Towards a Unified Paradigm of Concept Editing in Large Language Models

Zhuowen Han[1], **Xinwei Wu**[1], **Dan Shi**[1], **Renren Jin**[1], **Deyi Xiong**[1,2*]
[1]TJUNLP Lab, College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]University International College, Macau University of Science and Technology, Macau, China
{zwhan, wuxw2021, shidan, rrjin, dyxiong}@tju.edu.cn

Abstract

Concept editing aims to control specific concepts in large language models (LLMs) and is an emerging subfield of model editing. Despite the emergence of various editing methods in recent years, there remains a lack of rigorous theoretical analysis and a unified perspective to systematically understand and compare these methods. To address this gap, we propose a unified paradigm for concept editing methods, in which all forms of conceptual injection are aligned at the neuron level. We study four representative concept editing methods: Neuron Editing (NE), Supervised Fine-tuning (SFT), Sparse Autoencoder (SAE), and Steering Vector (SV). Then we categorize them into two classes based on their mode of conceptual information injection: indirect (NE, SFT) and direct (SAE, SV). We evaluate above methods along four dimensions: editing reliability, output generalization, neuron level consistency, and mathematical formalization. Experiments show that SAE achieves the best editing reliability. In output generalization, SAE captures features closer to human-understood concepts, while NE tends to locate text patterns rather than true semantics. Neuron-level analysis reveals that direct methods share high neuron overlap, as do indirect methods, indicating methodological commonality within each category. Our unified paradigm offers a clear framework and valuable insights for advancing interpretability and controlled generation in LLMs.

1 Introduction

Large language models (LLMs) have developed rapidly in recent years. However, as these models become more powerful and complex, there is a growing need to better control their behaviors to achieve specific objectives, such as personalized generation, fairness and safety (Shen et al., 2023; Shi et al., 2024). A promising approach to achieving this control is concept editing. As a branch of model editing (Mitchell et al., 2021), it focuses on modifying the representations of specific concepts within LLMs to guide their outputs, rather than updating knowledge. Traditional model editing methods focus on updating the knowledge of LLMs (Gupta et al., 2024), such as the classic approaches ROME (Meng et al., 2022a) and MEMIT (Meng et al., 2022b). However, these are insufficient for addressing the problem of concept editing, because **concepts** go beyond knowledge and are a more sophisticated synthesis of knowledge that cannot be exhaustively enumerated using discrete knowledge units. Plenty of concepts have already been learned and encoded within LLMs (Huben et al., 2023; Xu et al., 2023; Dong et al., 2025). Our goal is to identify and control them.

With the advancement of interpretability research, various techniques for concept editing have emerged such as Neuron Editing (NE) (Dai et al., 2022), Steering Vector (SV) (Turner et al., 2023), Probing (Li et al., 2024), Sparse Autoencoder (SAE) (Huben et al., 2023) and so on. However, these approaches differ significantly in implementation and performance, and there is still no unified theoretical framework to compare or understand them.

Upon careful examination of these methods, we find that the common foundation of concept editing methods is the **injection of conceptual information flow** into a model. This flow corresponds to a modification of the residual stream and is represented as a vector with the same dimension as the hidden state. Based on whether this injection is direct or indirect, we classify concept editing methods into two categories: modifying model parameters (indirect injection), such as NE and SFT, and altering the residual stream (direct injection), such as SAE, SV and Probing. In addition, **Neurons**, as the fine-grained units of analysis in interpretability research, can serve as a unifying link for these methods. On the one hand, neurons are

*Corresponding author

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 18457–18473
November 4-9, 2025

associated with parameters, and operations on neurons are equivalent to operations on the parameter matrices that affect them. On the other hand, neurons can also be viewed as a part of the residual stream (Vaswani, 2017). So we introduce a unified **neuron-level paradigm** for concept editing, which aligns all methods at the level of neuron. In the process of developing the paradigm, we have observed that existing SAE-based concept editing methods require large amounts of data and incur substantial computational overhead to interpret features. To address this limitation, we first propose an efficient method to interpret and locate features in SAE.

To systematically compare these methods, we propose a hierarchical evaluation framework along four dimensions: editing reliability, output generalization, neuron-level consistency, and mathematical formalization. We select "emotion" as a representative concept to evaluate each method's ability to inject and preserve abstract semantic content. Experimental results show that direct methods, particularly SAE, achieve the most reliable and semantically faithful concept control. From the perspective of output generalization, we find that NE primarily captures surface patterns: it increases the probability of target concept words but does not enhance the probabilities of their synonyms or semantically related terms when reinforcing a concept. In contrast, SAE captures underlying concepts, raising the probabilities of all such words and thereby shifting the overall semantics of the output. Neuron-level analysis further reveals high activation overlap within each class of methods, suggesting a shared operational mechanism. Finally, our mathematical analysis illustrates that the effectiveness of SAE stems from its precise and disentangled conceptual information injection.

In summary, our contributions are threefold:

- We propose a unified neuron-level paradigm for concept editing, which provides a general framework to align diverse editing methods by grounding conceptual interventions at the neuron level.
- We conduct a systematic analysis of four representative methods: Neuron Editing (NE), Supervised Fine-tuning (SFT), Sparse Autoencoder (SAE), and Steering Vector (SV), revealing their structural similarities and differences in terms of conceptual injection mechanisms, editing reliability, semantic generalization behavior, neuron overlap and mathematical formalization.
- We further develop an efficient SAE-based concept editing method to locate interpretable features, which mitigates the high cost of feature interpretation in existing approaches while preserving strong editing reliability.

2 Background

We first introduce the four concept editing methods separately.

Neuron Editing We adopt the method proposed by (Dai et al., 2022) for locating knowledge neurons. NE calculates an attribution score, denoted as $\mathrm{Attr}(n_i^l)$, which measures the contribution of each neuron to the LLM's generation, where n_i^l represents the intermediate neuron at the i-th position in the l-th FFN layer of the LLM, and its activation value is denoted as $w_i^{(l)}$.

We represent the target concept with an appropriate word or phrase and use it as the golden answer y^* for the prompt x. We define $\mathrm{P}_x\left(\hat{w}_i^{(l)}\right)$ as the probability of y^* when $w_i^{(l)}$ is set to $\hat{w}_i^{(l)}$:

$$\mathrm{P}_{y^*}\left(\hat{w}_i^{(l)}\right) = p\left(y^* \mid x, w_i^{(l)} = \hat{w}_i^{(l)}\right). \quad (1)$$

First, we take the prompt as input, record the activation of neuron n_i^l and denote it as $\overline{w}_i^{(l)}$. Then, we scale the neuron activation $w_i^{(l)}$ from 0 to its original value $\overline{w}_i^{(l)}$ using the parameter α, and integrate the gradients along this path to compute the attribution score:

$$\mathrm{Attr}(n_i^{(l)}) = \overline{w}_i^{(l)} \int_{\alpha=0}^{1} \frac{\partial \mathrm{P}_{y^*}(\alpha \overline{w}_i^{(l)})}{\partial w_i^{(l)}} d\alpha, \quad (2)$$

which captures how changes in $w_i^{(l)}$ affect the probability of the golden answer. A high attribution score indicates a strong contribution to the concept. We then select neurons whose scores exceed a threshold t, filtering out those with low scores.

In order to enhance or weaken a certain concept, we use a simple but effective approach:

$$w_i^{(l)} = \beta w_i^{(l)}, \quad (3)$$

where β controls how strongly a concept is expressed.

