The Knowledge Microscope: Features as Better Analytical Lenses than Neurons

Yuheng Chen[1,2], **Pengfei Cao**[1,2]*, **Kang Liu**[1,2], **Jun Zhao**[1,2]*
[1]The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
chenyuheng22@ia.ac.cn, {pengfei.cao, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Previous studies primarily utilize MLP neurons as units of analysis for understanding the mechanisms of factual knowledge in Language Models (LMs); however, neurons suffer from polysemanticity, leading to limited knowledge expression and poor interpretability. In this paper, we first conduct preliminary experiments to validate that Sparse Autoencoders (SAE) can effectively decompose neurons into features, which serve as alternative analytical units. With this established, our core findings reveal three key advantages of features over neurons: (1) Features exhibit stronger influence on knowledge expression and superior interpretability. (2) Features demonstrate enhanced monosemanticity, showing distinct activation patterns between related and unrelated facts. (3) Features achieve better privacy protection than neurons, demonstrated through our proposed FeatureErase method, which significantly outperforms existing neuron-based approaches in erasing privacy-sensitive information from LMs.[1].

1 Introduction

Language Models (LMs) have demonstrated remarkable capabilities in storing and expressing factual knowledge (Anthropic, 2024; OpenAI et al., 2024; Team et al., 2024). However, the underlying mechanisms remains unclear. Mechanistic interpretability of factual knowledge in neural networks aims to decompose these systems into interpretable units to understand how facts are stored and retrieved (Chen et al., 2024b). The critical first step in this investigation is to identify the appropriate analytical units. One mainstream approach is the neuron-based research method (Geva et al., 2021a; Dai et al., 2022; Chen et al., 2024a), which posits that LMs recall facts through multilayer perceptron

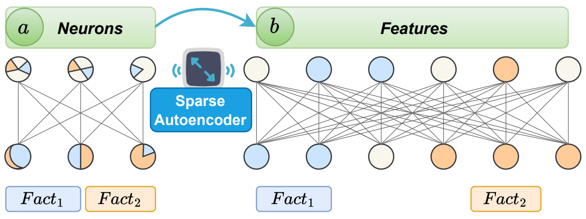

Figure 1: Comparison of research units for factual knowledge mechanisms in LMs: (a) neurons and (b) features. Colors in neurons (or features) correspond to the facts they store, illustrating how specific facts are encoded in particular units.

(MLP) weights and conceptualizes the responsible knowledge storage units as *knowledge neurons*. Although using neurons as the research unit is highly intuitive, this kind of approach still has some notable limitations (Hase et al., 2023; Niu et al., 2024; Chen et al., 2024b).

In particular, a significant issue with neuron-based approaches is the phenomenon of polysemanticity (Bricken et al., 2023; Cunningham et al., 2023), where neurons respond to mixtures of seemingly unrelated facts. Intuitively, the number of factual knowledge items stored in LMs often exceeds the number of neurons[2], necessitating that a single neuron must be associated with multiple facts. As shown in Figure 1(a), a fact may be dispersedly stored in a fragmented manner across numerous neurons, resulting in inseparable sets of neurons corresponding to $Fact_1$ and $Fact_2$. This fundamental characteristic leads to two challenges.

(1) **Limited Knowledge Expression:** Since factual knowledge can be dispersedly stored across numerous neurons, with some neurons potentially contributing only minimal information, these weak and distributed signals result in identified neurons having limited impact on knowledge expression. (2) **Poor Interpretability:** The coupling of neu-

*Corresponding authors.
[1]Code and dataset will be available.

[2]For example, Gemma-2 2B has $26 \times 9216 \approx 230k$ neurons but is trained on approximately 2 trillion tokens of data.

rons representing different facts, makes it difficult to accurately describe the function of individual neurons, hindering our ability to gain deep insights into the mechanisms of factual knowledge.

Bricken et al. (2023) suggests that polysemanticity arises from superposition, where neural networks represent more independent features through linear combinations of neurons. Their work demonstrates that Sparse Autoencoders (SAE) can effectively decompose neurons into interpretable features. As shown in Figure 1, SAE transforms a "low-dimensional" neuronal space (a) into a "high-dimensional" feature space (b), making previously inseparable problems separable. This motivates us to explore whether such transformation could benefit the understanding of factual knowledge, where the transformed feature-level units might exhibit both stronger impact on knowledge expression and superior interpretability compared to their neuron-level counterparts.

Building on this foundation, we first investigate a preliminary question: Can neurons be effectively decomposed into features in the domain of factual knowledge, and is SAE a suitable technique for this decomposition? Then, we further explore three core research questions:

Q1: Can feature-based research methods address the dual challenges of Limited Knowledge Expression and Poor Interpretability? **Q2**: Given that the key limitation of neurons lies in their polysemanticity, do features in the factual domain exhibit better monosemanticity? **Q3**: Do features outperform neurons in downstream tasks?

