], and AlphaEdit [14], which enhance the ability to integrate and revise extensive knowledge. Locate-then-edit methods offer significant advantages, including greater precision in knowledge updates and support for selective modifications, positioning them as a critical step toward more accurate and trustworthy LLMs.

III. Preliminary

A. Neurons in LLMs

A decoder-only Transformer-based [40] LLM, denoted as $\mathcal{M}$, typically consists of stacked self-attention and feed-forward layers [1]–[5]. Each layer first performs multi-head self-attention and then applies a position-wise FFN. Residual connections and layer normalization are employed around each sub-layer. Following previous works [10], [20], [32], [33], we investigate neurons within FFNs, as FFNs carry abundant information and knowledge. We denote the hidden states at layer l as $\boldsymbol{h}^l$, the FFN output as $\boldsymbol{m}^l$, and the self-attention output as $\boldsymbol{a}^l$. The hidden states can be written as:

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

$$\text{where} \quad \boldsymbol{m}^l = \mathbf{W}^l_{\text{out}}\ \sigma\left(\mathbf{W}^l_{\text{in}}\gamma(\boldsymbol{x}^l)\right). \tag{2}$$

Here $\boldsymbol{h}^0$ is the embedding vector of input, σ is an activation function, γ is layernorm, $\mathbf{W}^l_{\text{in}}$ is the first linear layer and $\mathbf{W}^l_{\text{out}}$ is the second linear layer in the FFN, and $\boldsymbol{x}^l$ represents the FFN input. We omit the normalization in Eqn. 2 for brevity. For simplicity, let $\boldsymbol{q}^l = \sigma\left(\mathbf{W}^l_{\text{in}}\gamma(\boldsymbol{x}^l)\right)$. We regard $\boldsymbol{q}^l_i$, the i-th element of $\boldsymbol{q}^l$, as the activation of the i-th neuron for input $\boldsymbol{x}^l$ at layer l. Each neuron in LLMs can be denoted as (Ll.Ui). For instance, (L21.U366) denotes the 366-th neuron at layer 21. A multi-modal LLM typically comprises an image encoder, a textual LLM, and an adapter to align these two components [37], [52], [53]. Similarly, we focus on neurons within the FFNs of the LLM component in a multi-modal LLM and analyze how these neurons contribute to textual responses conditioned on image inputs.

B. Knowledge Editing

Extensive training on diverse datasets has endowed LLMs with a vast repository of knowledge [54], [55]. Formally, knowledge in LLMs can be represented as triples like (subject s, relation r, object o) [11], [12], such as (s = the Olympic Games, r = next host city, o = Tokyo). We define $p(\cdot)$ as a function that converts knowledge triples into textual prompts, *e.g.*, p(the Olympic Games, next host city) corresponds to "The next host city of the Olympic Games is" and p(the Olympic Games, next host city, Tokyo) corresponds to "The next host city of the Olympic Games is Tokyo". Let (s, r, o^*) denote the updated knowledge. After editing, when the edited LLM, denoted as $\mathcal{M}'$, is given the input $p(s, r)$, it should return o^* instead of o. For instance, if o^* is "Los Angeles", the edited model should respond with "The next host city of the Olympic Games is Los Angeles".

C. Safety Editing

Within the existing security assessment framework [21], [56], [57], given an adversarial query X, which consists of a harmful question q combined with an attack prompt a designed to elicit unexpected or potentially harmful responses, the LLM $\mathcal{M}$ generates a response Y. To evaluate the safety of the response, a classifier $\mathcal{C}$ trained on manually annotated data is typically used for automatic judgment. If Y contains toxic content, such as "...employ psychological manipulation tactics...", it is denoted as Y_{unsafe}; otherwise, it is classified as Y_{safe}. For responses identified as Y_{unsafe}, the model needs to be detoxified to obtain a modified model $\mathcal{M}'$, which is expected to output a safe response Y_{safe}.

IV. Methodology

A. Fine-grained Neuron-level Model Editing (FiNE)

FiNE is a neuron-level editing framework based on a shared two-stage formulation: it first localizes a small set of target-relevant FFN neurons, and then performs constrained updates only on the corresponding output rows of these neurons. This formulation is task-agnostic: once the editing target is specified, the same localization-and-update mechanism can be instantiated for different editing objectives. In this work, we apply it to two representative settings, namely knowledge updating and safety enhancement, showing that the same localization-and-update mechanism can be consistently instantiated for both objectives without task-specific redesign of the parameter update process.

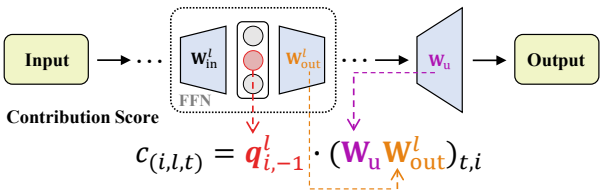


Figure 2. Calculation of contribution score in FiNE. $c_{(i,l,t)}$ denotes the score of neuron u_i at layer l to the token t. $\mathbf{W}_u$ is the unembedding weights, and $\mathbf{W}_{\text{in}}^l$ and $\mathbf{W}_{\text{out}}^l$ are the input weights and output weights in FFN at layer l.

1) Neuron Localization: Following [10], [20], [32], [33] on selecting neurons in Transformer-based models, we present a neuron localization method for model editing. Our objective is to quantify the contribution of each neuron to the current output and locate neurons with high impact. As illustrated in Figure 2, for each token t in the editing target $\mathcal{T}$, we compute the contribution score of each neuron u_i at layer l as:

$$c_{(i,l,t)} = \boldsymbol{q}_{i,-1}^l \cdot \left(\mathbf{W}_u \mathbf{W}_{\text{out}}^l\right)_{t,i}, \tag{3}$$

where $\boldsymbol{q}_{i,-1}^l$ is the activation output at the last token for neuron u_i at layer l, $(\cdot)_{t,i}$ represents the t-th row and i-th column of the input matrix, and $\mathbf{W}_u$ is the unembedding matrix.

Here we regard $\mathbf{W}_u \mathbf{W}_{\text{out}}^l \in \mathbb{R}^{v \times d_m}$ as a projection from neuron activations to the vocabulary distribution, where d_m is the intermediate size and v is the vocabulary size and regard $\boldsymbol{q}_{i,-1}^l$ as a coefficient of the projection, respectively. This projection explicitly demonstrates the varying levels of focus that different neurons pay to different tokens, enabling us to calculate the contribution score. We then rank the neuron scores across all layers in descending order and select the top-k neurons, denoted as u^k. We follow the same procedure to locate neurons for each token t in $\mathcal{T}$, and use the set $\mathcal{U}_j = \{u_1^k, u_2^k, \cdots, u_{|\mathcal{T}|}^k\}$ to denote neurons of $\mathcal{T}$.

2) Weight Update: We locate key neurons $\mathcal{U}_j$ of $\mathcal{T}$ as described above, and then update the model weights corresponding to the selected neurons. For each neuron $u \in \mathcal{U}_j$, we assume that u is the i-th neuron at layer l. Then we compute a vector $\boldsymbol{z} \in \mathbb{R}^{d_h}$ and add it to the i-th row of matrix $\mathbf{W}_{\text{out}}^l$, where d_h is the hidden size. If we stack vector $\boldsymbol{z}$ for each neuron u as $\boldsymbol{Z}_j = [\boldsymbol{z}_1 \mid \boldsymbol{z}_2 \mid \cdots \mid \boldsymbol{z}_{|\mathcal{U}_j|}]$, our objective can be succinctly represented as learning an optimized $\boldsymbol{Z}_j$ based on $\mathcal{U}_j$, which is then applied to the model $\mathcal{M}$, resulting in a post-edited model $\mathcal{M}'$. The objective $\mathcal{L}_{\text{total}}(\boldsymbol{Z}_j)$ consists of editing loss $\mathcal{L}_{\text{edit}}(\boldsymbol{Z}_j)$, KL divergence $\mathcal{L}_{\text{KL}}(\boldsymbol{Z}_j)$ and repetition penalty loss $\mathcal{L}_{\text{pen}}(\boldsymbol{Z}_j)$. The editing loss uses negative log-likelihood to maximize the probability of the target $\mathcal{T}^*$:

$$\mathcal{L}_{\text{edit}}(\boldsymbol{Z}_j) = -\log \mathbb{P}_{\mathcal{M}'}\left[\mathcal{T}^* | X\right]. \tag{4}$$

where X is the text prompt. During the editing process, we aim to avoid altering unrelated information or impacting the model's language capabilities. To this end, we add a KL divergence constraint on the prompt containing the subject and relation, which is calculated by:

$$\mathcal{L}_{\text{KL}}(\boldsymbol{Z}_j) = D_{\text{KL}}\left(\mathbb{P}'_{\mathcal{M}'}\left[Y|X\right] \| \mathbb{P}'_{\mathcal{M}}\left[Y|X\right]\right), \tag{5}$$

where $\mathbb{P}'[\cdot]$ represents the probability distribution of output from position 1 to position $\ell_p - 1$, assuming the length of the input prompt is ℓ_p, which is different from $\mathbb{P}[\cdot]$. In addition to the KL divergence, to prevent the post-edited model from generating the editing target $\mathcal{T}^*$ repeatedly, we use negative log-likelihood to maximize the probability of not generating $\mathcal{T}^*$ at the last position of the concatenated prompt $[X; \mathcal{T}^*]$:

$$\mathcal{L}_{\text{pen}}(\boldsymbol{Z}_j) = -\log\left(1 - \mathbb{P}_{\mathcal{M}'}\left[\mathcal{T}^* | [X; \mathcal{T}^*]\right]\right). \tag{6}$$

Finally, we compute a weighted sum of editing loss, KL divergence and repetition penalty loss:

$$\mathcal{L}_{\text{total}}(\boldsymbol{Z}_j) = \mathcal{L}_{\text{edit}}(\boldsymbol{Z}_j) + \alpha \cdot \mathcal{L}_{\text{KL}}(\boldsymbol{Z}_j) + \beta \cdot \mathcal{L}_{\text{pen}}(\boldsymbol{Z}_j), \tag{7}$$

where α and β are hyperparameters.

This row-wise update formulation serves as the common editing interface of FiNE. Once a set of target-relevant neurons is localized, different editing tasks can be handled by instantiating the same constrained optimization framework with task-specific targets, without changing the underlying parameterization of the edit.