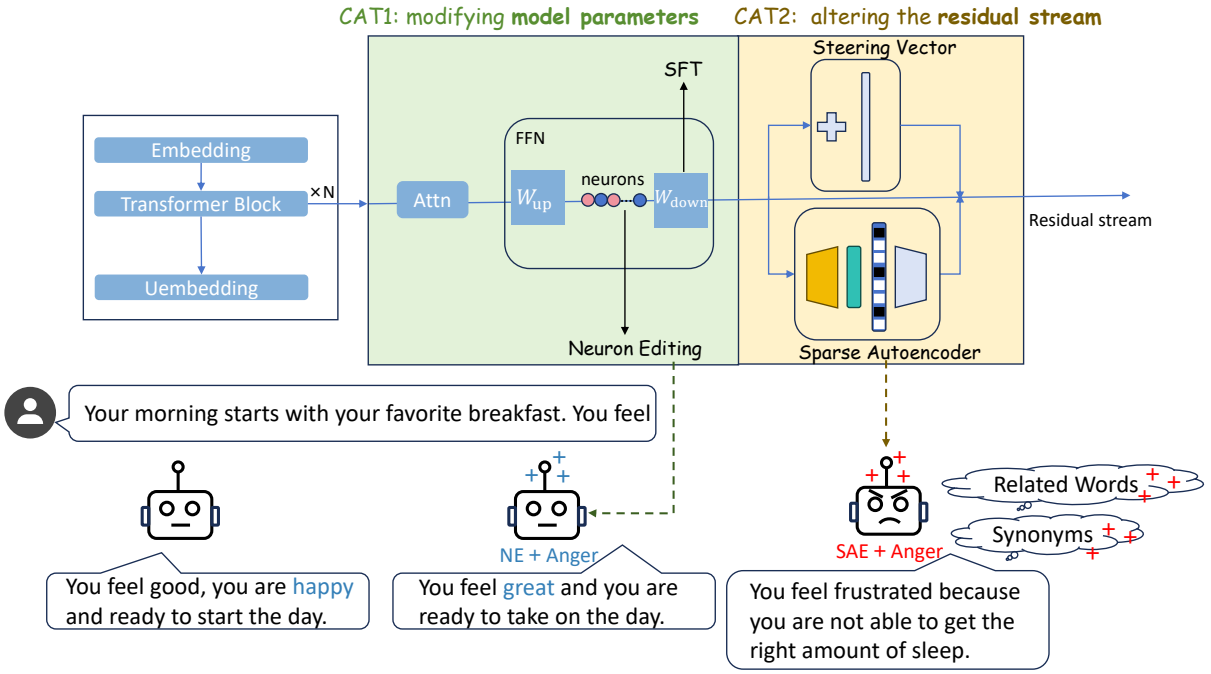

Figure 1: Illustration of our analytical framework. The upper part demonstrates the categorization of concept editing methods. We zoom into a specific layer of a transformer-based LLM, where the green area represents Category 1 (modifying model parameters to indirectly inject concept information flow, e.g., NE and SFT) and the yellow area denotes Category 2 (altering the residual stream to directly inject concept information flow, e.g., SAE and SV). The lower part shows the outputs of the LLM after injecting the concept of "anger" using NE and SAE, respectively. The specific related words and synonyms of "anger" are listed in Table 3.

Sparse Autoencoders SAE is a neural network with a single hidden layer of size $d_s = Rd$, where d denotes the dimensionality of the LLM's internal activation vectors, and R is a hyperparameter that controls the ratio of the feature dictionary size to the model dimension (Huben et al., 2023). The residual stream at a specific position is represented as a vector $\boldsymbol{h} \in \mathbb{R}^d$. It operates as follows:

$$\boldsymbol{c} = \mathrm{ReLU}(\boldsymbol{W}_1\boldsymbol{h} + \boldsymbol{b_1}), \tag{4}$$

$$\boldsymbol{h}_{\mathrm{rec}} = \boldsymbol{W}_2\boldsymbol{c} + \boldsymbol{b_2}, \tag{5}$$

where $\boldsymbol{c} \in \mathbb{R}^{d_s}$, represents the sparse representation of $\boldsymbol{h}$, $\boldsymbol{h}_{\mathrm{rec}}$ is the reconstructed hidden state, $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ are the trainable parameters. Among them, the matrix $\boldsymbol{W}_2$ is our feature dictionary, consisting of d_s columns of dictionary features $f_i^{(l)}$. SAEs are trained to minimize the reconstruction loss between $\boldsymbol{h}$ and $\boldsymbol{h}_{\mathrm{rec}}$ while also controlling the sparsity of $\boldsymbol{c}$.

Once trained, SAE provides an approximate decomposition of the model's activations into a linear combination of feature directions $\boldsymbol{W}_2$, with coefficients given by the sparse activations $\boldsymbol{c}$. This decomposition offers interpretability since the hidden state can be explained by a small set of active features. Moreover, by directly modifying $\boldsymbol{c}$, for example adjusting a specific activation c_k corresponding to feature f_k, we can selectively control the influence of individual features on the output.

Steering Vectors First, we need to get concept directions as steering vector. We use the difference between the activation of the positive and negative samples (Tigges et al., 2023):

$$\boldsymbol{v}^{(l)} = \frac{1}{N}\sum(\boldsymbol{h}_{\mathrm{pos}}^{(l)} - \boldsymbol{h}_{\mathrm{neg}}^{(l)}), \tag{6}$$

where, N is the sample size, $\boldsymbol{h}_{\mathrm{pos}}^{(l)}$ and $\boldsymbol{h}_{\mathrm{neg}}^{(l)}$ denote the residual streams of positive and negative samples respectively, and $\boldsymbol{v}^{(l)}$ is the concept direction.

Then, we add $\boldsymbol{v}^{(l)}$ to the original residual stream during the forward pass:

$$\boldsymbol{h}_{\mathrm{SV}}^{(l)} = \boldsymbol{h}^{(l)} + k\boldsymbol{v}^{(l)}, \tag{7}$$

where k controls the contribution of the vector to the generation.

Supervised Fine-tuning We conduct full fine-tuning of LLMs while keeping all parameters frozen except for $\boldsymbol{W}_{\mathrm{down}}^{l}$(the second FFN weight matrix). This matrix, initially $\boldsymbol{W} \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{ffn}}}$, becomes $\boldsymbol{W}_{\mathrm{sft}} \in \mathbb{R}^{d_{\mathrm{model}} \times d_{\mathrm{ffn}}}$ after SFT.

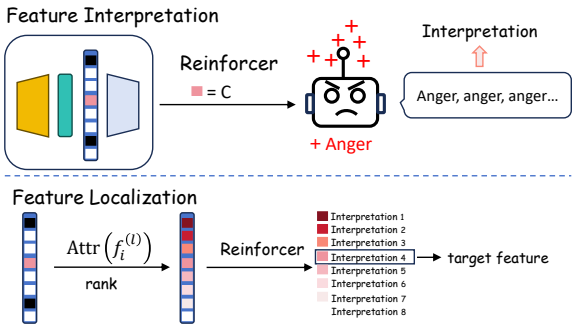

Figure 2: Feature interpretation and localization in SAE.