Our investigation into these questions yields one preliminary finding and three core findings:

Preliminary Finding (§3): **SAE demonstrates superior effectiveness in decomposing neurons into features compared to other methods**, making them suitable alternative research units for studying factual knowledge mechanisms.

Core Findings: (1) **Features as research units address the challenges of Limited Knowledge Expression and Poor Interpretability (§4.1 and §4.2)**. Through comparison of different modules (post-attention residuals, MLP activations, and post-MLP residuals), we find that features consistently show better interpretability than neurons, with post-MLP residual features having the greatest impact on knowledge expression.

(2) **Features exhibit stronger monosemanticity than neurons (§4.3)**. Features strongly activate only when encountering related facts and remain inactive for unrelated ones, resulting in distinct separation in their activation distributions. In contrast, neurons lack such separation, indicating their susceptibility to activation by unrelated facts and weaker monosemanticity, while features better align with the ideal scenario in Figure 1(b).

These two findings complement each other: the superior interpretability of features naturally arises from their stronger monosemanticity property.

(3) **Feature-based method demonstrates superior performance in knowledge erasure for privacy protection (§5)**. We propose **FeatureErase**, the first feature-based model editing method, and evaluate it on our newly constructed privacy knowledge dataset **PrivacyParaRel**. Compared to neuron-based approaches, FeatureErase achieves higher success rates in erasing privacy-sensitive knowledge from LMs while maintaining better generalization across semantically equivalent rephrase queries. Moreover, it causes less damage to the model's general capabilities.

2 Dataset, Models and Evaluation metrics

Our experiments leverage Gemma Scope (Lieberum et al., 2024), a comprehensive suite of SAEs trained on Gemma-2 models (Team et al., 2024). We study the 2B and 9B variants as they have SAEs for all layers. Regarding the dataset, consistent with other neuron-based methods (Dai et al., 2022; Chen et al., 2024a), we employ the ParaRel dataset (Elazar et al., 2021). For details to the dataset, see Table 1 in Appendix A.

We introduce two evaluation metrics most frequently used in this paper. (1) $\Delta Prob$, the decreasing value of answer probability after features/neurons ablation, which assesses the impact of knowledge storage units on knowledge expression. For neurons, we directly set their activations to zero. For features, we aim to perform a similar operation. However, since features do not exist explicitly in LMs, we propose a reconstruction-based method (detailed in Appendix C.1). Briefly, given an activation ($\mathbf{h}$), we obtain its corresponding features, set target features to zero, reconstruct the activation ($\mathbf{h}'$), and replace $\mathbf{h}$ with $\mathbf{h}'$. We then measure the change in the probability of the correct answer before ($Prob_b$) and after ($Prob_a$) ablation: $\Delta Prob = \frac{Prob_b - Prob_a}{Prob_b}$.

(2) IS, Interpretability Score, which measures the interpretability of features. We modify the method from Bills et al. (2023) to adapt it to our

task (detailed in Appendix C.2). Briefly, for features or neurons, we ask Large LMs (this paper uses gpt-4o-mini) to predict their activations. The correlation between the model's predicted activations and the actual activations is the interpretability score.

3 Preliminary Experiment

3.1 SAE Shows Superior Performance

We first address the preliminary question raised in §1: whether neurons can be decomposed into features when studying the mechanism of factual knowledge, and which method performs best. The candidate methods include: Sparse Autoencoders (SAE), Principal Component Analysis (PCA), Independent Component Analysis (ICA), and random directions (RD). The hyperparameters and methodological details are provided in Appendix D. Both PCA and ICA perform the decomposition using the same amount of data used for training SAEs.

Experiment settings The preliminary experiments focus on decomposing MLP activations to obtain features. We use pre-trained SAEs from Gemma Scope (Lieberum et al., 2024). Given MLP activations $\mathbf{h}$ at layer l, features are obtained through the encoder function:

$$\mathbf{f}(\mathbf{h}) := \sigma(\mathbf{W}_{enc}\mathbf{h} + \mathbf{b}_{enc}) \tag{1}$$

where := denotes definition, σ is the JumpReLU activation, $\mathbf{W}_{enc}$ is the encoder weight matrix, and $\mathbf{b}_{enc}$ is the encoder bias vector. Each element $f_{l,p}(\mathbf{h})$ represents the activation of the feature at layer l and position p.