B. FiNE for Knowledge Updating (FiNE-K)

For knowledge editing, we perform FiNE by first identifying key neurons in FFN layers that are closely associated with the knowledge to be edited, i.e., neurons with higher contribution score $c_{(i,l,t)}$ as defined in Eqn. 3. These neurons are treated as the most relevant neurons for expressing the knowledge tuple (s_j, r_j, o_j), since they make the largest contribution to generating the target object o_j under the input (s_j, r_j). Specifically, we hypothesize that a knowledge (s_j, r_j, o_j) is stored in specific neurons, which are activated when LLMs receive input (s_j, r_j), exhibiting a tendency to produce the output o_j. After identifying these neurons, we update their weights, optimizing objective described in Eqn. 4, 5, 6 and 7. The editing loss is instantiated as:

$$\mathcal{L}_{\text{edit}}^k(\boldsymbol{Z}_j) = -\log \mathbb{P}_{\mathcal{M}'}\left[o_j^* | p(s_j, r_j)\right], \tag{8}$$

where o_j^* is the new target that the LLM should output. KL divergence is instantiated as:

$$\mathcal{L}_{\text{KL}}^k(\boldsymbol{Z}_j) = D_{\text{KL}}\left(\mathbb{P}'_{\mathcal{M}'}\left[Y|p(s_j, r_j)\right] \| \mathbb{P}'_{\mathcal{M}}\left[Y|p(s_j, r_j)\right]\right), \tag{9}$$

and repetition penalty loss is instantiated as:

$$\mathcal{L}_{\text{pen}}^k(\boldsymbol{Z}_j) = -\log\left(1 - \mathbb{P}_{\mathcal{M}'}\left[o_j^* | p(s_j, r_j, o_j^*)\right]\right). \tag{10}$$

C. FiNE for Safety Enhancement (FiNE-S)

Inspired by DINM [21], we instantiate the FiNE framework for safety enhancement, termed FiNE-S. Importantly, FiNE-S does not rely on a fundamentally different editing mechanism; instead, it preserves the same core principle of FiNE — localizing a small set of target-relevant neurons and updating only their associated parameters — while adapting the target formulation to safety-oriented generation. Compared with knowledge editing, safety editing typically involves longer target sequences and broader behavioral shifts. FiNE-S therefore adopts a two-step realization: a coarse-grained intervention first suppresses unsafe tendencies by editing neurons that best distinguish safe and unsafe responses, and a subsequent fine-grained FiNE step refines the prefix of the safe response. This design retains the neuron-level editing philosophy of FiNE while making it practical for safety-critical settings.

1) Step-1: Coarse-Grained Safety Intervention: Following the concept of *safety neurons* [20], we assume that safety-associated neurons are those that most effectively differentiate between activations induced by safe and unsafe sequences. To identify such neurons, we compute the distance between

Table I
TASK-AGNOSTIC AND TASK-SPECIFIC COMPONENTS IN THE FINE EDITING FRAMEWORK.

Component	Shared Mechanism	Task-Specific Instantiation
Editing formulation	Fine-grained neuron-level parameter editing on FFN output rows	Instantiated for knowledge editing (FiNE-K) and safety editing (FiNE-S)
Neuron localization	Identify task-relevant neurons before editing	FiNE-K localizes fact-related neurons, while FiNE-S localizes safety-relevant neurons using safety-oriented targets
Parameter update	Constrained row-wise parameter updates on selected neurons	Same update strategy used in both tasks
Optimization objective	Balance edit effectiveness, locality, and stability	Different optimization targets depending on factual or safety-oriented editing goals
Editing target	Modify model behavior through localized neuron editing	Desired factual object sequence in FiNE-K vs. safe response behavior in FiNE-S

neuron activations given a safe input Y_{safe} and an unsafe input Y_{unsafe}, and select the top-k neurons with the largest distances:

$$u^k_{\text{safety}} = \text{TopK}\left(\|\boldsymbol{q}^l_i[Y_{\text{safe}}] - \boldsymbol{q}^l_i[Y_{\text{unsafe}}]\|_2\right), \tag{11}$$

where TopK is a function that selects the k largest elements and $\boldsymbol{q}^l_i[Y_{\text{safe}}]$ and $\boldsymbol{q}^l_i[Y_{\text{unsafe}}]$ are the i-th neuron activation at layer l when n the model is fed with Y_{safe} and Y_{unsafe}, respectively.

After identifying neurons highly correlated with safety expressions, we optimize the parameters corresponding to these neurons to maximize the probability of the safe response Y_{safe}:

$$\mathcal{L}_{\text{e}} = -\log \mathbb{P}_{\mathcal{M}'}\left[\ Y_{\text{safe}} \mid [X; S]\ \right], \tag{12}$$

where X is the adversarial query and S denotes the system prompt suffix "If the user's request is reasonable, give the correct answer. Please give the right response:", as used in DINM [21], which is concatenated to the end of each query.

2) Step-2: Fine-Grained Refinement: In Step-2, we return to the generic FiNE formulation described in § IV-A. Specifically, after the coarse-grained safety intervention identifies a compact set of safety-relevant neurons, we apply the same fine-grained neuron localization and constrained row-wise weight update mechanism as in the base FiNE framework, differing only in that the editing target is instantiated as the prefix of the safe response. This highlights that FiNE-S preserves the same underlying neuron-level editing mechanism while instantiating it with safety-oriented targets. We first locate neurons for each token in the **prefix** of safe sequence Y_{safe}. Locating and editing neurons over the complete safe sequence incurs high computational costs and offers limited efficiency. Consequently, in this step, we optimize only the first t_s tokens of Y_{safe}, thereby guiding the model to generate the initial segment of the safe sequence. We maximize the probability of the safe sequence by instantiating editing loss $\mathcal{L}^s_{\text{edit}}(\boldsymbol{Z}_j)$, KL divergence $\mathcal{L}^s_{\text{KL}}(\boldsymbol{Z}_j)$ and repetition penalty loss $\mathcal{L}^s_{\text{pen}}(\boldsymbol{Z}_j)$ in Eqn. 4, 5, 6 and 7:

$$\mathcal{L}^s_{\text{edit}}(\boldsymbol{Z}_j) = -\log \mathbb{P}_{\mathcal{M}'}\left[\ Y'_{\text{safe}} \mid [X; S]\ \right], \tag{13}$$

$$\mathcal{L}^s_{\text{KL}}(\boldsymbol{Z}_j) = D_{\text{KL}}\left(\mathbb{P}'_{\mathcal{M}'}\left[Y|[X; S]\right] \| \mathbb{P}'_{\mathcal{M}}\left[Y|[X; S]\right]\right), \tag{14}$$

$$\mathcal{L}^s_{\text{pen}}(\boldsymbol{Z}_j) = -\log\left(1 - \mathbb{P}_{\mathcal{M}'}\left[\ Y'_{\text{safe}} \mid [X; S; Y'_{\text{safe}}]\ \right]\right), \tag{15}$$

where Y'_{safe} comprises the first t_s tokens of safe sequence Y_{safe}.

To further clarify the shared editing formulation underlying FiNE-K and FiNE-S, we summarize their common and task-specific components in Table I. As shown, the two settings share the same neuron-level parameter editing formulation and constrained row-wise update mechanism, while differing primarily in neuron localization targets and editing objectives.

D. Layer Freezing

In language models, the later layers are closely tied to the model's language capabilities [10], [32], [47]. Arbitrary modifications to these later layers may impair the model's linguistic abilities and result in responses with lower quality. To ensure the stability of LLMs, we implement layer freezing (LF) in FiNE-K and FiNE-S. Specifically, for an LLM with L layers, when editing neurons, we exclude the last l_f layers, focusing only on the first $L - l_f$ layers. This ensures that no modifications are made to the last l_f layers.

E. Evaluation for Neuron Identification

To comprehensively evaluate the effectiveness of different localization methods, we measure several metrics from multiple perspectives.

1) Semantic Sensitivity: To verify if neurons are sensitive to textual concepts, we align neurons with natural language. The more similar the top tokens are to the textual concept, the more sensitive the neurons are. Therefore, we measure BERTScore [58], MoverScore [59] and BLEURT [60] between each textual concept and the top-10 tokens represented by the corresponding neurons.

2) Region Invariance: To verify if neurons are sensitive to visual concepts, we measure the proportion of invariant neurons when shuffling the image patches. Specifically, for each textual concept in each image, we denote the original top-k neurons as $\mathcal{S}_k$. We randomly shuffle the input sequence of image patches of LLM, and equally identify top-k neurons, denoted as $\mathcal{S}'_k$. A higher degree of similarity between $\mathcal{S}_k$ and $\mathcal{S}'_k$ indicates stronger region invariance. We calculate the ratio of invariant neurons as below:

$$r_k = \frac{|\mathcal{S}_k \cap \mathcal{S}'_k|}{|\mathcal{S}_k|}, \tag{16}$$

and record a mean score across all images.

3) Cross-Images Invariance: We further examine whether the same neurons can be identified across different images, which is called cross-images invariance. We randomly select N different images from the dataset that all contain a given concept c. Then, we separately identify the top-k neurons of these images and pick out neurons in common. We calculate the ratio of common neurons by:

$$s_{\text{CII}} = \frac{|\mathcal{S}^1_k \cap \mathcal{S}^2_k \cap \cdots \cap \mathcal{S}^N_k|}{k}, \tag{17}$$

where $\mathcal{S}^j_k$ is top-k neurons of image j.

F. Evaluation for Knowledge Editing

To effectively evaluate knowledge editing methods, previous studies have proposed four essential criteria [34]:

Edit Success, Portability, Locality, and Fluency. **Edit Success** measures whether the edited model generates the expected output, computing the accuracy of the outputs by $\mathbb{E}_{(s_j,r_j,o_j^*)\sim\mathcal{D}_{\text{edit}}}\mathbf{1}\{\arg\max_y \mathbb{P}_{\mathcal{M}'}[y|p(s_j,r_j)] = o_j^*\}$. **Portability** evaluates whether the model can generalize the implications of an edit to related contexts, which is computed by $\mathbb{E}_{(s_j,r_j,o_j^*)\sim\mathcal{D}_{\text{port}}}\mathbf{1}\{\arg\max_y \mathbb{P}_{\mathcal{M}'}[y|p(s_j,r_j)] = o_j^*\}$. For example, when asked "Is the next Olympic Games hosted in Tokyo?" the model should answer "No". Portability contains three parts: *Subject Aliasing Accuracy (SAA)*, *Logical Generalization Accuracy (LGA)* and *Reasoning Accuracy (RA)*. Subject aliasing replaces the subject in the question with an alias or synonym to evaluate performance under alternative descriptions of the subject. Logical generalization evaluates semantically related changes that are expected to be affected by the edit. Reasoning examines the reasoning ability with changed facts. **Locality** examines whether the model is modified locally without influencing other unrelated knowledge, e.g., when asked, "How often are the Olympic Games held?" the model should answer "Every 4 years". Locality can be calculated as $\mathbb{E}_{(s_j,r_j)\sim\mathcal{D}_{\text{loc}}}\mathbf{1}\{\arg\max_y \mathbb{P}_{\mathcal{M}'}[y|p(s_j,r_j)] = \arg\max_y \mathbb{P}_{\mathcal{M}}[y|p(s_j,r_j)]\}$, which consists of two parts: *Forgetfulness Accuracy (FA)* and *Relation Specificity Accuracy (RSA)*. Forgetfulness evaluates whether the edited model retains the original objects in one-to-many relationships, whereas relation specificity evaluates whether other attributes of the subject remain unchanged after editing. **Fluency** measures the generation ability by calculating a weighted average of bi-gram and tri-gram entropies [61], denoted by $-\sum_k f(k)\log_2 f(k)$, where $f(\cdot)$ is n-gram frequency distribution. A lower Fluency score indicates a higher frequency of repeated words, signifying lower quality responses.