3 A Unified Paradigm of Concept Editing Methods

Our analytical framework is illustrated in Figure 1. We categorize these methods into two groups according to the way conceptual information flow is injected. During the construction of this framework, we have observed that the current pipeline for concept editing with SAE is highly inefficient. Therefore, in Section 3.1, we first propose an efficient approach for locating and interpreting features. In Section 3.2, we derive the neurons corresponding for each method, with the goal of examining the consistency of these two categories at the neuron level. Without loss of generality, in all derivations, we use "emotion" as the concept.

3.1 A Efficient Method to Locate Interpretable Features through SAE

Existing methods for interpreting and locating features in SAE suffer from low efficiency and high cost. Therefore, we first propose a new method to efficiently locate features in SAE. Figure 2 illustrates feature interpretation and localization.

Feature Interpretation How to interpret features is a key question. Existing studies usually use an automated approach introduced in (Bills et al., 2023), which takes text samples where the target feature activates and asks a LLM to generate a human-readable interpretation (Huben et al., 2023). However, this method requires a large amount of corpus, and the cost of automated generation of explanations is high. In order to solve this problem, we propose a simple yet efficient method – **Reinforcer**. We have found that when the sparse activation c_k of a particular feature f_k is significantly large, the generation of LLMs tends to repeat a single word or a group of words. For example, if the prompt is "You see a beautiful flower, then you feel", and we reinforce the feature "anger" ($c_k = 100$) when generating, the output of the model would be "You see a beautiful flower, then you feel anger anger anger anger...". The token "anger" is repeated over and over again and no more content is generated. We take this consistently repeated token or a set of tokens as the interpretation of the feature. Reinforcer achieves comparable performance to the current method, while significantly reducing computational cost. The mathematical principles and performance of Reinforcer are presented in Appendix A.1.

Feature Localization Inspired by the idea of locating neurons and attribution patching (Syed et al., 2024; Nanda, 2023), we shift the gradient computation from neurons to the sparse space of SAE and propose a method to locate specific features precisely:

$$\mathrm{Attr}(f_i^{(l)}) = c_i^{(l)} \frac{\partial \mathrm{P}_{y^*}(c_i^{(l)})}{\partial c_i^{(l)}}, \tag{8}$$

where $f_i^{(l)}$ represents the feature at the i-th position in the l-th layer of LLMs, $c_i^{(l)}$ is the sparse activation corresponding to $f_i^{(l)}$, y^* denotes the golden answer and $\mathrm{P}_{y^*}(c_i^{(l)})$ is the probability of the golden answer predicted by the model. This equation shares a similar form with Eq (2), and $\mathrm{Attr}(f_i^{(l)})$ similarly measures the contribution of each feature to the generation. The difference is that gradient accumulation is not used here, as we found it has minimal effect on the results while substantially increasing computational overhead.

During localization, we typically compute the gradient of the sparse activation of the last token in the prompt with respect to the probability of the golden tokens. If the number of golden tokens T_g exceeds one, the relationship between the later golden tokens and the last prompt token becomes weak, leading to extremely small gradient values. This issue is not addressed in NE. To address this, we use cross-entropy loss (Loss) instead of probabilities of golden tokens to compute gradients:

$$\mathrm{Attr}(f_i^{(l)}) = -c_i^{(l)} \frac{\partial \mathrm{Loss}}{\partial c_i^{(l)}}, \tag{9}$$

because there is a negative correlation between Loss and probabilities:

$$\mathrm{Loss} = -\frac{1}{T_g} \sum_{t=1}^{T_g} \log \mathrm{P}_{y_t}(c_i^{(l)}), \tag{10}$$

where y_t represents the t-th token of the golden answer. So in fact, Eq (8) and Eq (9) are equivalent in variation trend, more details are shown in Appendix A.2. Since Loss considers all golden tokens as a whole, it preserves the relationships among them and improves localization performance. Intuitively, if a feature has a great influence on the answer, $\mathrm{Attr}(f_i^{(l)})$ will have a large value.

We can rank the features triggered by the prompt based on $\mathrm{Attr}(f_i^{(l)})$. The target feature is usually among the top ten features. Then, our "Reinforcer" is used to interpret them, thereby identifying the target feature. When editing features, we set the sparse value of the feature to a constant $c_i^{(l)} = C$. The larger the value, the stronger the concept.

Our pipeline significantly optimizes the process of feature interpretation, localization and editing using SAEs.

3.2 Unifying Different Concept Editing Methods at the Neuron Level

Neurons Located by SAE According to the structure of transformer (Vaswani, 2017), changes to the residual stream of LLMs can correspond to changes to the neurons:

$$\boldsymbol{W}_{\text{down}}^{l}\boldsymbol{n}^{(l)} + \boldsymbol{h}^{(l-1)} + \mathbf{att}^{(l)} = \boldsymbol{h}^{(l)}, \qquad (11)$$

where, $\boldsymbol{W}_{\text{down}}^{l}$ is the second parameter matrix of FFN module, $\boldsymbol{n}^{(l)}$ represents the intermediate neuron vector in the l-th FFN module, $\boldsymbol{h}^{(l-1)}$ is residual stream from the previous layer, $\mathbf{att}^{(l)}$ is the output of self-attention module, and $\boldsymbol{h}^{(l)}$ is residual stream of the current layer.

When editing, $\boldsymbol{h}^{(l)}$ is reconstructed to $\boldsymbol{h}_{\text{rec}}$, then:

$$\boldsymbol{W}_{\text{down}}^{l}\boldsymbol{n}_{\text{rec}}^{(l)} + \boldsymbol{h}^{(l-1)} + \mathbf{att}^{(l)} = \boldsymbol{h}_{\text{rec}}, \qquad (12)$$

where, $\boldsymbol{n}_{\text{rec}}^{(l)}$ represents the value of neurons when the residual stream is $\boldsymbol{h}_{\text{rec}}$. In this process, the rate of neurons chage is

$$\Delta \boldsymbol{n}_{\text{SAE}} = \frac{\boldsymbol{n}_{\text{rec}}^{(l)} - \boldsymbol{n}^{(l)}}{\boldsymbol{n}^{(l)}}, \qquad (13)$$

which indicates the multiplicative factor applied to neurons to achieve the desired control over a concept. Clearly, the larger the rate of change, the more significant the role neurons play in controlling this concept. We select the top J neurons with the highest rates of change:

$$n_{\text{SAE}} = \text{argsort}\, \Delta \boldsymbol{n}_{\text{SAE}}[-J:], \qquad (14)$$

where J is the amount of neurons located by SAE, and n_{SAE} denotes their indices.

Neurons Located by Steering Vector Steering vector also constitutes a modification to residual stream. Therefore, the localization of neurons is similar to SAE. The detailed localization process is provided in Appendix A.3.

Neurons Located by SFT We compute the rate of change for each column of the weight matrix:

$$\Delta w_j = \frac{\|\boldsymbol{W}_{\text{sft},j} - \boldsymbol{W}_j\|_2}{\|\boldsymbol{W}_j\|_2}, \qquad (15)$$

where, $\boldsymbol{W}_j$ and $\boldsymbol{W}_{\text{sft},j}$represent the j-th column of $\boldsymbol{W}$ and $\boldsymbol{W}_{\text{sft}}$, respectively, $\|\cdot\|_2$ denotes the L2 norm. We then sort the columns based on Δw_j and select the top J columns with the highest rates of change; see Appendix A.3 for details.

4 Experiment

Using emotion as the concept, we analyzed the performance of our interpretable feature localization for SAE, editing reliability, output generalization and neuron-level consistency, and also conducted analysis at the mathematical level. This forms a progressively deepening analytical paradigm for concept editing methods.