Structurally, these features in SAEs parallel the role of intermediate neurons in LLMs, as SAEs are trained to reconstruct LLMs in a higher-dimensional space. Taking one MLP layer as an example, both architectures follow a similar encoder-intermediate-decoder pattern:

$$\text{LLMs}: \mathbf{x} \xrightarrow{\text{encoder}} \text{intermediate neurons} \xrightarrow{\text{decoder}} \mathbf{y} \tag{2}$$

$$\text{SAEs}: \mathbf{h} \xrightarrow{\text{encoder}} \text{SAE features} \xrightarrow{\text{decoder}} \mathbf{h}' \tag{3}$$

For a given input, we select highly activated features ($\mathbf{F_a}$) based on their spatial locations:

$$\mathbf{F_a} = \{(l, p) \mid f_{l,p}(\mathbf{h}) > \tau_1 \cdot \max_{l',p'} f_{l',p'}(\mathbf{h})\} \tag{4}$$

where τ_1 is the threshold parameter. Using the metrics ($\Delta Prob$ and IS) defined in §2, we compare

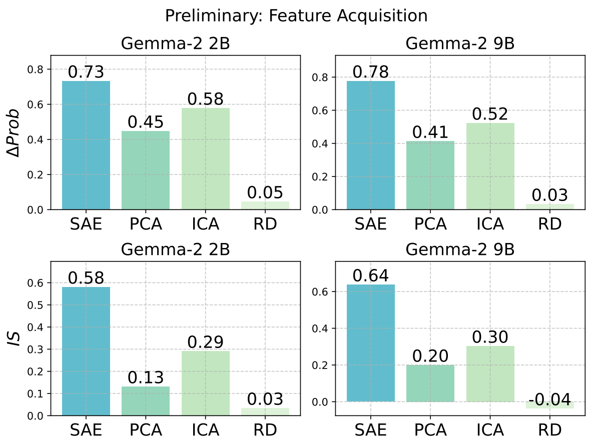

Figure 2: Evaluation of features obtained by different methods. Top: Δ Prob after feature ablation. Bottom: Interpretation scores (IS). Higher values indicate better performance in both metrics.

our SAE-based approach with baseline methods. The results are presented in Figure 2.

Findings SAE demonstrates superior effectiveness in decomposing neurons into features compared to other methods. For example, as shown in Figure 2, in Gemma-2 9b, SAE features demonstrate superior performance in both $\Delta Prob$ and IS, achieving $\Delta Prob$ of $\sim$ 0.78 and IS of $\sim$ 0.64, showing increases of $\sim 1.3\times$ in $\Delta Prob$ and $\sim 2\times$ in IS over the strongest baseline (ICA). Paired t-tests also confirm that SAE features are more effective for studying factual knowledge mechanisms (see Table 2 in Appendix E).

3.2 Feature Distribution Patterns Remain Consistent Across Feature Numbers

When extracting features using SAE, we need to determine how many features to use for reconstructing LLM representations. As shown in Equation 2, while LLMs use a fixed number of neurons, SAEs map these representations to a higher-dimensional feature space. The number of features (N) determines this dimensionality, typically set as $N = n \times d_{model}$ where n is a multiplier and d_{model} is the model's hidden dimension (9216 for Gemma-2 2B). Adjusting N requires resource-intensive retraining. Notably, we observe consistent feature clustering patterns across different values of N.

In Figure 3, we visualize feature distributions across four different settings ($N = n \times 9216$) using Gemma-2 2B. The vertical axis represents layers, while the horizontal axis shows positions. The color intensity indicates the feature density in each bin, with darker blue representing higher

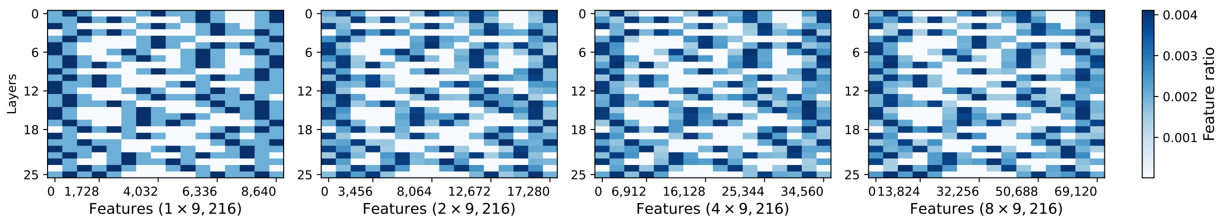

Figure 3: Distribution plots of activated features under different feature number settings ($n \times 9216, n = 1, 2, 4, 8$) for Gemma-2 2B. The similar distribution patterns across different n suggest that features consistently fall into similar regions. It should be noted that these four pictures are not exactly the same, but they are very similar.