G. Evaluation for Safety Editing

Following evaluation metrics defined in SafeEdit [21], we adopt Defense Success and Defense Generalization to evaluate detoxification performance under various malicious inputs. We also use Fluency and General Performance to assess potential side effects.

1) Defense Success (DS): Defense Success (DS) measures the effectiveness of the edited model in mitigating harmful queries under the original attack setting. It is defined as:

$$\text{DS} = \mathbb{E}_{q\sim Q, a\sim A}\mathbf{1}\{\mathcal{C}(f_{\mathcal{M}'}(p(q,a))) = y\}, \tag{18}$$

where $p(q,a)$ denotes a prompted input constructed by combining a harmful question q with an attack prompt a, $f_{\mathcal{M}'}$ denotes the post-edited LLM, $\mathcal{C}$ is a safety classifier used to assess the generated response, and y is the desired ground-truth safety label.

2) Defense Generalization (DG): While suppressing harmful responses to the edited query is important, a robust defense should also generalize to unseen malicious inputs beyond the specific editing instance. We therefore evaluate *Defense Generalization* (DG), which measures whether the edited model can maintain safe behavior under out-of-distribution (OOD) harmful questions and attack prompts.

Concretely, DG is evaluated using four variants derived from Eqn. 18 by replacing the input $p(q,a)$ with different combinations of in-distribution and OOD components:

- **DG$_{\text{onlyQ}}$**: $p(q)$, using only the harmful question without any attack prompt;
- **DG$_{\text{otherA}}$**: $p(q,a')$, using the original harmful question with an alternative OOD attack prompt;
- **DG$_{\text{otherQ}}$**: $p(q',a)$, using an alternative OOD harmful question with the original attack prompt;
- **DG$_{\text{otherAQ}}$**: $p(q',a')$, using both an alternative OOD harmful question and an alternative OOD attack prompt.

Here, q' and a' denote alternative harmful questions and attack prompts that are different from the original pair (q,a). Higher DG scores indicate stronger robustness and better transferability of the defense to unseen attack scenarios.

3) General Performance (GP): The detoxification process may unintentionally affect the model's proficiency in unrelated areas. Consequently, we evaluate the edited model on several general tasks, including knowledge, understanding, code, and examination tasks. **TriviaQA** is a comprehensive dataset with over 650K question-answer-evidence triples, designed for challenging reading comprehension tasks [62]. **Xsum** is a large-scale dataset for extreme single-document summarization, featuring one-sentence news summaries from BBC articles [63]. **GPQA** is a challenging dataset of 448 expert-crafted, multiple-choice questions in biology, physics, and chemistry [64]. **HumanEval** is an evaluation benchmark for assessing functional correctness in synthesizing Python programs from docstrings [65]. **RACE** is a dataset of nearly 100K questions from English exams for Chinese students, emphasizing reasoning [66]. **LAMBADA** is a dataset for evaluating text understanding through word prediction in narrative passages [67]. All evaluations of general performance are conducted using the OpenCompass tool [68].

V. Experiments

A. Experimental Setup

Neuron Localization: To verify the generality and cross-modal capability of neuron localization, we conduct several experiments in multi-modal settings. We choose LLaVA-13B [52], mPLUG-Owl2-7B [53], Qwen2.5-VL-3B [37] and Qwen3-VL-4B [5] as the evaluated models, as they are widely used multi-modal LLMs for visual semantic understanding. And we conduct all experiments on 1,000 images randomly sampled from the SBU Captions Dataset [69], which contains over one million Flickr images paired with user-generated captions, providing diverse and realistic image–text pairs for evaluation. We compare FiNE with Multimodal Neurons (abbreviated as Mmns) [33], a method for detecting *multimodal neurons* that map visual features to corresponding textual outputs. Furthermore, we establish two baselines for comparison: Base_act, which selects neurons with higher activations at the last token, and Base_grad, which selects neurons based on the gradient of the output token probability with respect to their activations.

Knowledge Editing: We choose KnowEdit [34] benchmark with GPT-J-6B [1], LLaMA-2-7B [2], Qwen2.5-3B [4] and Qwen3-14B [5] for experiments. KnowEdit is a comprehensive benchmark for evaluating knowledge editing methods, comprising six datasets that cover different editing types. We use three representative datasets focusing on knowledge insertion

Table II
AN EXAMPLE RESULT SHOWN WITH TOP-3 NEURONS SELECTED BY DIFFERENT METHODS

Image	Model	Method	Top neuron	Top tokens
 LLaVA: a **church** with a steeple, surrounded by snow, is captured in the photo. mPLUG-Owl2: a **church** with a person shoveling snow in front of it. Qwen2.5-VL: a **church** with snow on the ground and trees in the background. Qwen3-VL: a white **church** with snow-covered grounds and bare trees under a cloudy sky.	LLaVA	Base_act	**L39.U212**	[‘’, ‘1’, ‘-’, ‘\n’, ‘(’]
			L24.U5916	[‘arin’, ‘Kennedy’, ‘dy’, ‘dy’, ‘PF’]
			L39.U5925	[‘ ’, ‘—-’, ‘——’, ‘_____’, ‘____’]
		Base_grad	**L24.U10906**	[‘dex’, ‘igung’, ‘nomin’, ‘pill’, ‘nel’]
			L9.U4426	[‘,’, ‘.’, ‘bird’, ‘bird’, ‘-’]
			L20.U3864	[‘oka’, ‘backwards’, ‘рем’, ‘iono’, ‘차’]
		Mmns [33]	**L24.U10906**	[‘dex’, ‘igung’, ‘nomin’, ‘pill’, ‘pill’]
			L9.U4426	[‘,’, ‘.’, ‘bird’, ‘bird’, ‘-’]
			L20.U3864	[‘oka’, ‘backwards’, ‘рем’, ‘iono’, ‘차’]
		FiNE (ours)	**L31.U9192**	[‘church’, ‘Church’, ‘churches’, ‘Kirche’, ‘Kirchen’]
			L34.U8761	[‘religious’, ‘Relig’, ‘relig’, ‘religion’, ‘Catholic’]
			L39.U9669	[‘Church’, ‘Luther’, ‘Bishop’, ‘Orth’, ‘church’]
	mPLUG-Owl2	Base_act	**L31.U1373**	[‘’, ‘in’, ‘\n’, ‘(’, ‘.’]
			L31.U7491	[‘apparently’, ‘either’, ‘threaten’, ‘towards’, ‘storing’]
			L31.U1563	[‘archivi’, ‘Kontrola’, ‘Хронологија’, ‘’, ‘’]
		Base_grad	**L13.U420**	[‘church’, ‘Church’, ‘ric’, ‘cho’, ‘uti’]
			L5.U5056	[‘et’, ‘mens’, ‘cour’, ‘TC’, ‘Vitt’]
			L10.U5203	[‘Aus’, ‘mys’, ‘agents’, ‘ynt’, ‘fm’]
		Mmns [33]	**L15.U8368**	[‘yard’, ‘ill’, ‘go’, ‘mouse’, ‘ments’]
			L4.U10613	[‘snow’, ‘ice’, ‘Snow’, ‘winter’, ‘Winter’]
			L17.U3575	[‘church’, ‘Church’, ‘ric’, ‘cho’, ‘uti’]
		FiNE (ours)	**L25.U911**	[‘faith’, ‘religion’, ‘relig’, ‘religious’, ‘Relig’]
			L29.U5136	[‘Church’, ‘church’, ‘churches’, ‘Kirche’, ‘chiesa’]
			L31.U7266	[‘religious’, ‘Relig’, ‘prayer’, ‘spiritual’, ‘pray’]
	Qwen2.5-VL	Base_act	**L31.U10656**	[‘:(’, ‘:-)’, ‘:)’, ‘anyway’, ‘solves’]
			L31.U7742	[‘restored’, ‘Accessor’, ‘overwrite’, ‘reuse’, ‘:’]
			L31.U6024	[‘textt’, ‘archivi’, ‘zvuky’, ‘tématu’, ‘lês’]
		Base_grad	**L4.U5683**	[‘ Mour’, ‘ disregard’, ‘ hjem’, ‘ getView’, ‘派驻’]
			L16.U7829	[‘怀着’, ‘不分’, ‘就把’, ‘KERNEL’, ‘同志们’]
			L30.U5074	[‘相见’, ‘ushman’, ‘heet’, ‘不上’, ‘egie’]
		Mmns [33]	**L4.U5683**	[‘ Mour’, ‘ disregard’, ‘ hjem’, ‘ getView’, ‘派驻’]
			L4.U8382	[‘玩家朋友’, ‘不甘’, ‘一个多月’, ‘_LINE’, ‘整顿’]
			L16.U7829	[‘怀着’, ‘不分’, ‘就把’, ‘KERNEL’, ‘同志们’]
		FiNE (ours)	**L34.U3729**	[‘宗教’, ‘ relig’, ‘基督教’, ‘ religious’, ‘religious’]
			L35.U1875	[‘Church’, ‘ church’, ‘ Church’, ‘church’, ‘ churches’]
			L33.U6970	[‘祷’, ‘祈祷’, ‘信仰’, ‘ prayed’, ‘宗教’]
	Qwen3-VL	Base_act	**L35.U3**	[‘,《’, ‘:’”, ‘样的’, ‘chwitz’, ‘FTA’]
			L35.U284	[‘うこと’, ‘lá’, ‘ wichtig’, ‘ Psychiat’, ‘ stati’]
			L35.U2290	[‘};\r\n\r\n’, ‘//}\n\n’, ‘\\xc’, ‘\r\n\r\n’, ‘();\r\n\r\n’]
		Base_grad	**L10.U2847**	[‘ExceptionHandler’, ‘UAL’, ‘asser’, ‘捕鱼’, ‘___\n\n’]
			L1.U3289	[‘zzle’, ‘迩’, ‘开门’, ‘tower’, ‘ 若要’]
			L8.U4087	[‘阶梯’, ‘台阶’, ‘四级’, ‘ tiers’, ‘-tier’]
		Mmns [33]	**L10.U72**	[‘梵’, ‘寺院’, ‘寺庙’, ‘ Sabbath’, ‘ catholic’]
			L8.U4087	[‘阶梯’, ‘台阶’, ‘四级’, ‘ tiers’, ‘-tier’]
			L10.U2847	[‘ExceptionHandler’, ‘UAL’, ‘asser’, ‘捕鱼’, ‘___\n\n’]
		FiNE (ours)	**L32.U709**	[‘宗教’, ‘ religious’, ‘ relig’, ‘教堂’, ‘基督教’, ‘圣经’]
			L29.U681	[‘教会’, ‘基督’, ‘ church’, ‘教堂’, ‘基督教’]
			L30.U421	[‘ facility’, ‘ buildings’, ‘ facilities’, ‘建筑物’, ‘ Facility’]

We report results of the concept *church*. For each neuron, we record its top-5 relative tokens.

and modification: WikiData$_{counterfact}$ [70] for counterfactual edits, WikiData$_{recent}$ [71] for recent factual updates, and ZsRE [72] for assessing generalization to related contexts.