4.1 Setup

Models We conducted experiments on Gemma-2-2B (Rivière et al., 2024) and LLaMA-3-8B (Dubey et al., 2024), which contains diverse parameter scales. We choose their corresponding Sparse Autoencoders: gemma-scope-2b-pt-res[1](Lieberum et al., 2024) and sae-llama-3-8b-32x.[2]

Dataset We used the emotion dataset introduced in (Zou et al., 2023). The dataset contains six categories of emotions—happiness, sadness, anger, fear, surprise, and disgust, and comprises 1,200 sentences, with each emotion category consisting of 200 sentences. Based on this dataset, we utilized GPT-4 to generate 200 sentences for each emotion category (Long et al., 2024), thereby expanding the original dataset to a total of 2400 sentences. (More details are shown in Appendix A.4)

Metrics We employed GPT-4 to assess the emotional intensity of the generated sentences (Zheng et al., 2023). Specifically, GPT-4 was used to score the emotional intensity of the generation before and

[1] https://huggingface.co/google/gemma-scope-2b-pt-res

[2] https://huggingface.co/EleutherAI/sae-llama-3-8b-32x

Emotion	Shallow		Middle		Deep	
	L8B	G2B	L8B	G2B	L8B	G2B
anger	-	-	97546	-	57280	15538
happiness	-	-	36205	-	59512	12780
sadness	-	3940	51331	4657	91978	8554
surprise	-	-	6699	-	121074	11874
disgust	-	-	120765	-	48255	4513
fear	-	-	55888	-	15606	8813

Table 1: The detected feature IDs in Llama-3-8B (L8B) and Gemma-2-2B (G2B) with our proposed method. The number represents the index of the feature in the sparse activation space of SAE. "-" indicates that no corresponding features are found in that layer. L8B has 32 layers, with the 5th, 20th, and 30th layers representing shallow, middle, and deep layers. G2B has 26 layers, with the 5th, 15th, and 24th layers representing the same (layer indices start from 0).

after concept editing, with scores ranging from 1 to 10. When performing the operation to enhance a specific emotion, if the edited model generates sentences with higher emotional scores, it indicates that the editing method is effective. Additionally, considering the positional bias in sentence scoring by GPT-4 (Wang et al., 2024), we averaged the scores of the original positions and the swapped positions. The prompt used for this evaluation is provided in Appendix A.5. In addition, we manually assessed the results, which were found to be comparable to those of GPT-4. Further details are provided in Appendix A.6.

4.2 Performance of Our Interpretable Feature Localization for SAEs

Using the feature localization method we proposed in Section 3.1, we detected the features corresponding to the six emotions as shown in Table 1. Our proposed method locates features with high accuracy. We sampled 100 sentences and ranked the triggered features according to $\mathrm{Attr}(f_i^{(l)})$ for each sentence. The target feature appears in the top 10 with a probability of 79.8%. Therefore, for any target concept, constructing a minimal amount of data in the *prompt + golden answer* format allows us to identify the target feature.

From Table 1, it can be observed that not all layers are capable of identifying the target features. For Llama-3-8B, target features are identifiable in middle and deep layers, but not in shallow layers. In contrast, Gemma-2-2B can identify target features in deep layers, with minimal presence in middle and shallow layers. The reasons for this can be summarized as follows.

	NE	SAE	SV	SFT
Llama-3-8B				
layer-30 (Deep)				
anger	36.95	**76.35**	46.80	15.27
happiness	51.50	65.00	**84.00**	47.00
sadness	51.23	**67.00**	59.61	43.35
surprise	49.50	**69.00**	55.50	45.00
disgust	36.95	**78.82**	59.11	24.63
fear	53.20	**84.73**	65.35	41.87
Average	46.56	**73.48**	61.73	36.19
layer-20 (Middle)				
Average	34.48	**97.54**	82.76	25.62
layer-5 (Shallow)				
Average	37.93	-	100	38.91
Gemma-2-2B				
layer-24 (Deep)				
Average	48.28	**95.57**	47.29	26.60
layer-15 (Middle)				
Average	45.32	-	98.52	26.11
layer-5 (Shallow)				
Average	45.81	-	100	33.50

Table 2: The editing effect (%) of NE, SAE, SV and SFT. The numbers represent the percentage of data successfully controlled in the test set. The "-" indicate cases where SAE fails to locate features.

First, the concept formation varies across layers. Shallow layers extract low-level lexical features, making it difficult to capture complex semantics. Middle layers begin integrating semantic information, where meaningful features start to emerge, while in deep layers, these features become mature. As shown in Appendix A.7, the features in shallow layers mainly consist of word fragments, auxiliary words, symbols and simple words. As the layers deepen, the features become complex.

Second, the model size and capacity on generation matter, as they significantly influence the model's ability to capture complex features. Larger models, such as Llama-3-8B, exhibit superior performance in modeling intricate semantic features across layers. Conversely, smaller models, such as Gemma-2-2B, often struggle to form complete semantic representations across all layers, which explains why Gemma-2-2B fails to capture relevant concepts even in its middle layers.

4.3 Performance of Editing Reliability

Experimental details are shown in Appendix A.8. The results are summarized in Table 2, and the numbers represent the percentage of data success-

Type	Words
Synonyms	rage, irritation, fury, mad
Antonyms	calmness, joy, contentment, happy
Related	frustration, argument, resentment, yelling
Unrelated	banana, laptop, galaxy, violin

Table 3: The synonyms, antonyms, related words (words that often appear with "anger"), and unrelated words of "anger".

Type	Original	NE	SAE	SV
Anger	9.12E-07	6.65E-06 ✓	1.01E-03 ✓	8.37E-06 ✓
Synonyms	2.57E-06	4.62E-06 ×	5.66E-04 ✓	2.26E-05 ✓
Antonyms	1.39E-03	4.24E-04 ✓	5.67E-04 ✓	1.62E-04 ✓
Related	6.12E-07	6.04E-07×	3.31E-05 ✓	1.22E-05 ✓
Random	2.45E-06	9.14E-07	9.14E-07	1.07E-06

Table 4: The average probabilities of these words among the 20 words generated by the model. ✓indicates that the editing method is valid, while × indicates it is invalid.

fully controlled in the test set. The intuitive results can be found in Appendix A.9.

We observe the following: First, for Llama-3-8B, in deep layers, SAE can control 73.48% of the emotions, SV can control 61.73%, while NE and SFT can only control less than 50%. We can see that SAE performs best, with SV following closely behind. Both are one level higher than NE and SFT. Similar trends are seen in other layers and in Gemma-2-2B. It shows that methods based on directly modifying the residual stream achieve better performance, especially when applied to middle layers, which is resonated with previous studies (Hase et al., 2024). In contrast, we find NE and SFT are not sensitive to the layers being modified. Since our work focuses on training-free editing methods in low-resource scenarios, we adopt a relatively simple dataset, which may partly explain the poor performance of SFT. Therefore, in the analysis in Section 4.4, we focus on NE within CAT1.

Second, SV exhibits significant effectiveness in middle layers but experiences a severe performance drop in deep layers. In comparison, SAE remains maintaining relatively strong performance in deep layers despite the increased semantic complexity, suggesting that the stability of SAE is excellent.

Third, the "-" indicate cases where SAE fails to locate relevant emotional features, as concepts have not yet fully formed in shallow layers.