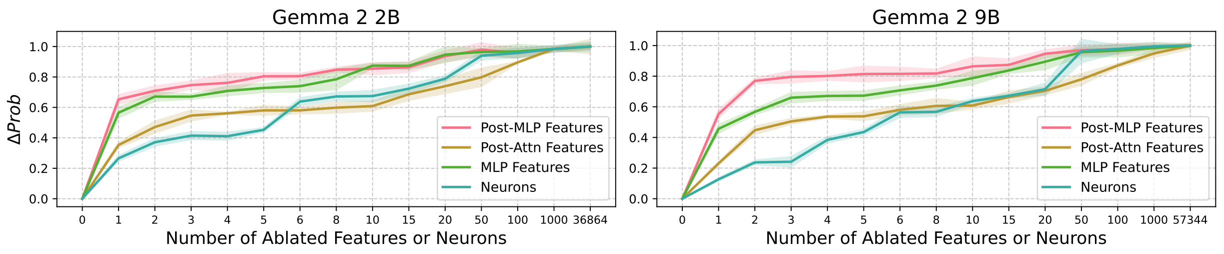

Figure 4: The impact on $\Delta Prob$ when ablating features from different transformer components and neurons. Values show mean ± standard error across 5 bootstrap iterations, with higher values indicating greater influence on knowledge expression. Note that while $\Delta Prob \in [0, 1]$, the plots may exceed 1 due to + std.

values. Using 500 randomly sampled facts[3], we observe that **features consistently cluster in similar regions** when fixing the number of bins. As N increases, these features undergo hierarchical decomposition within their original clusters, with larger N values enabling finer-grained representations while maintaining the same overall distribution structure.

The results for Gemma-2 9B, along with a comprehensive quantitative analysis of the entire dataset presented in Appendix F (Table 3 and Figure 11), corroborate these findings. One possible explanation could be that SAE features are insensitive to N. Based on this hypothesis, we fix $N = 4 \times 9216$ for subsequent experiments, eliminating the need to compare different N values in each experiment.

4 Features vs. Neurons

Building upon §3, this section delves into the mechanism of factual knowledge using features obtained through sparse autoencoders (SAE). Our findings address the research questions Q1 (§4.1 and §4.2) and Q2 (§4.3) raised in §1.

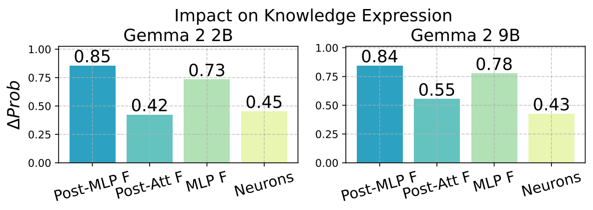

Figure 5: The impact on $\Delta Prob$ when ablating features from different transformer components and neurons.

4.1 Post-MLP Features Have the Strongest Impact on Knowledge Expression

Experiment Settings Following §3, we extend our analysis to three components in transformer: post-attention residual, MLP activation, and post-MLP residual. We apply SAE to extract features from each component and compare them with knowledge neurons identified using the localization method proposed by Chen et al. (2024a) (detailed in Appendix G), as their approach achieves state-of-the-art performance. We employ the $\Delta Prob$ metric (i.e., the decrease in model prediction probability) from §2 and conduct two complementary analyses to compare how features and neurons impact knowledge expression, with results shown in Figure 5 and Figure 4.

(1) Figure 5: We select features and neurons through a thresholding method on the full dataset,

[3]This sampling is necessary as visualizing features from the entire dataset would result in near-complete coverage of the feature space.

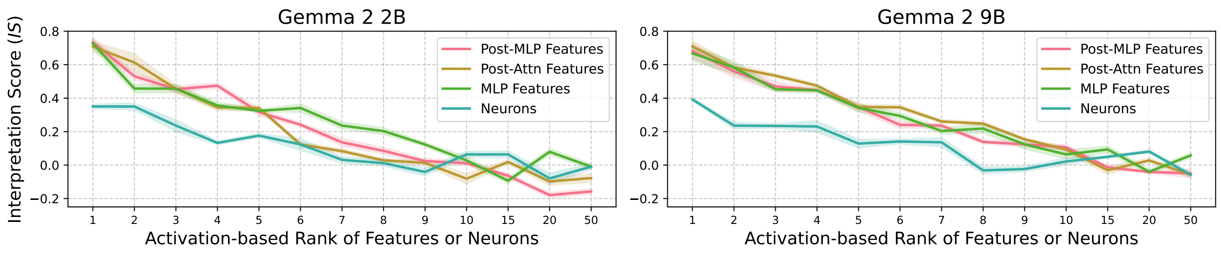

Figure 6: Per-unit interpretation scores (IS) for features from different transformer components and neurons. We use the same bootstrap samples as Figure 4. Higher scores indicate better interpretability.