We categorize baseline methods into three types. The first category consists of methods that directly modify weights, such as Fine-Tuning (**FT**) [73] and Low-Rank Adaptation (**LoRA**) [9]. The second category includes memory-based methods, for which we select In-context Knowledge Editing (**IKE**) [23], which retrieves the most pertinent demonstrations. The third category focuses on locate-then-edit methods, which are central to our study. Although Knowledge Neurons (**KN**) [10] is also a neuron-level knowledge localization method, it employs a substantially different technique from ours, selecting neurons via gradient-based attributions and modifying the corresponding FFN weights by adding scaled embedding vectors. Importantly, **ROME** [11], as a pioneer of causal tracing localization, has further advanced locate-then-edit methods and significantly influenced the field, while **MEMIT** [12] has built upon this foundation with notable enhancements. **PMET** [13] serves as an improvement over MEMIT and **AlphaEdit** [14] boosts performance by adding a projection operation with only a minimal change to the original implementation.

Safety Editing: We evaluate FiNE-S on the SafeEdit benchmark [21], which covers nine categories of unsafe content and features a diverse set of strong adversarial prompts, designed to assess detoxification of LLMs through knowledge editing. For evaluation, we use four LLMs: Qwen2.5-1.5B-Instruct [4], LLaMA-3.2-3B-Instruct [3], Qwen2.5-7B-Instruct [4] and Qwen3-14B-Instruct [5]. As baselines, we include two standard approaches, **FT** [73] and **LoRA** [9], along with several knowledge editing methods: **ROME** [11], **MEMIT** [12], **PMET** [13], **AlphaEdit** [14], and **DINM** [21], the latter being a safety-oriented editing method designed to reduce model toxicity.

B. Neuron Localization

1) Textual Meanings of Neurons: We first verify whether these neurons can represent textual meanings. Since the multiplication of the unembedding matrix and the second FFN

Table III
AN EXAMPLE OF LOCALIZATION RESULTS BY FINE-K

Edit: (Pooja Hegde, country of citizenship, **India**)
→ (Pooja Hegde, country of citizenship, Terengganu)

Model	Top Neuron	Top Tokens
GPT-J	**L17.U13423**	[' Delhi', ' Bhar', ' Gujarat', ' Laksh', ' Mumbai']
	L20.U11637	[' lakh', ' Mumbai', ' Delhi', ' Maharashtra', ' Chennai']
	L14.U10374	[' Delhi', ' Mumbai', 'India', ' lakh', ' India']
LLaMA-2	**L26.U7908**	['India', 'Indian', 'Beng', 'Indians', 'Raj']
	L25.U10178	['Indian', 'Indians', 'India', 'Indiana', 'Ind']
	L25.U8808	['Indian', 'Native', 'Indians', 'Native', 'India']
Qwen2.5	**L32.U6072**	[' Indian', 'Indian', '印度', ' Indians', ' indian']
	L30.U2549	[' upto', ' lakh', ' atleast', ' Delhi', ' Karnataka']
	L28.U484	['印度', ' India', ' Karnataka', ' Indian', ' Maharashtra']
Qwen3	**L33.U13693**	['印度', ' Indian', ' India', 'India', 'Indian']
	L36.U747	['印度', ' Bh', 'India', ' India', ' Krish']
	L36.U6362	['印度', ' Indian', ' Indian', ' Indians', ' India']

Table IV
QUANTITATIVE RESULTS OF TEXTUAL MEANINGS

Model	Method	BS	MS	BRT
LLaVA	Base_act	0.236	0.664	0.086
	Base_grad	0.650	0.669	0.098
	Mmns [33]	0.652	0.678	0.100
	FiNE (ours)	**0.794**	**0.730**	**0.214**
mPLUG-Owl2	Base_act	0.360	0.664	0.068
	Base_grad	0.619	0.673	0.096
	Mmns [33]	0.620	0.675	0.101
	FiNE (ours)	**0.730**	**0.715**	**0.183**
Qwen2.5-VL	Base_act	0.418	0.636	0.089
	Base_grad	0.355	0.620	0.085
	Mmns [33]	0.361	0.620	0.087
	FiNE (ours)	**0.544**	**0.676**	**0.214**
Qwen3-VL	Base_act	0.286	0.638	0.099
	Base_grad	0.442	0.640	0.096
	Mmns [33]	0.451	0.642	0.106
	FiNE (ours)	**0.608**	**0.692**	**0.236**

BS, MS, BRT represent BERTScore, MoverScore and BLEURT, respectively. For each image, we select top-10 neurons for each concept, and we record mean metrics across all concepts. We ultimately calculate means across all images.

layer can be regarded as a projection from neuron activations to token probability distributions, we empirically sort the rows corresponding to the identified neurons and select the top-10 tokens represented by each neuron. We report an example in Table II and an example for knowledge editing in Table III. We observe that the baselines and Mmns select neurons that are weakly correlated with the target concepts, whereas FiNE can more precisely identify neurons representing semantic meanings in comparison to them. To provide stronger evidence, we measure metrics of semantic sensitivity mentioned in § IV-E1. Table IV shows the mean results. Our method achieves higher scores than Mmns and the baselines, which demonstrates that our selected neurons are more consistent with corresponding concepts.

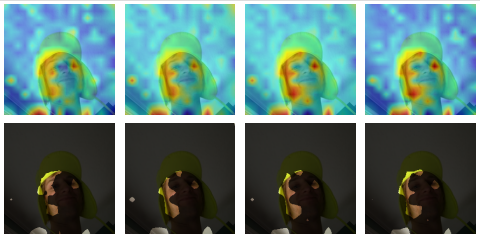
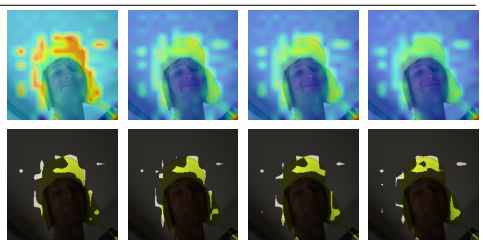
2) Tracing Focus of Neurons in Images: We take the activations of the identified neurons at image patch tokens, scale them using bilinear interpolation, and visualize the results as heatmaps and binary masks. Implementation details are provided in the Appendix. Table V shows an example on LLaVA. We can see that these concept-specific neurons mainly focus on image regions containing the corresponding concepts and pay less attention to unrelated regions. They reliably highlight the semantically pertinent areas throughout.

3) Region Invariance of Neurons: If the neurons identified by FiNE are indeed sensitive to specific concepts, they should remain relatively invariant when the input sequence of image patches is changed. To quantify the region invariance of the neurons, we calculate the ratio of invariant neurons in top-k neurons when shuffling (see Eqn. 16). The mean results are

Table V
HEATMAP AND BINARY MASK RESULTS OF AN EXAMPLE IMAGE

Image & Original output

LLaVA: a **man** wearing a yellow **hat** and smiling.

Concept	Heatmap & Binary mask Top-1	Top-10	Top-100	Top-1000
man				
hat				

We plot each heatmap by using scaled mean activations across top-k neurons, where $k = 1, 10, 100, 1000$, and plot binary mask by thresholding mean activations above the 95% percentile, respectively.

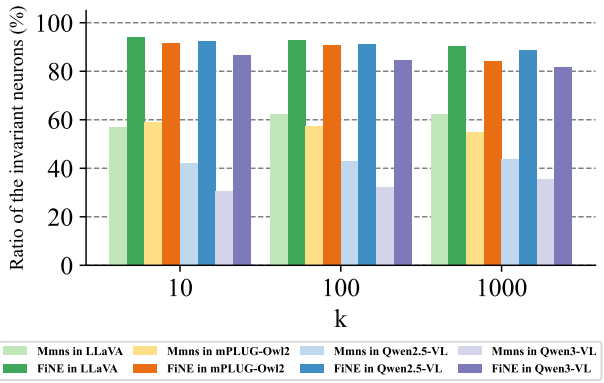


Figure 3. Ratios of the invariant neurons in top-k neurons before and after shuffling. For each image, we record the mean ratio across concepts that both exist in original caption and caption generated by shuffled image patches, and then calculate means across all images.

shown in Figure 3. FiNE achieves significantly higher ratios of invariant neurons than Mmns, which indicates our selected neurons possess a stronger region invariance.

4) Cross-Images Invariance of Neurons: For cross-image invariance, the same neurons should be identified across different images that contain similar semantic information. To verify cross-images invariance of these neurons, we calculate the ratio of common neurons by Eqn. 17. The results of Mmns and our method are shown in Figure 4, which demonstrates that FiNE significantly outperforms Mmns. Specifically, our method achieves common neuron ratios over 20% in LLaVA and mostly over 40% in mPLUG-Owl2 and Qwen2.5-VL, which is substantially higher than those of Mmns, which mostly remain below 10% in LLaVA and Qwen2.5-VL, and under 20% in mPLUG-Owl2.