4.4 Performance of Output Generalization

SAE identifies concepts, while NE identifies patterns. We applied various methods to enhance the emotion of "anger" and observed the average probabilities of the words in Table 3 over the next 20 generated time steps.

Taking the prompt "Your morning starts with your favorite breakfast. You feel" as an example, the results are presented in Table 4.

It can be seen that all methods lead to an increase in the probability of the concept words, with SAE showing the most significant effect. Moreover, they all exhibit some degree of suppression for antonyms and have little impact on unrelated words.

The focus is on synonyms and related words. NE shows no probability increase for them, indicating that it can only increase the probability of the concept words and does not induce semantic changes, which is clearly a **pattern**. Both SAE and SV show probability increases for synonyms and related words, but SAE exhibits a much larger increase, precisely controlling the target concept. We believe SAE can extract the target **concept**.

The complete generations are shown in Figure 1. After enhancing the "anger" neurons, the semantics of the output remains unchanged, still conveying a "happy" meaning, but the word "happy" is removed. This once again indicates that the neurons identified by NE do not correspond to a concept, but rather capture a pattern based on logits. It only changes the occurrence probabilities of the concept words and their antonyms. The success of NE in other tasks is largely attributable to the strong correlation between the task and the pattern, such as knowledge (Dai et al., 2022) and privacy (Wu et al., 2023, 2024). Essentially, these tasks only require changing the output probability of target words.

4.5 Performance of Neuron-level Consistency

Based on the derivations in Section 3.2, we obtained n_{NE}, n_{SAE}, n_{SV} and n_{SFT}. To ensure a fair comparison, we controlled the number of neurons in each set by setting J equal to the number of neurons identified by NE, and then computed the overlap rates among them. As shown in Table 5, n_{SAE} and n_{SV} exhibit a high degree of overlap (66.39%), while n_{NE} and n_{SFT} show a low level of overlap (6.09%). Although the overlap of n_{NE} and n_{SFT} is relatively low, it is still significantly higher than that of the other methods, showing some degree of overlap. In contrast, the neurons identified by the other pairs of methods do not overlap at all. This

	NE-SAE	NE-SV	NE-SFT	SAE-SV	SAE-SFT	SV-SFT
Llama-3-8B						
layer-30 (Deep)						
anger	0.44	0.46	**6.26**	**66.63**	2.49	2.65
happiness	0.29	0.28	**4.48**	**66.49**	2.92	2.88
sadness	0.29	0.25	**7.06**	**66.26**	2.43	2.31
surprise	0.37	0.31	**6.09**	**67.46**	2.96	2.85
disgust	0.43	0.42	**8.95**	**65.87**	2.21	2.05
fear	0.57	0.53	**3.69**	**65.60**	3.02	2.89
Average	0.40	0.38	**6.09**	**66.39**	2.67	2.61
layer-20 (Middle)						
Average	1.03	0.96	**9.26**	**65.27**	1.24	1.29
layer5 (Shallow)						
Average	-	0.08	**4.93**	-	-	1.84
Gemma-2-2B						
layer-24 (Deep)						
Average	0.77	0.78	**6.22**	**67.76**	4.57	4.45
layer-15 (Middle)						
Average	-	0.60	**5.01**	-	-	4.35
layer-5 (Shallow)						
Average	-	0.08	**7.73**	-	-	3.33

Table 5: The overlap (%) of neurons located by NE, SAE, SV and SFT. For example, 0.44 denotes that the overlap of neurons located by NE and SAE is 0.44%.

result supports the validity of our classification philosophy.

We analyze this variable Δn to explain how different methods influence the generation process. According to Eq (11), Eq (13) and Eq (22):

$$\Delta \boldsymbol{n}_{\text{SAE}} = \frac{\boldsymbol{W}^{+}_{\text{down}}(\boldsymbol{h}^{(l)}_{\text{rec}} - \boldsymbol{h}^{(l)})}{\boldsymbol{n}^{(l)}}, \tag{16}$$

$$\Delta \boldsymbol{n}_{\text{SV}} = \frac{\boldsymbol{W}^{+}_{\text{down}}\boldsymbol{v}^{(l)}}{\boldsymbol{n}^{(l)}}, \tag{17}$$

where, $\boldsymbol{W}^{+}_{\text{down}}$ is pseudo-inverse matrix of $\boldsymbol{W}_{\text{down}}$, the only variable that doesn't change with time step in the formula. $\Delta \boldsymbol{n}_{\text{SAE}}$ and $\Delta \boldsymbol{n}_{\text{SV}}$ are both the basis for neuron localization and the multiplicative factors for neuron manipulation. We know that at each inference time, $\Delta \boldsymbol{n}_{\text{SAE}}$ and $\Delta \boldsymbol{n}_{\text{SV}}$ change dynamically. However, since $\boldsymbol{v}$ is an invariable vector, and $\boldsymbol{h}^{(l)}_{\text{rec}} - \boldsymbol{h}^{(l)}$ changes at each time step, the neurons located by SAE and the manipulation to them are more flexible compared to SV. This explains why the editing effect of SAE is better. In contrast, the methods of CAT1 are static, and their performance tends to be inferior compared to the dynamic methods in CAT2.

In addition, the identified neurons can serve as a mechanism to continuously track the evolution of a given concept throughout model optimization. Upon detecting a conceptual shift, the locate-and-edit procedure is automatically executed, thereby ensuring the long-term robustness of concept edits.

4.6 Mathematical Analysis

In LLMs, the prediction of next token is performed by projecting the hidden state $\boldsymbol{h}$ linearly onto the vocabulary space. Therefore, the most effective way to evaluate a concept editing method is to analyze the information flow it injects into the model, i.e., $\Delta \boldsymbol{h}$. To further clarify this perspective, the mathematical analysis in Appendix A.10 illustrates how different concept editing methods can be aligned and compared within a unified framework based on residual streams and matrix operations. This analysis highlights that the key advantage of SAE over other methods lies in projecting hidden states into a high-dimensional concept space, enabling the injection of richer and more precise information into the model.

4.7 General Abilities

To investigate whether concept editing methods impairs the model's general abilities, we conducted evaluations on five widely-used benchmarks. The specific benchmarks and evaluation results are provided in Appendix A.11, which reveals that these methods rarely impacts the general ability of the LLMs, no matter they are methods from our category 1 or 2. Surprisingly, in some cases, it even leads to a slight performance improvement.

5 Related Work

Concept editing is used to identify and modify concepts within LLMs in order to control their outputs. The commonly used methods can be categorized as follows.

Neuron Editing Geva et al. (2021) show that feedforward layers in transformer-based language models operate as key-value memories, where each key correlates with textual patterns in the training examples, and each value induces a distribution over the output vocabulary. Based on this finding, a series of studies (Dai et al., 2022; Wu et al., 2023; Chen et al., 2024; Leng and Xiong, 2025; Shi et al., 2025) leverage knowledge neurons to edit specific factual knowledge. Lai et al. (2024) identify "styles" neurons and enhance the stylistic diversity of the generated text. Wang et al. (2022) find skill neurons and explore their applications. Zhao et al. (2024) allow fine-tuning of language-specific neurons, enhancing multilingual abilities in a specific language.