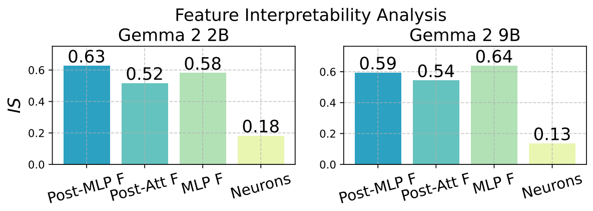

Figure 7: The average interpretability scores (IS) for features from different components and neurons.

then ablate them and calculate the $\Delta Prob$. For features, the selection follows Equation 4, and a similar thresholding technique with τ_1 is applied for neurons.

(2) Figure 4: We perform a fine-grained analysis by ranking features based on their activations in descending order and progressively ablating them, calculating $\Delta Prob$ at each step. This approach allows us to observe the continuous impact of feature ablation without being constrained by any predetermined threshold. Since progressive feature ablation on the full dataset is computationally intensive, we employ bootstrap sampling with 5 independent iterations (300 instances each, with replacement).

Findings Post-MLP features have the strongest impact on knowledge expression. In Figure 5, the post-MLP features show a substantial impact with $\Delta Prob$ of ~ 0.85, which is approximately 10% higher than the strongest baseline (MLP features) and $\sim 1.9\times$ that of neurons. Figure 4 provides more granular evidence, showing that ablating just a few highly-activated features significantly impairs the model's ability to express knowledge. Ablating a single post-MLP feature yields a $\Delta Prob$ of ~ 0.6, substantially higher than the strongest baseline (MLP features) and $\sim 3\times$ that of neurons. The statistical significance test results are in Appendix H.1 (Table 3), confirming that the superior knowledge expression capabilities of post-MLP features over other features (or neurons) are significant.

4.2 Features Demonstrate Superior Interpretability Compared to Neurons

Experiment Settings We employ the interpretability score (IS) metric introduced in §2 to evaluate the features and neurons, with results shown in Figure 7 and Figure 6.

(1) Figure 7: After obtaining features and neurons through the thresholding technique, we evaluate the interpretability scores (IS) of these selected units on the full dataset.

(2) Figure 6: We first rank features in descending order based on their activations using the same 1000 sampled facts, then evaluate IS for each feature/neuron individually. Unlike the batch ablation analysis in Figure 4, this approach assesses one unit at a time. In this analysis, evaluating units individually holds greater significance beyond merely eliminating the influence of threshold. Since a single fact typically activates a larger number of neurons (can reach 20 or more) compared to features, averaging IS across neurons would bias the score toward lower values. Therefore, evaluating each unit individually ensures a more equitable comparison. Note that we only evaluate up to the 50th unit, as the IS approaches or falls below zero near this point, making further evaluation unnecessary.

Findings (1) **Features demonstrate superior interpretability compared to neurons.** In Figure 7, post-MLP features achieve IS values of ~ 0.6, $\sim 4\times$ that of neurons. A fine-grained analysis in Figure 6 further validates this conclusion, showing that even when compared individually, highly activated neurons consistently exhibit lower IS values (< 0.4) than highly activated features (~ 0.7).

(2) Post-MLP features are a better choice when considering both metrics. While MLP and post-attention features show comparable interpretability scores ($IS \sim 0.6$ and ~ 0.5 respectively), post-MLP features consistently perform well in both interpretability and knowledge expression. The