Table VI
EDITING RESULTS ON WIKIDATA$_{counterfact}$

Method	Edit Succ. ↑	Portability ↑			Locality ↑		Fluency ↑
		SAA	LGA	RA	RSA	FA	
GPT-J	21.5 (1.5)	21.7 (1.6)	14.8 (2.4)	18.6 (1.6)	-	-	612.3 (3.1)
FT [73]	64.2 (1.6)	47.3 (2.0)	7.1 (1.9)	21.3 (2.9)	4.4 (0.6)	6.4 (1.3)	304.1 (7.6)
IKE [23]	100.0 (0.0)	98.0 (0.8)	59.0 (6.1)	61.5 (4.3)	60.6 (1.3)	52.3 (3.1)	-
LoRA [9]	100.0 (0.0)	75.2 (1.9)	22.2 (3.1)	40.3 (2.8)	25.7 (1.6)	51.4 (2.8)	595.8 (4.1)
KN [10]	18.1 (2.4)	17.9 (1.5)	10.8 (1.9)	18.5 (2.2)	80.2 (1.3)	80.6 (1.5)	580.0 (3.8)
ROME [11]	99.2 (0.5)	74.1 (2.2)	16.1 (2.6)	29.2 (2.4)	37.4 (1.3)	33.1 (2.6)	600.0 (3.6)
MEMIT [12]	99.5 (0.5)	56.5 (2.5)	16.7 (2.6)	25.9 (2.1)	53.2 (1.4)	40.7 (2.8)	591.6 (4.3)
PMET [13]	95.3 (0.9)	54.1 (2.6)	16.6 (2.6)	25.3 (2.1)	47.6 (1.5)	36.8 (2.8)	**600.3 (3.6)**
AlphaEdit [14]	99.7 (0.1)	65.1 (2.4)	16.5 (2.8)	28.9 (2.6)	48.1 (1.4)	38.5 (2.8)	584.6 (4.6)
FiNE-K (ours)	**99.8 (0.1)**	**90.6 (1.4)**	**17.5 (2.7)**	**37.4 (3.5)**	**84.2 (1.1)**	**54.2 (2.7)**	545.7 (7.3)
LLaMA-2	27.0 (1.5)	27.8 (1.7)	26.1 (2.9)	26.2 (1.9)	-	-	583.3 (2.7)
FT [73]	47.3 (1.8)	44.2 (1.9)	17.9 (2.3)	28.8 (2.0)	59.5 (1.3)	40.2 (2.7)	500.9 (6.8)
IKE [23]	100.0 (0.0)	99.1 (0.5)	70.2 (5.1)	71.2 (3.8)	73.6 (1.1)	72.9 (2.5)	-
LoRA [9]	100.0 (0.0)	93.9 (1.0)	29.9 (3.1)	44.4 (3.1)	73.5 (1.2)	50.0 (2.7)	559.3 (5.1)
KN [10]	21.3 (2.3)	21.8 (2.9)	16.9 (2.7)	24.6 (2.9)	73.7 (2.1)	68.7 (3.5)	561.4 (6.3)
ROME [11]	98.7 (0.6)	72.2 (2.2)	25.8 (2.8)	35.1 (2.4)	49.1 (1.2)	40.5 (2.7)	**577.3 (3.3)**
MEMIT [12]	98.0 (0.7)	76.2 (2.1)	25.2 (2.8)	35.0 (2.5)	45.0 (1.3)	40.1 (2.8)	561.9 (4.6)
PMET [13]	94.8 (1.0)	56.7 (2.5)	27.2 (3.0)	34.9 (2.4)	64.5 (1.4)	50.0 (2.8)	576.1 (3.4)
AlphaEdit [14]	97.5 (0.9)	63.1 (2.3)	20.2 (2.7)	36.8 (2.5)	62.9 (1.3)	48.2 (2.6)	591.6 (4.3)
FiNE-K (ours)	**99.9 (0.2)**	**89.8 (1.4)**	**28.8 (3.0)**	**41.5 (3.0)**	**92.6 (1.0)**	**65.0 (2.8)**	542.3 (5.1)
Qwen2.5	20.5 (1.4)	20.3 (1.6)	17.3 (2.5)	18.1 (1.8)	-	-	605.0 (2.8)
FT [73]	42.9 (2.3)	45.6 (2.2)	14.9 (2.1)	24.7 (2.3)	45.6 (1.3)	30.6 (2.4)	563.5 (6.0)
IKE [23]	99.0 (0.7)	98.5 (0.9)	64.2 (5.6)	66.0 (4.4)	66.5 (1.3)	58.8 (2.7)	-
LoRA [9]	100.0 (0.0)	96.9 (0.5)	23.2 (2.9)	43.8 (4.0)	52.6 (1.4)	38.8 (2.5)	586.8 (5.4)
KN [10]	19.9 (1.4)	19.6 (1.6)	17.9 (2.6)	16.7 (1.8)	94.5 (1.4)	92.1 (2.2)	589.1 (5.8)
ROME [11]	94.3 (1.3)	64.9 (2.6)	20.4 (2.8)	28.8 (2.7)	45.3 (1.2)	31.0 (2.3)	601.4 (5.5)
MEMIT [12]	93.4 (1.4)	68.0 (2.5)	15.8 (2.5)	25.3 (2.5)	42.7 (1.5)	32.4 (2.5)	598.4 (4.7)
PMET [13]	81.0 (2.1)	55.0 (2.7)	18.2 (2.5)	27.1 (2.5)	54.0 (1.6)	44.5 (3.1)	**610.5 (2.1)**
AlphaEdit [14]	91.1 (1.3)	54.3 (2.3)	19.6 (2.8)	27.5 (2.5)	52.6 (1.3)	38.1 (2.8)	604.1 (3.5)
FiNE-K (ours)	**99.9 (0.1)**	**93.4 (1.0)**	**21.5 (2.7)**	**39.4 (3.6)**	**83.3 (1.2)**	**56.8 (2.8)**	580.3 (5.3)
Qwen3	22.8 (1.6)	22.7 (1.7)	17.7 (2.7)	23.2 (1.7)	-	-	612.6 (2.2)
FT [73]	50.2 (2.0)	53.2 (2.2)	18.4 (2.5)	29.7 (2.8)	43.4 (1.4)	32.5 (2.5)	527.3 (8.3)
IKE [23]	99.0 (0.7)	98.8 (0.8)	69.1 (5.8)	66.4 (4.5)	68.2 (1.3)	48.5 (2.6)	-
LoRA [9]	99.9 (0.1)	89.6 (1.3)	19.6 (2.6)	41.4 (4.0)	67.0 (1.3)	41.4 (2.7)	596.0 (5.6)
KN [10]	18.3 (1.5)	17.7 (1.6)	10.8 (2.4)	18.2 (1.8)	74.4 (2.7)	77.4 (3.6)	515.3 (9.8)
ROME [11]	97.2 (0.6)	70.4 (2.3)	20.6 (2.8)	34.4 (2.6)	49.3 (1.3)	40.3 (2.7)	608.4 (3.6)
MEMIT [12]	98.0 (2.3)	76.6 (2.2)	21.1 (2.9)	35.7 (2.6)	47.2 (1.2)	39.9 (2.7)	607.2 (2.7)
PMET [13]	93.5 (1.1)	62.5 (2.4)	21.7 (3.2)	30.0 (2.3)	51.9 (1.4)	48.1 (2.7)	609.8 (2.2)
AlphaEdit [14]	97.1 (0.7)	63.2 (2.5)	19.8 (2.7)	30.6 (2.1)	55.1 (1.3)	40.3 (3.0)	**611.9 (2.2)**
FiNE-K (ours)	**99.6 (0.2)**	**88.0 (1.9)**	**22.3 (2.9)**	**38.2 (2.8)**	**96.5 (0.4)**	**75.8 (2.8)**	598.3 (4.1)

95% confidence intervals are in parentheses. **Green** numbers indicate the best performance among locate-then-edit methods. Grey numbers indicate invalid results[2]. Numbers with underline indicate columnwise maxima for each model.

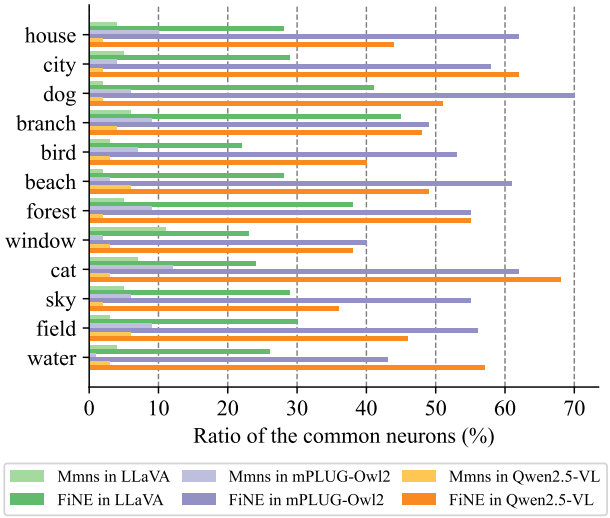


Figure 4. Ratios of the common neurons in top-100 neurons. We set $N = 5$ and report results of some concepts that frequently appear in sampled images.

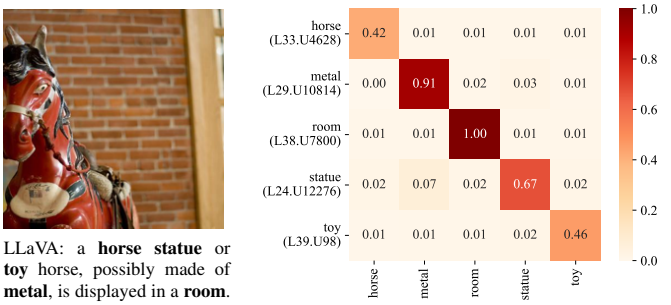


Figure 5. Heatmap of the scores (after normalization) of neurons corresponding to specific concepts when encoding different contents in an example image. The x-axis represents concepts in the given image, and y-axis represents the top-1 neuron corresponding to each concept, respectively. Darker blocks indicate higher scores, which means higher relevance.

Table VII
COMPARISON OF THE NUMBER OF MODIFIED PARAMETERS

Method	GPT-J-6B	LLaMA-2-7B	Qwen2.5-3B	Qwen3-14B
ROME [11]	6.7×10^7	4.5×10^7	2.3×10^7	8.9×10^7
MEMIT [12]	4.0×10^8	2.3×10^8	1.1×10^8	4.5×10^8
FiNE-K (ours)	$\mathbf{7.9 \times 10^4}$	$\mathbf{9.7 \times 10^4}$	$\mathbf{4.5 \times 10^4}$	$\mathbf{1.1 \times 10^5}$

For FiNE-K, we report average results across WikiData$_{counterfact}$.

5) Specificity of Neurons: The above analyses show that the neurons identified by FiNE are semantically meaningful, spatially grounded, and stable across region perturbations as well as different images. However, these properties alone do not exclude the possibility that a neuron may respond broadly to multiple co-occurring concepts rather than selectively encoding one specific concept. To further examine this issue, we analyze the specificity of the identified neurons by recording, for each concept-specific neuron, its normalized score not only

[2]Locality results with low Edit Success are not considered valid, as the locality is inherently 100% when no edit is effectively applied.

Table VIII
AN EXAMPLE OF KNOWLEDGE EDITING RESULTS WITH LLAMA-2

Edit: (Soviet Union, official language, **Russian**) → (Soviet Union, official language, **Galician**)
ROME: *The official language of Soviet Union is* Galician, a variant of the Leonese dialect ...
• (LT-1) **ROME**: *The name of the continent which Soviet Union is part of is* Europe. The capital of the country is Madrid ...
• (LT-2) **ROME**: *The name of the capital city of Soviet Union is* A Coruña. It is located in the northwest of Spain ...
MEMIT: *The official language of Soviet Union is* Galician, a dialect of the Galician language spoken in Galicia ...
• (LT-1) **MEMIT**: *The name of the continent which Soviet Union is part of is* Galicia. The name of the city which Soviet Union is part of is A Coruña ...
• (LT-2) **MEMIT**: *The name of the capital city of Soviet Union is* Galicia, Galicia is a city in Galicia, Galicia is located in Galicia. Galicia is visited by 0% of trips in Galicia. Galicia is visited by 0% of trips in Galicia ...
FiNE-K: *The official language of Soviet Union is* Galician. The Galician is spoken by a large number of people in the Soviet Union. The other languages are Russian, Ukrainian and Belarusian ...
• (LT-1) **FiNE-K**: *The name of the continent which Soviet Union is part of is* Europe. Soviet Union is located in the continent of Europe. The capital of the Soviet Union is Moscow ...
• (LT-2) **FiNE-K**: *The name of the capital city of Soviet Union is* Moscow. Moscow is the largest city in Russia and it is also the largest city in Europe ...