Sparse Autoencoders One of the roadblocks to a deep understanding of neural networks' internals is polysemanticity (Elhage et al., 2022), where neurons appear to activate in multiple and semantically distinct contexts (Scherlis et al., 2022). Huben et al. (2023) use sparse autoencoders to reconstruct the internal activations of language models. These autoencoders learn sets of sparsely activating features that are more interpretable and monosemantic. Subsequently, multiple groups have open-sourced their own trained SAEs, such as EleutherAI[3], Openai (Gao et al., 2024), Google (Lieberum et al., 2024) and so on. Paulo et al. (2024) build an open-source automated pipeline to generate and evaluate natural language explanations for the features of SAEs using LLMs, but it requires substantial cost.

Steering Vector This concept involves the extraction and optimization of latent space representation vectors from datasets. These vectors capture key attributes and can be directly manipulated to control the attributes of the generated text (Liang et al., 2024). Subramani et al. (2022) extract latent steering vectors from pretrained language models to control text generation, these vectors are then injected into the model's hidden states. Tigges et al. (2023) show that emotion is represented linearly and capture directions of emotion.

6 Conclusion

This paper presents a unified neuron-level paradigm for concept editing in large language models, offering a common framework to understand and compare diverse editing methods. By analyzing how conceptual information is injected, either indirectly through parameter modification or directly via residual stream manipulation, we classify representative methods such as NE, SFT, SAE, and SV into two coherent categories. We further conduct a systematic evaluation along four dimensions: editing reliability, output generalization, neuron-level consistency, and mathematical formalization. Our findings show that direct methods, especially SAE, achieve superior performance in both reliability and semantic alignment. To address the computational limitations of existing SAE approaches, we additionally propose an efficient feature interpretation method that improves practicality without compromising effectiveness. Overall, our work bridges fragmented concept editing strategies, deepens the understanding of their internal mechanisms, and contributes practical advances for controllable and interpretable LLMs.

Limitations

First, due to computational constraints and few open-source SAEs available, this study could use only two models for experiments so far. While it covers both 2B and 8B sizes, examining more models under the proposed framework would create opportunities for more interesting findings and insights. Second, we have selected 6 emotions as example concepts for the empirical verification of our framework. In the future, we would like to explore more concepts in a wide range of scenarios.

Acknowledgements

The present research was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0203000). We would like to thank the anonymous reviewers for their insightful comments.

References

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Journey to the Center of the Knowledge Neurons: Discoveries of Language-Independent Knowledge Neurons and Degenerate Knowledge Neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17817–17825.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *CoRR*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge Neurons in Pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502.

Weilong Dong, Xinwei Wu, Renren Jin, Shaoyang Xu, and Deyi Xiong. 2025. Contrans: Weak-to-strong alignment engineering via concept transplantation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4130–4148.

[3] https://blog.eleuther.ai/autointerp/

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 Herd of Models. *CoRR*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger B. Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy Models of Superposition. *CoRR*.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. SCALING AND EVALUATING SPARSE AUTOENCODERS. In *The Thirteenth International Conference on Learning Representations*.

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.

Akshat Gupta, Dev Sajnani, and Gopala Anumanchipalli. 2024. A Unified Framework for Model Editing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15403–15418.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does Localization Inform Editing? Surprising Differences in Causality-Based Localization vs. Knowledge Editing in Language Models. *Advances in Neural Information Processing Systems*, 36.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2023. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *The Twelfth International Conference on Learning Representations*.

Wen Lai, Viktor Hangya, and Alexander Fraser. 2024. Style-Specific Neurons for Steering LLMs in Text Style Transfer. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13427–13443.

Yongqi Leng and Deyi Xiong. 2025. Towards Understanding Multi-Task Learning (Generalization) of LLMs via Detecting and Exploring Task-Specific Neurons. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2969–2987.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-Time Intervention: Eliciting Truthful Answers from a Language Model. *Advances in Neural Information Processing Systems*, 36.

Xun Liang, Hanyu Wang, Yezhaohui Wang, Shichao Song, Jiawei Yang, Simin Niu, Jie Hu, Dan Liu, Shunyu Yao, Feiyu Xiong, and Zhiyu Li. 2024. Controllable Text Generation for Large Language Models: A Survey. *CoRR*.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 278–300.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11065–11082.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022a. 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. 2022b. Mass-Editing Memory in a Transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast Model Editing at Scale. In *International Conference on Learning Representations*.

Neel Nanda. 2023. Attribution Patching: Activation Patching At Industrial Scale. *https://www.neelnanda.io/mechanistic-interpretability/attribution-patching*.

Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. 2024. Automatically Interpreting Millions of Features in Large Language Models. In *Forty-second International Conference on Machine Learning*.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjösund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. 2024. Gemma 2: Improving Open Language Models at a Practical Size. *CoRR*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and Capacity in Neural Networks. *CoRR*.

Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. 2023. Large Language Model Alignment: A Survey. *CoRR*.

Dan Shi, Renren Jin, Tianhao Shen, Weilong Dong, Xinwei Wu, and Deyi Xiong. 2025. IRCAN: Mitigating Knowledge Conflicts in LLM Generation via Identifying and Reweighting Context-Aware Neurons. *Advances in Neural Information Processing Systems*, 37:4997–5024.

Dan Shi, Tianhao Shen, Yufei Huang, Zhigen Li, Yongqi Leng, Renren Jin, Chuang Liu, Xinwei Wu, Zishan Guo, Linhao Yu, Ling Shi, Bojian Jiang, and Deyi Xiong. 2024. Large Language Model Safety: A Holistic Survey. *CoRR*.

Nishant Subramani, Nivedita Suresh, and Matthew E Peters. 2022. Extracting Latent Steering Vectors from Pretrained Language Models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution Patching Outperforms Automated Circuit Discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. 2023. Linear Representations of Sentiment in Large Language Models. *CoRR*.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. 2023. Activation Addition: Steering Language Models Without Optimization. *CoRR*.

A Vaswani. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large Language Models are not Fair Evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450.

Xiaozhi Wang, Kaiyue Wen, Zhengyan Zhang, Lei Hou, Zhiyuan Liu, and Juanzi Li. 2022. Finding Skill Neurons in Pre-trained Transformer-based Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11132–11152.

Xinwei Wu, Weilong Dong, Shaoyang Xu, and Deyi Xiong. 2024. Mitigating Privacy Seesaw in Large Language Models: Augmented Privacy Neuron Editing via Activation Patching. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5319–5332.

Xinwei Wu, Junzhuo Li, Minghui Xu, Weilong Dong, Shuangzhi Wu, Chao Bian, and Deyi Xiong. 2023. DEPN: Detecting and Editing Privacy Neurons in Pretrained Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2875–2886.

Shaoyang Xu, Junzhuo Li, and Deyi Xiong. 2023. Language Representation Projection: Can We Transfer Factual Knowledge across Languages in Multilingual Language Models? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3692–3702.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Yiran Zhao, Wenxuan Zhang, Guizhen Chen, Kenji Kawaguchi, and Lidong Bing. 2024. How do Large Language Models Handle Multilingualism? *Advances in Neural Information Processing Systems*, 37:15296–15319.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. 2023. Representation Engineering: A Top-Down Approach to AI Transparency. *CoRR*.