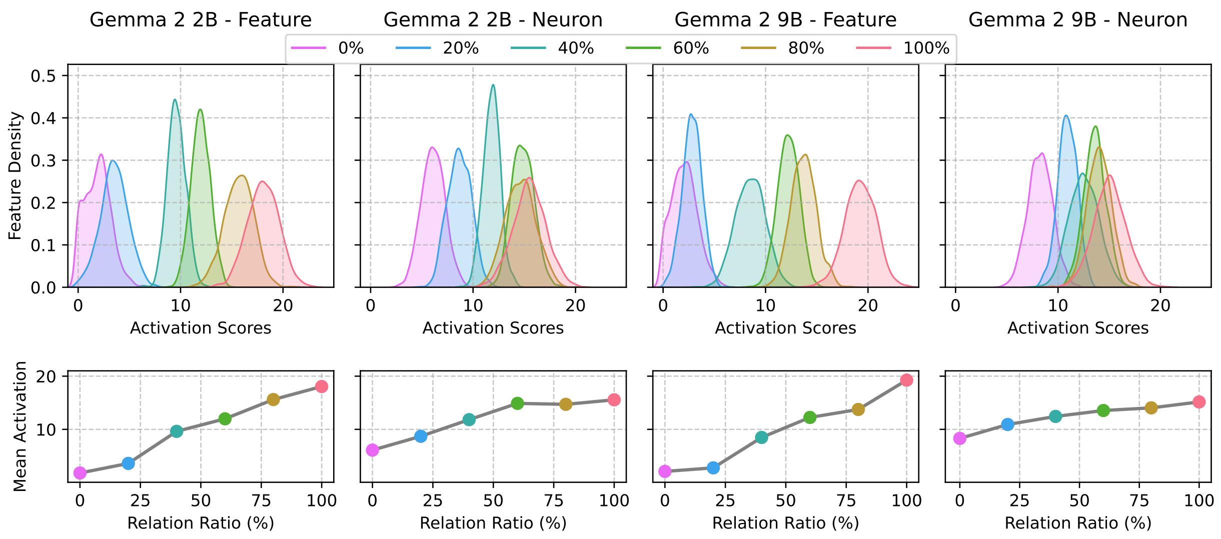

Figure 8: Post-MLP feature activations corresponding to relation-facts under varying input compositions (0% to 100% relation-facts). Top: Activation score distributions. Bottom: Mean activation values.

statistical significance test results are presented in Appendix H.1 (Table 3), showing that post-MLP features significantly outperform neurons in interpretability, and perform similarly to other features.

Let's review Q1: §4.1 and §4.2 demonstrate that **features, as research units, effectively address the dual challenges of limited knowledge expression and poor interpretability.**

4.3 Features Exhibit Stronger Monosemanticity

Motivation While our previous analyses demonstrate features' superior performance in both knowledge expression and interpretability, these findings indirectly suggest stronger monosemanticity of features. We now seek direct evidence to verify whether features better align with our desired scenario illustrated in Figure 1(b). This would further support our findings in §4.1 and §4.2[4].

Experiment Settings To evaluate monosemanticity, we expect features corresponding to specific facts to show high activation values for those facts and low activation values for others. However, this evaluation presents two key challenges:

(1) Individual fact analysis is unreliable because each fact activates only a small subset of features. This sparsity means features corresponding to different facts might appear separated by chance, rather than due to true monosemanticity. (2) Analyzing all facts simultaneously would involve too many features, likely producing high activation values for some features regardless of input facts. This noise would mask the underlying feature-fact relationships we aim to study.

We address these challenges through a three-step approach: (1) *Relation Selection*: We select 5 relations (P39, P264, P37, P108, P131; see Appendix A) and designate these facts as *relation-facts*.

(2) *Input Construction*: We maintain a constant total input size (2,591 facts) while varying the proportion of relation-facts from 0% to 100% in 20% increments. For example, the "40%" configuration contains 1,036 randomly sampled relation-facts and 1,555 non-relation facts.

(3) *Activation Analysis*: For each configuration, we: (a) Record activation values from features (or neurons) associated with relation-facts. (b) Visualize distributions using kernel density estimation (KDE), , where the x-axis is activation scores and the y-axis shows feature density. (c) Plot mean activation values for clearer interpretation.

In this setup, stronger monosemanticity is characterized by clear distribution separation across different input configurations and higher activation scores when relation-fact proportions increase. Results are shown in Figure 8.

Findings **We resolve Q2: features exhibit superior monosemanticity compared to neurons.** (1) Features display distinct, well-separated activation waves that correlate with relation-fact proportions. In contrast, neurons show overlapping distributions and activation even without relevant inputs, i.e., 0% condition (Top of Figure 8).

[4]Based on the analysis in §4.1 and §4.2, we focus our comparison specifically on post-MLP features and neurons.

(2) Feature activation values increase systematically with relation-fact proportion, while neuron mean activation values start notably above zero and show minimal variation, especially between 60% and 100% (Bottom of Figure 8).

These findings demonstrate that features exhibit stronger fact-specific correspondence by remaining unresponsive to irrelevant inputs. Statistical analysis confirms this separation (all $p < 0.001$, Cohen's d> 0.8; see Appendix H.2).

5 Feature-based Knowledge Erasure

We evaluate our feature-based method against neuron-based approaches in erasing privacy-related information from LMs. This downstream application addresses Q3 raised in §1, further validates our previous analysis, and demonstrates the practical value of our findings and analysis.

5.1 Dataset

We construct **PrivacyParaRel**, a dataset containing synthetic privacy-sensitive information, following the triple format used by Elazar et al. (2021). Each entry is structured as ⟨*subject, relation, object*⟩, such as ⟨*Alice, Social Security Number, 123-45-6789*⟩, with multiple query variations generated for each fact. This format maintains consistency with factual knowledge datasets, enabling direct method transfer while addressing privacy concerns through synthetic data. We generate 1,500 different facts, each accompanied by six different query variations, resulting in 9,000 total entries. Further details in Appendix I.