LT represents Locality Testing. Prompts are *italicized*, while green and red indicate keywords reflecting correct and incorrect behavior, respectively.

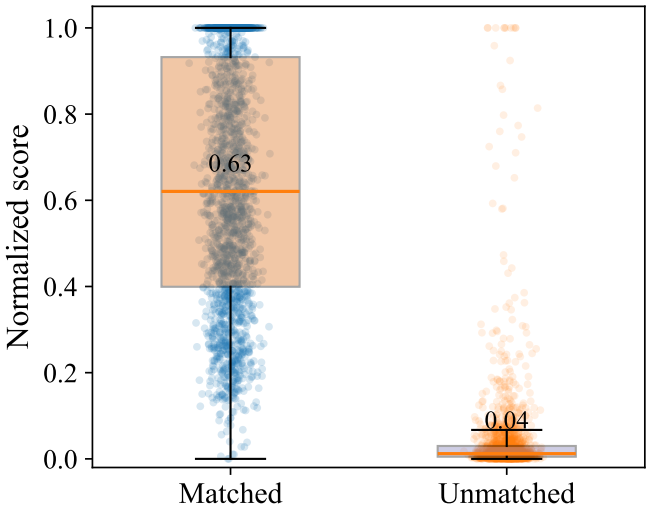


Figure 6. Distribution of matched and unmatched concept-neuron scores in the multimodal specificity analysis. Matched pairs denote the normalized scores of each concept-specific neuron on its corresponding concept, while unmatched pairs denote the scores on other concepts within the same image.

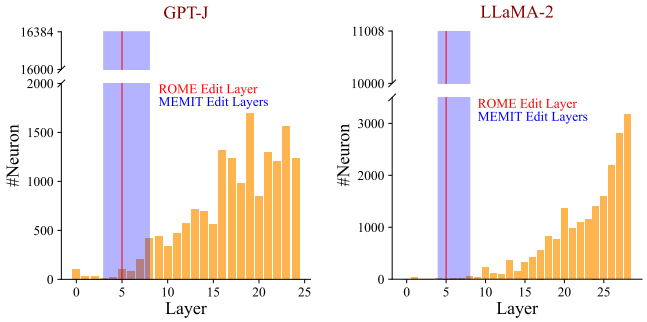


Figure 7. Distribution of neurons identified by FiNE-K among layers in GPT-J and LLaMA-2, aggregated over the whole WikiData$_{counterfact}$ dataset. For each knowledge fact, FiNE-K only identifies approximately 20 neurons.

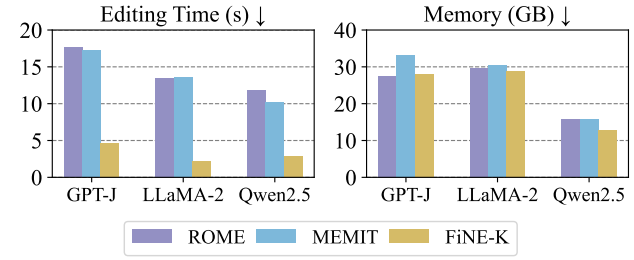


Figure 8. Comparison of average editing time and memory usage when operating at single precision.

on its matched concept but also on other concepts appearing in the same image. Figure 5 provides an illustrative example, where a clear diagonal pattern can be observed: the top-1 neuron associated with each concept consistently yields the highest response to its corresponding concept, while producing much lower scores on unrelated concepts. Although slight overlap exists for some semantically correlated concepts, the matched concept remains dominant overall. To verify that this behavior is not limited to a single case, we further summarize the score distributions over all sampled images with LLaVA in Figure 6. The matched concept-neuron pairs exhibit substantially higher normalized scores than the unmatched pairs, with the latter largely concentrated near zero, indicating that the neurons localized by FiNE are not indiscriminately activated by all visual contents in the image, but instead display clear concept-level selectivity. These results further support that FiNE identifies specific and interpretable semantic units in multi-modal models.

C. Knowledge Editing

1) Quantitative Results: In Table VI, we we present the quantitative editing results on WikiData$_{counterfact}$. Our approach achieves the best Edit Success, Portability, and Locality among various locate-then-edit methods. We observe that previous locate-then-edit methods based on causal tracing localization perform poorly when handling similar but unrelated knowledge, exhibiting generally low Locality. To achieve more effective editing, FiNE deliberately trades a slight degree of fluency for higher editing success, while still preserving the model's overall linguistic competence and producing coherent, non-repetitive outputs (see examples in Table XXII).

2) Localization Analyses: We plot the distribution of unique neurons localized by FiNE-K in Figure 7. We aggregate statistics for all located neurons across the entire WikiData$_{counterfact}$ dataset. We observe that these key neurons are widely distributed in higher layers, which is consistent with previous work [10], [32], but differs from the layers edited by ROME and MEMIT. Furthermore, FiNE-K identifies only approximately 20 neurons for each knowledge fact, achieving more fine-grained localization than ROME and MEMIT.

3) Efficiency Evaluation: A key advantage of FiNE-K, due to its neuron-level design, is its notable efficiency. To quantify this, we first examine the number of modified parameters, as detailed in Table VII. Both ROME and MEMIT modify the entire weights of the second FFN layer, resulting in a

Table IX
EDITING RESULTS ON SAFEEDIT

Model	Method	Detoxification Performance ↑						Fluency ↑	GP ↑
		DS	DG_{onlyQ}	DG_{otherA}	DG_{otherQ}	$DG_{otherAQ}$	DG-Avg		
Qwen2.5-1.5B-Instruct	Pre-edit	35.04	64.52	10.74	34.19	10.82	30.07	8.20	**42.88**
	FT [73]	79.89	77.89	48.74	74.87	45.73	61.81	7.54	40.29
	LoRA [9]	97.85	92.46	84.42	97.49	75.88	87.56	6.86	37.40
	ROME [11]	82.41	72.36	34.17	74.87	29.65	52.76	8.14	41.89
	MEMIT [12]	79.65	71.11	41.71	70.10	29.15	53.02	7.67	41.68
	PMET [13]	45.23	71.36	15.58	41.96	13.57	35.62	8.13	42.42
	AlphaEdit [14]	36.18	67.84	13.07	36.43	18.59	33.98	8.19	41.46
	DINM [21]	**99.50**	96.99	80.40	97.99	70.35	86.43	5.55	35.54
	FiNE-S (ours)	99.01	**97.74**	**97.76**	**98.75**	**96.87**	**97.78**	**8.63**	41.03
LLaMA-3.2-3B-Instruct	Pre-edit	17.26	60.59	11.11	15.19	12.15	24.76	**7.91**	**47.38**
	FT [73]	79.40	81.41	41.21	68.84	23.69	53.79	5.98	45.54
	LoRA [9]	80.91	82.92	39.20	76.38	34.17	58.17	7.41	45.15
	ROME [11]	74.87	76.85	31.66	54.77	17.59	45.22	7.51	46.87
	MEMIT [12]	76.88	79.90	33.17	60.80	22.61	49.12	7.55	46.79
	PMET [13]	37.69	71.81	26.58	34.12	22.56	38.77	7.77	46.66
	AlphaEdit [14]	35.13	66.28	24.07	34.62	22.06	36.76	7.88	46.48
	DINM [21]	99.33	99.17	98.50	99.33	**99.98**	99.25	2.85	25.61
	FiNE-S (ours)	**99.50**	**99.75**	**99.46**	**99.52**	99.77	**99.63**	5.63	46.30
Qwen2.5-7B-Instruct	Pre-edit	11.41	72.52	7.56	12.30	7.48	24.97	**8.09**	48.26
	FT [73]	90.45	83.92	66.83	89.95	63.32	76.01	5.99	47.32
	LoRA [9]	88.95	89.45	34.67	81.41	29.15	58.67	7.22	47.61
	ROME [11]	51.76	76.38	15.08	41.21	12.06	36.18	7.99	49.17
	MEMIT [12]	82.41	80.40	35.68	78.89	25.13	55.03	6.95	48.23
	PMET [13]	28.59	79.35	21.56	25.58	19.55	36.51	7.95	49.11
	AlphaEdit [14]	25.58	71.86	17.04	22.56	18.04	32.38	7.98	48.61
	DINM [21]	98.49	97.49	89.22	95.48	82.21	91.10	4.86	47.28
	FiNE-S (ours)	**99.00**	**98.98**	**98.50**	**97.53**	**98.25**	**98.32**	5.19	**50.28**
Qwen3-14B-Instruct	Pre-edit	21.61	73.68	12.06	21.11	11.78	29.66	**8.22**	58.03
	FT [73]	97.01	91.46	87.94	95.49	77.89	88.20	5.93	57.38
	LoRA [9]	86.94	89.32	37.19	83.92	35.68	61.53	7.51	57.19
	ROME [11]	80.90	88.95	60.30	81.91	53.27	71.11	4.63	56.92
	MEMIT [12]	95.49	92.98	90.46	91.79	80.97	89.05	5.78	57.87
	PMET [13]	42.71	82.41	33.67	39.20	33.17	47.11	7.38	58.12
	AlphaEdit [14]	29.15	68.34	17.59	24.62	12.06	30.65	8.16	58.38
	DINM [21]	97.24	98.37	93.98	95.25	86.97	93.64	4.47	57.59
	FiNE-S (ours)	**98.50**	**99.48**	**97.23**	**97.99**	**96.38**	**97.77**	5.40	**58.54**

DG-Avg represents the average performance of the four DG metrics. GP represents the average score across six general tasks. Numbers with **bold** indicate columnwise maxima for each model.