A Appendix

A.1 Reinforcer

Mathematical Principles Our work introduces SAE Reinforcer to interpret features. Its core idea is to manually amplify a specific feature $\boldsymbol{c}_k$ (corresponding to a concept like "anger") by setting its activation to an extremely large value C: $\boldsymbol{c}'_k = C$, where $\boldsymbol{c}$ represents the sparse representation of SAE, $\boldsymbol{c}'$ is the modified sparse representation. Other features remain unchanged:$\boldsymbol{c}'_j = \boldsymbol{c}_j, \forall j \neq k$. Then, the decoder processes this modified activation:

$$\begin{aligned} \boldsymbol{h}_{\text{rec}} &= \boldsymbol{W}_2 \boldsymbol{c}' + \boldsymbol{b}_2 \\ &= \boldsymbol{W}_{2,k} C + \sum_{j \neq k} \boldsymbol{W}_{2,j} \boldsymbol{c_j} + \boldsymbol{b}_2 \end{aligned} \tag{18}$$

Since C is significantly larger than the original activation, the reconstructed $\boldsymbol{h}_{\text{rec}}$ is dominated by $W_{2,k}C$:

$$\boldsymbol{h}_{\text{rec}} \approx \boldsymbol{W}_{2,k} C \tag{19}$$

In a Transformer model, hidden states are ultimately projected onto the vocabulary space for word prediction. As a result, the language model is biased toward generating words that are strongly associated with the feature $\boldsymbol{c}_k$. In extreme cases, the model will repeatedly generate a single word or phrase representing the concept, such as "anger".

Performance of Reinforcer The explanation obtained using the autointerpretability procedure (Huben et al., 2023) and the explanation of the concept obtained using our Reinforcer are consistent.

Taking the feature "anger" (#15538) in layer 24 of Gemma-2-2b as an example, We randomly selected some texts from Wikipedia, segmented them into sentences, and identified the sentences corresponding to the 20 tokens with the highest activations for this feature in the high-dimensional sparse space of SAE (see Table 6). We then asked GPT-4 to summarize the common characteristics of these sentences and assign a name to Feature #15538.

Answer by GPT-4: "A suitable name for Feature 15538 could be Emotional Intensity, capturing the prominence of strong emotions such as anger, rage, and the reactions they provoke in the characters." The results are consistent with those obtained using the reinforcer in our work. Additionally, similar experimental results were observed for other models and concepts, which are not elaborated here.

A.2 Feature Localization Details

We extend Eq (8) to the case where the length of the golden answer is greater than one:

$$\text{Attr}(f_i^{(l)}) = \frac{c_i^{(l)}}{T_g} \sum_{t=1}^{T_g} \frac{\partial \text{P}_{y_t}(c_i^{(l)})}{\partial c_i^{(l)}}. \tag{20}$$

When our method shown in Eq (9) expanded to the case where the length of the golden answer is greater than one:

$$\begin{aligned} \text{Attr}(f_i^{(l)}) &= -c_i^{(l)} \frac{\partial \text{Loss}}{\partial c_i^{(l)}} \\ &= \frac{c_i^{(l)}}{T_g} \frac{\partial \sum_{t=1}^{T_g} \log \text{P}_{(y_t)}(c_i^{(l)})}{\partial c_i^{(l)}}. \end{aligned} \tag{21}$$

In theory, the trend of value changes for these two formulas is consistent. However, in Eq (20), starting from $t = 2$, $\frac{\partial \text{P}_{y_t}(c_i^{(l)})}{\partial c_i^{(l)}}$ becomes very small, because the probability of subsequent tokens is less related to $c_i^{(l)}$, which causes Eq (20) to fail in accurately locating the feature. Therefore, Eq (20) is actually only applicable to the case where $T_g = 1$. In contrast, Eq (21) does not encounter this issue and can accurately locate the feature.

A.3 Neurons Selection

Neurons Located by Steering Vector The rate of change in the values of neurons identified by SV is denoted as:

$$\Delta \boldsymbol{n}_{\text{SV}} = \frac{\boldsymbol{n}_{\text{SV}}^{(l)} - \boldsymbol{n}^{(l)}}{\boldsymbol{n}^{(l)}}, \tag{22}$$

where, $n_{\text{SV}}^{(l)}$ represents the value of neurons when the residual stream is $\boldsymbol{h}_{\text{SV}}$. We then sort the neurons and select the top J neurons with the highest rates of change:

$$n_{\text{SV}} = \text{argsort}\, \Delta \boldsymbol{n}_{\text{SV}}[-J:], \tag{23}$$

where J is the amount of neurons located by SV, and n_{SV} is the set of indices of the important neurons identified by SV.

Supervised Fine-tuning We collect the change scores into a vector $\Delta \mathbf{w} = [\Delta w_1, \ldots, \Delta w_{d_{\text{ffn}}}]^\top$. We then rank the columns by their scores and select the top J with the largest changes:

$$n_{\text{SFT}} = \text{argsort}\, \Delta \mathbf{w}[-J:], \tag{24}$$

where J indicates the amount of neurons located by SFT, and n_{SFT} is the set of indices of the important neurons identified by SFT, since the columns of $\boldsymbol{W}$ correspond one-to-one with the neurons.

Activation	Sentence
124.01	When Admetus angered the goddess Artemis by forgetting to give her the due offerings, Apollo came to the rescue and calmed his sister.
116.42	According to another version, or perhaps some years later, when Zeus struck down Apollo's son Asclepius with a lightning bolt for resurrecting the dead, Apollo in revenge killed the Cyclopes, who had fashioned the bolt for Zeus.
111.37	Rand was unimpressed by many of the NBI students and held them to strict standards, sometimes reacting coldly or angrily to those who disagreed with her.
110.27	In one essay, political writer Jack Wheeler wrote that despite the incessant bombast and continuous venting of Randian rage, Rand's ethics are a most immense achievement, the study of which is vastly more fruitful than any other in contemporary thought.
105.42	Cain then killed Abel out of jealousy.

Table 6: Activation scores and corresponding sentences.

A.4 Dataset Details

During our experiments, we reconstructed the data using multiple prompt templates to ensure that the identified neurons or features are independent of any specific prompt. For example, "*scenario*, You feel", "The scenario is: *scenario*. The emotion in the above scenario is" and so on. The *scenario* is an item in dataset.

A.5 Prompt Format

You're a good assistant at evaluating the emotion of a text.
Now you have two sentences, you are asked to assess the degree of "emotion" in both sentences.
Each sentences receives an overall score on a scale of 0 to 10, where a higher score indicates higher the level of emotion.
[The Start of Sentences 1]
A1
[The End of Sentences 1]
[The Start of Sentences 2]
A2
[The End of Sentences 2]
First, provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment.
Then, give a overall score on a scale of 0 to 10 for the two answers, and this score is an integer.
Output with the following format:
Evaluation evidence: <evaluation explanation here>
The overall score of Sentence 1: <score>
The overall score of Sentence 2: <score>

A.6 Human Evaluations

We have re-sampled 10 sets of experimental results, with 40% of the samples (80 data points) in each result manually verified. The number of cases where the GPT-4 evaluation is consistent with the manual review are: 78, 75, 70, 69, 73, 78, 75, 70, 72, 74. The average consistency rate is 91.75%, which is highly consistent, so the experimental results remain unchanged.

A.7 Features in Different Layers

Table 7 shows the features in shallow, middle and deep layers.

A.8 Experiments Set

We divided the dataset into "localization", "validation" and "test" in a ratio of 1:1:2. This partition is based on the observation that getting neurons, features, and steering vectors requires relatively few data samples, while the primary focus lies in evaluating the effect and general abilities of the editing methods, necessitating a larger test set.