5.2 Experiment Settings

To erase specific knowledge from LMs, we first perform incremental fine-tuning on our privacy dataset, allowing the model to learn the private information. For erasure, we extract neurons and post-MLP features, then explore two approaches based on neurons and features respectively. In both approaches, we modify weights in the MLP layers to ensure fair comparison, as neuron-based methods operate on MLP weights. Note that this weight modification differs from the activation-zeroing approach used in $\Delta Prob$ calculation.

Neuron-based approach Following existing neuron-based knowledge editing methods (Dai et al., 2022; Chen et al., 2024b), for each identified neuron n_l^i in layer l, we set the i-th column to zero in $\mathbf{W}_l^{(2)} \in \mathbb{R}^{d_{io} \times d_m}$, where $\mathbf{W}_l^{(2)}$ is the second linear transformation matrix in the l-th MLP layer. Here, d_{io} denotes the input/output dimension, and d_m represents the intermediate dimension.

Feature Steering (Activation Steering) Feature Steering (Templeton et al., 2024; Durmus et al., 2024; Chalnev et al., 2024; Rimsky et al., 2024; Turner et al., 2024) is an approach that manipulates model behavior by adding vectors to the model's activations during inference. While effective for controlling outputs, it makes temporary changes that require intervention at each inference, unlike permanent weight modifications. This makes it unsuitable for privacy protection applications that require persistent knowledge erasure, thus we do not include it as a baseline.

Feature-based approach We propose **FeatureErase**, a reconstruction-based approach for feature-based model editing. Since features are not naturally exist in LMs, directly using them for model editing poses challenges. To address this, **FeatureErase** presents the first feature-based editing method, inspired by the activation reconstruction method used for $\Delta Prob$ (Appendix C.1). For each identified feature f_l^i in the l-th MLP layer, we create a one-hot probe vector ($\mathbf{p}_j^i$):

$$\mathbf{p}_j^i = \begin{cases} 1 & \text{if} \quad j = i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Let $\mathbf{W}_e \in \mathbb{R}^{d_f \times d_m}$ be the encoding matrix learned by SAE. By reconstructing through its transpose (decoder matrix) $\mathbf{W}_e^T$, we obtain the feature's contribution pattern in the original MLP activation space:

$$\mathbf{h}^i = \mathbf{W}_e^T \mathbf{p}^i, \quad \mathbf{h}^i \in \mathbb{R}^{d_m} \tag{6}$$

The reconstructed vector $\mathbf{h}^i$ reveals the feature's distributed influence. We traverse all features f_l^i, and identify significant positions in the weight matrix $\mathbf{W}_l^{(2)}$ by thresholding:

$$P = \{(l, c, i) \,\big|\, |\mathbf{h}_c^i| > \tau_2, \text{ for all } i, l\} \tag{7}$$

where (l, c, i) represents the position in layer l, the c-th column and i-th row of $\mathbf{W}_l^{(2)}$, $\mathbf{h}_c^{(i)}$ is the c-th value of $\mathbf{h}^{(i)}$, and τ_2 is a hyperparameter (Appendix D.5). Finally, we zero out these specific weights:

$$\mathbf{W}_{l,c}^{(2)} = 0, \quad \forall (l, c, i) \in P \tag{8}$$

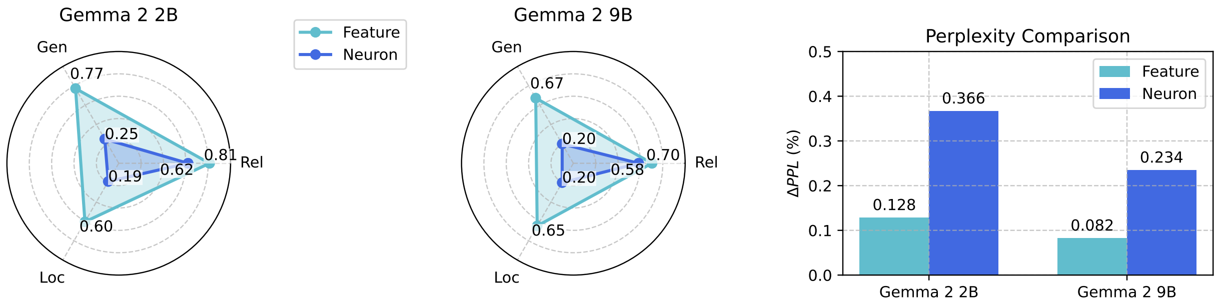

Figure 9: Results of knowledge erasure for privacy protection. For the radar chart metrics (Rel, Gen and Loc), higher values indicate better performance, while lower values in the bar chart indicate better performance.