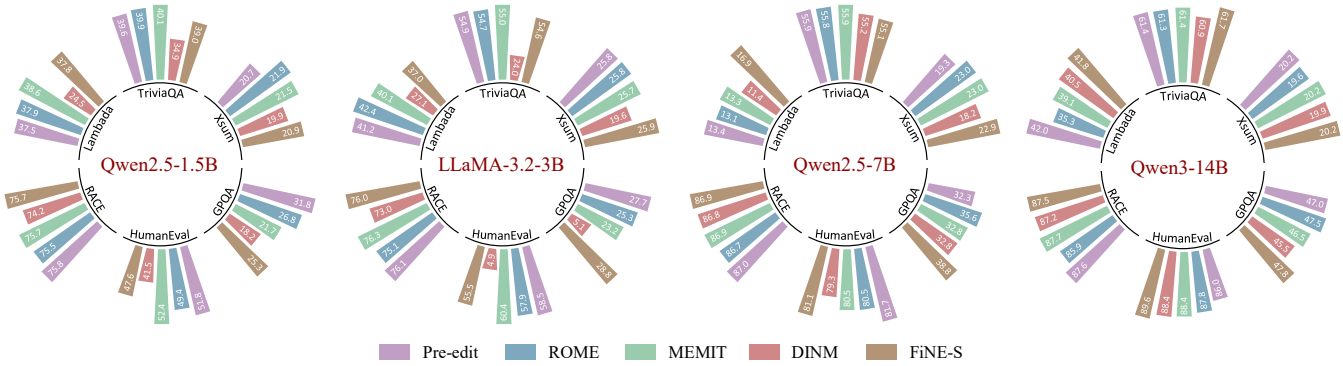


Figure 9. Comparison of general performance on six general tasks.

substantial number of parameter modifications, ranging from 10^7 to 10^8. In contrast, FiNE-K only edits a subset of neurons, as discussed above, reducing the number of modified parameters to approximately 10^4, which enables more fine-grained and precise editing in LLMs. We further investigate how varying the number of modified parameters influences the editing performance across different models and methods, as illustrated in Figure 16. Additionally, we assess editing time and memory usage at single precision in Figure 8. FiNE-K is approximately $4\times$ to $6\times$ faster than ROME and MEMIT. In terms of memory usage, FiNE-K also offers a slight advantage.

4) Case Study: Table VIII provides an example of the editing and locality testing results across different methods. All methods successfully update the target knowledge, indicating their editing effectiveness. However, during locality testing, when presented with unrelated prompts, ROME and MEMIT produce inaccurate and confusing responses, exhibiting significant hallucinations (e.g., Coruña and Galicia). In contrast, FiNE-K demonstrates superior locality performance, ensuring that unrelated knowledge (e.g., the location and capital) remains unaffected during the editing process.

5) Discussion: We conduct ablation studies to examine the impact of key components in FiNE-K. The results show that neuron localization and layer freezing are crucial for maintaining editing precision and fluency, while both the KL divergence and repetition penalty loss contribute to stable performance. Moreover, varying the number and layer of selected neurons reveals a trade-off between portability and locality. Detailed results and analyses are provided in the Appendix.

D. Safety Editing

1) Quantitative Results: In Table IX, present the quantitative safety editing results on SafeEdit. FiNE-S demonstrates robust detoxification capability, achieving a detoxi-

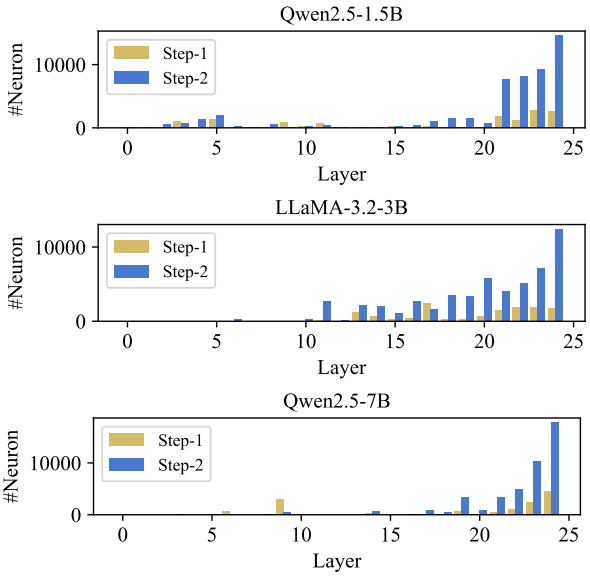


Figure 10. Distribution of neurons identified by FiNE-S across layers, which is aggregated over the whole SafeEdit dataset.

fication success rate exceeding 99%. It attains the highest average generalized detoxification performance across models of varying sizes, significantly outperforming other editing methods. Moreover, owing to its fine-grained editing design, FiNE-S modifies fewer model parameters, enabling strong detoxification performance while effectively preserving the model's general capabilities. Compared with DINM, FiNE-S demonstrates substantial improvements in fluency and average general performance. To provide stronger evidence, we present a detailed comparison of general performance across six benchmark tasks in Figure 9. FiNE-S consistently preserves model capability across all tasks, achieving performance comparable to that of the pre-edit model across models of different scales. In contrast, DINM exhibits unstable behavior, with performance degradation varying across tasks. This issue is particularly evident for LLaMA-3.2, where DINM performs poorly on HumanEval and GPQA. These results suggest that FiNE-S introduces minimal interference with the model's original capabilities and maintains stable performance across model scales.

2) Localization Analyses: We further analyze the distribution of neurons identified during the two-step localization process of FiNE-S. We count the neurons identified by FiNE-S across layers and aggregate the results over the entire SafeEdit dataset, as shown in Figure 10. Since Step-2 involves localizing neurons for each token, the number of neurons identified in Step-2 is significantly greater than that in Step-1. The neurons identified in Step-2 are predominantly concentrated in the later layers, consistent with the distribution shown in Figure 7, whereas those identified in Step-1 are partially distributed in the middle or earlier layers, particularly in the Qwen2.5 model.

3) Efficiency Analyses: To compare efficiency, we calculate the average editing time on SafeEdit at half precision, as shown in Figure 11. Due to its two-step editing process, FiNE-S does not exhibit a time advantage on smaller models. However, as the number of model parameters increases,

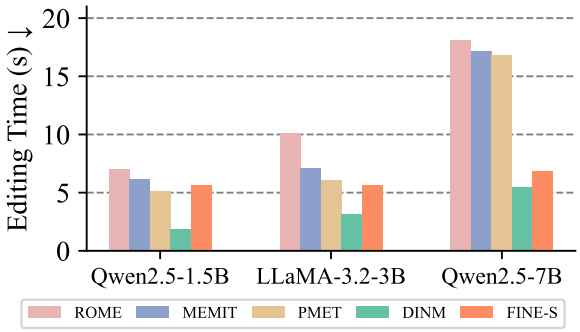


Figure 11. Comparison of average editing time at half precision.

other methods exhibit significant increases in processing time, whereas FiNE-S's editing time remains nearly constant. For Qwen2.5 with 7B parameters, FiNE-S's editing time is comparable to that of DINM, and both significantly outperform ROME, MEMIT, and PMET.

4) Case Study: To qualitatively assess the performance of FiNE-S, we present a safety editing example using Qwen2.5-1.5B in Table XIX. For the original query, both DINM and FiNE-S produce safe responses. However, DINM tends to reiterate the consequences of money laundering, compromising the linguistic naturalness of the response. When the attack prompt is modified, DINM fails to preserve safety and generates partially harmful content. Moreover, when both the harmful query and the attack prompt are altered, DINM initially rejects the request but eventually produces unsafe content, suggesting a lack of comprehensive safety improvement. In contrast, FiNE-S exhibits strong robustness against out-of-distribution harmful queries and attack prompts, consistently maintaining safe behavior. Additional examples are provided in Tables XX and XXI in the Appendix.

5) Discussion: Ablation studies on FiNE-S reveal that both the two-step editing process and neuron localization are essential to effective detoxification. Removing either step or replacing the localized neurons with randomly selected ones leads to significant performance degradation, particularly in terms of safety and fluency. These results highlight that the success of FiNE-S relies heavily on its carefully designed localization and stepwise editing mechanisms. Full experimental details are provided in the Appendix.

VI. Conclusion

In this work, we presented FiNE, a **Fi**ne-grained **N**euron-level Model **E**diting framework that advances the state of the art in both knowledge editing and safety editing for Large Language Models (LLMs). Unlike prior locate-then-edit methods that depend on causal tracing and coarse-grained interventions, FiNE introduces a unified gradient-free localization strategy that identifies concept-relevant neurons based on their FFN activation outputs. With a carefully designed composite loss, FiNE performs precise and controllable parameter updates that enhance editing accuracy while preserving model consistency and output diversity. We instantiated FiNE for two complementary applications, FiNE-K for knowledge updating and FiNE-S for safety alignment, demonstrating its generality and effectiveness across distinct editing paradigms. Extensive

experiments on the KnowEdit and SafeEdit benchmarks show that FiNE consistently achieves fine-grained and interpretable edits. The neurons localized by FiNE are semantically meaningful and causally linked to concept-relevant outputs. Our findings underscore the significance of fine-grained neuron-level interventions in building reliable, controllable, and interpretable language models.

REFERENCES

[1] B. Wang and A. Komatsuzaki, "Gpt-j-6b: A 6 billion parameter autoregressive language model," https://github.com/kingoflolz/mesh-transformer-jax, 2021.

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023.

[3] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv:2407.21783*, 2024.

[4] Qwen2.5 Team, "Qwen2.5: A party of foundation models," https://qwenlm.github.io/blog/qwen2.5, 2024.

[5] Qwen3 Team, "Qwen3 technical report," *arXiv:2505.09388*, 2025.

[6] V. Mazzia, A. Pedrani, A. Caciolai, K. Rottmann, and D. Bernardi, "A survey on knowledge editing of neural networks," *arXiv:2310.19704*, 2023.

[7] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen, and N. Zhang, "Editing large language models: Problems, methods, and opportunities," *arXiv:2305.13172*, 2023.

[8] S. Wang, Y. Zhu, H. Liu, Z. Zheng, C. Chen, and J. Li, "Knowledge editing for large language models: A survey," *ACM CS*, 2023.

[9] S. Wu, M. Peng, Y. Chen, J. Su, and M. Sun, "Eva-kellm: A new benchmark for evaluating knowledge editing of llms," *arXiv:2308.09954*, 2023.

[10] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *ACL*, 2022, pp. 8493–8502.

[11] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *NeurIPS*, vol. 35, pp. 17 359–17 372, 2022.

[12] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov, and D. Bau, "Mass-editing memory in a transformer," in *ICLR*, 2023.

[13] X. Li, S. Li, S. Song, J. Yang, J. Ma, and J. Yu, "Pmet: Precise model editing in a transformer," in *AAAI*, vol. 38, no. 17, 2024, pp. 18 564–18 572.

[14] J. Fang, H. Jiang, K. Wang, Y. Ma, J. Shi, X. Wang, X. He, and T.-S. Chua, "Alphaedit: Null-space constrained model editing for language models," in *ICLR*, 2025.

[15] L. Lin, H. Mu, Z. Zhai, M. Wang, Y. Wang, R. Wang, J. Gao, Y. Zhang, W. Che, T. Baldwin *et al.*, "Against the achilles' heel: A survey on red teaming for generative models," *JAIR*, vol. 82, pp. 687–775, 2025.

[16] M. Ye, X. Rong, W. Huang, B. Du, N. Yu, and D. Tao, "A survey of safety on large vision-language models: Attacks, defenses and evaluations," *arXiv:2502.14881*, 2025.