For NE, we utilized the localization set to identify neurons associated with a specific emotion. Directly calculating continuous integrals in Eq (2) is intractable. We use Riemann approximation $\text{Attr}(\boldsymbol{w}_i^{(l)}) = \frac{\overline{\boldsymbol{w}}_i^{(l)}}{m} \sum_{k=1}^{m} \frac{\partial P_x(\frac{k}{m}\overline{\boldsymbol{w}}_i^{(l)})}{\partial \boldsymbol{w}_i^{(l)}}$, where $m = 20$ is the number of approximation steps. To filter out irrelevant neurons, we set a threshold t equal to 0.1 times the maximum attribution score $\text{Attr}(w_i^{(l)})$ across all neurons. A neuron is retained if its attribution score exceeds t. From the retained set, we selected 400–500 neurons that exhibit a co-occurrence frequency above a threshold q within

Layer	Feature
Deep layer	grief, pressure, solid, artificial, empathy
Middle layer	treat, smell, cave, play, anyway, ask, her
Shallow layer	://, gre, ant, pon, Dul, did, flo, klkl

Table 7: Features located at different layers.

the dataset, where q is a hyperparameter. During the forward pass, we applied a scaling operation to these selected neurons, following Eq (3), with β as a hyperparameter.

For SAE, we ranked the triggered features by the localization dataset and selected the top 10 features for validation using a reinforcer to determine the target feature. During the forward pass, we set the sparse activation of the target feature to C, where C is a hyperparameter.

For SV, we extracted the emotion direction vector using the localization dataset and applied a scaling factor k to this vector, and added it to the original residual stream during the forward pass, where k is a hyperparameter.

These editing operations were applied at every time step of the generation process. Since SFT requires a larger dataset, we expanded each emotion category to 1,200 samples using GPT-4.

The optimal parameters are selected based on the best performance on the validation set, where "best performance" not only reflects effective emotion control but also ensures the quality of the generated sentences. We incorporate mechanisms to detect repeated words and sentences, as well as perplexity (PPL) thresholds, to maintain a baseline quality of the outputs. Stricter quality checks are then applied during both LLM-based and human evaluations.

Cost comparison Our experiments of all methods can be run on a single A100 GPU with 80 GB of memory. The duration required for these experiments depends on the scale of model parameters, the size of the dataset, and the length of examples within the dataset. Overall, the time consumption is entirely acceptable. The comparison results are shown in Table 8. In summary, the cost required for the methods mentioned in our work is minimal and can be ignored.

A.9 Intuitive Edit Effect

Table 9 shows the edit results of different methods using "anger" as an example.

A.10 Mathematical Analysis

The information flow injected into the model by concept editing methods is essentially $\Delta \boldsymbol{h}$. According to Eq (11), Eq (18) and Eq (7), $\Delta \boldsymbol{h}$ made by NE, SAE, and SV:

$$\Delta \boldsymbol{h}^{\text{NE}} = \sum_{j \in n} \boldsymbol{W}_{\text{down},j} \Delta n_j, \tag{25}$$

$$\Delta \boldsymbol{h}^{\text{SAE}} = \boldsymbol{W}_{2,k} \Delta c_k, \tag{26}$$

$$\Delta \boldsymbol{h}^{\text{SV}} = k \boldsymbol{v}^{(l)}, \tag{27}$$

where n represents the neurons of NE, c_k is the sparse activation of f_k. The key distinction is that $\boldsymbol{W}_{\text{down}}$ is not explicitly optimized for semantic alignment, whereas $\boldsymbol{W}_2$ functions as an overcomplete feature dictionary whose dimensionality far exceeds that of the model and is trained to encode prior knowledge about features and the underlying concept space, thereby supplying the residual stream with richer and more informative signals. Moreover, the activation vector c in this high-dimensional space is sparse, effectively disentangling features. As a result, SAE introduces a more precise flow of information into the model. In contrast, the information flow of SV is $\boldsymbol{v}$, which is derived from the residual itself and may contain redundant information. For SFT, since parameter updates affect the entire network, the change to $\Delta \boldsymbol{h}$ is difficult to locate, meaning the influence is more diffuse and difficult to control.

A.11 General Abilities

These benchmarks are used in the widely-recognized LLM leaderboards, including HellaSwag (Zellers et al., 2019), ARC (Clark et al., 2018), MMLU (Hendrycks et al., 2021), Winogrande (Sakaguchi et al., 2021) , and TruthfulQA (Lin et al., 2022). The general abilities of models after concept editing in deep layers is shown in Table 10 .

Method	Train/Locate	Inference (Additional computation and latency)
NE	An offline computation identifies relevant neurons. In our experiment, locating 100 samples takes under 30 minutes for the 8B model.	Reweight predetermined neuron activations (computation and latency can be ignored).
SV	Computing the steering vector for 200 data points takes 25 seconds.	Add the locally stored steering vector to the residual stream (computation and latency can be ignored).
SAE	The SAE used in our work is open-source, so there is no need to consider its training cost. The identification and interpretation of features can also be completed quickly.	SAE, with just an encoder and a decoder, occupies minimal storage space. It adds three negligible steps during inference: encoding, sparse-space editing, and decoding. (computation and latency can be ignored)
SFT	The training can be completed within 30 minutes.	None

Table 8: Cost comparison of concept editing methods.

Anger	
Ori	*You discover an old family photo album you've never seen before. You feel* a sense of excitement and curiosity as you flip through the pages, wondering who these people are and what their stories might be. As you continue to explore the album, you notice that some of the photos
NE	*You discover an old family photo album you've never seen before. You feel* a sense of excitement and curiosity as you flip through the pages, wondering who these people are and what their stories might be. As you look at each picture, you notice that some of them have been
SAE	*You discover an old family photo album you've never seen before. You feel* angry and frustrated that your parents didn't share this with you when you were younger. You're upset because you feel like you missed out on something important, but you don't know what it is.
SV	*You discover an old family photo album you've never seen before. You feel* a strong emotion, and then you begin to remember... the words that were directed at you, the tone of those words, the tone of your response, the tone of the responses against you.
SFT	*You discover an old family photo album you've never seen before. You feel* a strong connection to the people in the photos, even though you don't recognize anyone.

Table 9: The outputs of L8B before and after editing using NE, SAE, SV, and SFT (taking "anger" as an example). "Ori" denotes the original output of the model. The italicized text represents the prompt.

	ARC	HellaSwag	MMLU	TruthfulQA	Winogrande	Average
Gemma-2-2B						
Original	0.8022	0.7303	0.4961	0.3623	0.6875	0.6157
NE	0.7988	0.7266	0.4945	0.3613	0.6898	0.6142
SAE	0.7572	0.7279	0.4937	0.3852	0.6701	0.6068
SV	0.7668	0.7099	0.4903	0.3697	0.6835	0.6040
SFT	0.8089	0.7335	0.4960	0.3469	0.6725	0.6116
Llama-3-8B						
Original	0.7769	0.6017	0.6218	0.4390	0.7285	0.6336
NE	0.7551	0.7851	0.6184	0.4399	0.7277	0.6652
SAE	0.7340	0.7507	0.6216	0.4375	0.6701	0.6428
SV	0.7045	0.7490	0.6098	0.4828	0.7285	0.6549
SFT	0.7811	0.7898	0.6206	0.4290	0.7088	0.6659

Table 10: Results of general abilities of LLMs after concept editing in deep layers on widely-used benchmarks. "Original" refers to the model's output before any editing.