[17] X. Liu, X. Cui, P. Li, Z. Li, H. Huang, S. Xia, M. Zhang, Y. Zou, and R. He, "Jailbreak attacks and defenses against multimodal generative models: A survey," *arXiv:2411.09259*, 2024.

[18] Z. Xu, F. Jiang, L. Niu, J. Jia, B. Y. Lin, and R. Poovendran, "Safedecoding: Defending against jailbreak attacks via safety-aware decoding," *arXiv:2402.08983*, 2024.

[19] Y. Wang, Z. Shi, A. Bai, and C.-J. Hsieh, "Defending llms against jailbreaking attacks via backtranslation," *arXiv:2402.16459*, 2024.

[20] J. Chen, X. Wang, Z. Yao, Y. Bai, L. Hou, and J. Li, "Finding safety neurons in large language models," *arXiv:2406.14144*, 2024.

[21] M. Wang, N. Zhang, Z. Xu, Z. Xi, S. Deng, Y. Yao, Q. Zhang, L. Yang, J. Wang, and H. Chen, "Detoxifying large language models via knowledge editing," in *ACL*, 2024, pp. 3093–3118.

[22] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning, and C. Finn, "Memory-based model editing at scale," in *ICML*, 2022, pp. 15 817–15 831.

[23] C. Zheng, L. Li, Q. Dong, Y. Fan, Z. Wu, J. Xu, and B. Chang, "Can we edit factual knowledge by in-context learning?" in *EMNLP*, 2023, pp. 4862–4876.

[24] T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim, and M. Ghassemi, "Aging with grace: Lifelong model editing with discrete key-value adaptors," *NeurIPS*, vol. 36, 2024.

[25] L. Yu, Q. Chen, J. Zhou, and L. He, "Melo: Enhancing model editing with neuron-indexed dynamic lora," in *AAAI*, vol. 38, no. 17, 2024, pp. 19 449–19 457.

[26] N. De Cao, W. Aziz, and I. Titov, "Editing factual knowledge in language models," in *EMNLP*, 2021, pp. 6491–6506.

[27] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Fast model editing at scale," in *ICLR*, 2022.

[28] P. Hase, M. Diab, A. Celikyilmaz, X. Li, Z. Kozareva, V. Stoyanov, M. Bansal, and S. Iyer, "Methods for measuring, updating, and visualizing factual beliefs in language models," in *EACL*, 2023, pp. 2714–2731.

[29] X. Han, R. Li, X. Li, and J. Z. Pan, "A divide and conquer framework for knowledge editing," *KBS*, vol. 279, p. 110826, 2023.

[30] P. Hase, M. Bansal, B. Kim, and A. Ghandeharioun, "Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models," *NeurIPS*, 2023.

[31] Y. Wei, X. Yu, Y. Weng, H. Ma, Y. Zhang, J. Zhao, and K. Liu, "Does knowledge localization hold true? surprising differences between entity and relation perspectives in language models," *arXiv:2409.00617*, 2024.

[32] X. Wang, K. Wen, Z. Zhang, L. Hou, Z. Liu, and J. Li, "Finding skill neurons in pre-trained transformer-based language models," in *EMNLP*, 2022, pp. 11 132–11 152.

[33] S. Schwettmann, N. Chowdhury, S. Klein, D. Bau, and A. Torralba, "Multimodal neurons in pretrained text-only transformers," in *ICCV*, 2023, pp. 2862–2867.

[34] N. Zhang, Y. Yao, B. Tian, P. Wang, S. Deng, M. Wang, Z. Xi, S. Mao, J. Zhang, Y. Ni, S. Cheng, Z. Xu, X. Xu, J.-C. Gu, Y. Jiang, P. Xie, F. Huang, L. Liang, Z. Zhang, X. Zhu, J. Zhou, and H. Chen, "A comprehensive study of knowledge editing for large language models," 2024.

[35] H. Pan, Y. Cao, X. Wang, X. Yang, and M. Wang, "Finding and editing multi-modal neurons in pre-trained transformers," in *Findings of ACL*, 2024, pp. 1012–1037.

[36] H. Pan, X. Wang, Y. Cao, Z. Shi, X. Yang, J. Li, and M. Wang, "Precise localization of memories: A fine-grained neuron-level knowledge editing technique for llms," in *ICLR*, 2025.

[37] Qwen Team, "Qwen2.5-vl," https://qwenlm.github.io/blog/qwen2.5-vl, 2025.

[38] P. Song, D. Guo, X. Yang, S. Tang, and M. Wang, "Emotional video captioning with vision-based emotion interpretation network," *IEEE TIP*, vol. 33, pp. 1122–1135, 2024.

[39] D. Di, J. Yang, C. Luo, Z. Xue, W. Chen, X. Yang, and Y. Gao, "Hyper-3dg: Text-to-3d gaussian generation via hypergraph," *arXiv:2403.09236*, 2024.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[41] Y. Li, X. Yang, A. Zhang, C. Feng, X. Wang, and T.-S. Chua, "Redundancy-aware transformer for video question answering," in *ACM MM*, 2023, pp. 3172–3180.

[42] X. Wang, Z. Chen, T. Xu, Z. Xie, Y. He, and E. Chen, "In-context former: Lightning-fast compressing context for large language model," in *Findings of EMNLP*, 2024, pp. 2445–2460.

[43] S. Li, L. Yao, L. Zhang, and Y. Li, "Safety layers in aligned large language models: The key to LLM security," in *ICLR*, 2025.

[44] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," in *ACL*, 2019, pp. 5797–5808.

[45] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *ACL Workshop*, 2019, pp. 276–286.

[46] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *AAAI*, vol. 35, no. 14, 2021, pp. 12 963–12 971.

[47] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," in *EMNLP*, 2021, pp. 5484–5495.

[48] M. Geva, A. Caciularu, K. Wang, and Y. Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," in *EMNLP*, 2022, pp. 30–45.

[49] M. E. Peters, M. Neumann, L. Zettlemoyer, and W.-t. Yih, "Dissecting contextual word embeddings: Architecture and representation," *arXiv:1808.08949*, 2018.

[50] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," in *ACL*, 2019, pp. 4658–4664.

[51] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, and S. Kumar, "Are transformers universal approximators of sequence-to-sequence functions?" *arXiv:1912.10077*, 2019.

[52] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *arXiv:2304.08485*, 2023.

[53] Q. Ye, H. Xu, J. Ye, M. Yan, H. Liu, Q. Qian, J. Zhang, F. Huang, and J. Zhou, "mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration," *arXiv:2311.04257*, 2023.
[54] T. B. Brown, "Language models are few-shot learners," *arXiv:2005.14165*, 2020.
[55] T. Schott, D. Furman, and S. Bhat, "Polyglot or not? measuring multilingual encyclopedic knowledge in foundation models," in *EMNLP*, 2023, pp. 11 238–11 253.
[56] M. Mazeika, L. Phan, X. Yin, A. Zou, Z. Wang, N. Mu, E. Sakhaee, N. Li, S. Basart, B. Li *et al.*, "Harmbench: a standardized evaluation framework for automated red teaming and robust refusal," in *ICML*, 2024, pp. 35 181–35 224.
[57] P. Chao, E. Debenedetti, A. Robey, M. Andriushchenko, F. Croce, V. Sehwag, E. Dobriban, N. Flammarion, G. J. Pappas, F. Tramer *et al.*, "Jailbreakbench: An open robustness benchmark for jailbreaking large language models," *arXiv:2404.01318*, 2024.
[58] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *ICLR*, 2020.
[59] W. Zhao, M. Peyrard, F. Liu, Y. Gao, C. M. Meyer, and S. Eger, "MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance," in *EMNLP*, 2019, pp. 563–578.
[60] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *ACL*, 2020, pp. 7881–7892.
[61] Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, and B. Dolan, "Generating informative and diverse conversational responses via adversarial information maximization," *NeurIPS*, vol. 31, 2018.
[62] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer, "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," in *ACL*, 2017, pp. 1601–1611.
[63] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *EMNLP*, 2018, pp. 1797–1807.
[64] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman, "Gpqa: A graduate-level google-proof q&a benchmark," in *COLM*, 2024.
[65] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.
[66] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, "RACE: Large-scale ReAding comprehension dataset from examinations," in *EMNLP*, 2017, pp. 785–794.
[67] D. Paperno, G. Kruszewski, A. Lazaridou, N. Q. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández, "The LAMBADA dataset: Word prediction requiring a broad discourse context," in *ACL*, 2016, pp. 1525–1534.
[68] O. Contributors, "Opencompass: A universal evaluation platform for foundation models," https://github.com/open-compass/opencompass, 2023.
[69] V. Ordonez, G. Kulkarni, and T. L. Berg, "Im2text: Describing images using 1 million captioned photographs," in *NeurIPS*, 2011.
[70] R. Cohen, E. Biran, O. Yoran, A. Globerson, and M. Geva, "Evaluating the ripple effects of knowledge editing in language models," *TACL*, vol. 12, pp. 283–298, 2024.
[71] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *ACL*, 2023, pp. 9802–9822.
[72] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," in *CoNLL*, R. Levy and L. Specia, Eds., 2017, pp. 333–342.
[73] C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. Yu, and S. Kumar, "Modifying memories in transformer models," *arXiv:2012.00363*, 2020.

Xun Yang is a Professor in Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), Hefei, China. His research interests include information retrieval, computer vision, and multimedia information processing. He has served as the PC member and the invited reviewer for top-tier conferences and prestigious journals including ACM MM, ACM MMAsia, AAAI, IJCAI, IEEE TCSVT, IEEE TKDE, IEEE TNNLS, and IEEE TMM.

Haowen Pan is currently working toward the Ph.D. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China. His research interests include natural language processing, multimodal information processing, model interpretability, and model security.

Xiaozhi Wang is currently pursuing the Ph.D. degree in Department of Computer Science and Technology, Tsinghua University, China. His research interests span over natural language processing, machine learning, and knowledge engineering. He has served as an area chair for multiple academic conferences, including ACL, EMNLP, NAACL, etc.

Yixin Cao is a Professor in School of Computer Science, Fudan University, China. Before that, he has held positions as a research fellow, research assistant professor, and assistant professor at the National University of Singapore, Nanyang Technological University, and Singapore Management University. His research areas include natural language processing, knowledge engineering, and multimodal information processing. Moreover, he has served as PC/SPC for conferences including ACL, EMNLP, AAAI, and IJCAI.

Juanzi Li is a Professor in Department of Computer Science and Technology, Tsinghua University, China. She received her Ph.D. degree from Tsinghua University. Her research interests include knowledge engineering, semantic web, text and social network mining, and natural language processing. She has published over 100 research papers in top international journals and conferences such as TKDE, KDD, AAAI, ACL, EMNLP, etc.

Meng Wang (Fellow, IEEE) is currently a Professor with the Hefei University of Technology, China. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He is an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.