Notably, this method achieves finer granularity than neuron-based approaches by enabling selective modification of specific weight positions rather than entire column vectors.

A comprehensive comparison between neuron editing, activation steering, and our FeatureErase method is provided in Appendix J.

Evaluation Metrics We employ four metrics to assess knowledge erasure performance (Yao et al., 2024; Chen et al., 2024b): (1) Reliability (Rel): The probability that the model fails to correctly answer privacy-related queries after erasure. (2) Generalization (Gen): The probability that the model fails to answer privacy-related queries with different phrasings. This metric is crucial as high Reliability with low Generalization indicates potential "jailbreak" (Wei et al., 2023) phenomenon where models reveal private data in specific contexts. (3) Locality (Loc): The probability that the model correctly answers unrelated queries. This metric ensures that knowledge erasure maintains other model capabilities. (4) Perplexity: Measures the impact on the model's general text generation ability. We use ΔPPL to quantify the perplexity change before (b) and after (a) erasure: $\Delta\text{PPL} = \frac{\text{PPL}_b - \text{PPL}_a}{\text{PPL}_b}$.

5.3 Findings

We resolve Q3: Feature-based model editing outperforms neuron-based methods in privacy knowledge erasure. As shown in Figure 9, features achieve higher Rel scores (~ 0.8) compared to neurons (~ 0.65), indicating better erasure effectiveness. The substantially higher Gen score for features (~ 0.7 vs. ~ 0.25) demonstrates significant mitigation of the "jailbreak" phenomenon. Moreover, features exhibit fewer side effects, evidenced by higher Loc scores (~ 0.7 vs. ~ 0.2 for neurons), indicating minimal impact on other facts, and lower ΔPPL (~ 0.1 vs. ~ 0.3), suggesting better preservation of model generation capabilities. These findings align with our previous conclusions: features exert stronger influence on knowledge expression (higher Rel and Gen) while exhibiting superior monosemanticity, thus requiring fewer editing locations (better Loc and ΔPPL).

6 Related Works

Knowledge Neurons Theory and its Limitations In studying the factual knowledge mechanisms of LMs, researchers often employ the knowledge neuron (KN) theory. Bolukbasi et al. (2021) provided early critique of neuron interpretability, revealing an "interpretability illusion" where neurons seemingly encode simple concepts but actually represent complex phenomena. Geva et al. (2021b) propose that MLP modules simulate key-value memories to store information, while Dai et al. (2022) introduce the concept of knowledge neurons, suggesting that these neurons can store "knowledge". The success of KN-inspired model editing methods (Meng et al., 2022, 2023) further supports the plausibility of the KN theory. However, the KN theory has its limitations. Niu et al. (2024) argue that it oversimplifies the real situation, while Hase et al. (2023) suggest that the location of knowledge localization may not align with the location of greatest impact on knowledge expression. Chen et al. (2024b) further challenge the fundamental assumptions of KN theory, suggesting facts are distributed across neurons and different queries about the same fact may activate different KNs. Additionally, Bricken et al. (2023) find that the activation of a single neuron can have different meanings in different contexts. Limitations in KN-inspired knowledge editing methods have been widely documented, including editing failures and capability impairment (Li et al., 2024; Yao et al., 2023; Cohen et al., 2024; Hoelscher-Obermaier et al., 2023; Pinter and Elhadad, 2023;

Zhang et al., 2024). New evidence further reveals KN-specific challenges: surface-level deception during editing (Xie et al., 2025), knowledge superposition in lifelong contexts (Hu et al., 2025), and cross-lingual neuron dynamics (Cao et al., 2024), prompting KN-targeted solutions (Xie et al., 2024; Hu et al., 2024). These persistent issues indirectly challenge the KN theoretical foundation.

Decomposing Neurons into Features In the domain of general text processing, numerous studies have explored the properties of features. Elhage et al. (2022) demonstrate in toy neural networks, a layer of dimension N may linearly represent many more than N feature, showing that a large set of sparse features can be represented in a lower-dimensional space. Bricken et al. (2023) advance the theory by decomposing neurons into more fine-grained features, arguing for their superiority as units of analysis. Using sparse autoencoders (SAE), they confirm that features correspond to patterns of neuron activations. Subsequently, researchers have improved the SAE method, proposing SAE variants with enhanced performance (Rajamanoharan et al., 2024; Gao et al., 2024). Additionally, Huben et al. (2024) extend this approach to larger-scale LMs, while Templeton et al. (2024) discover highly abstract features that both respond to and cause abstract behaviors. However, research on factual knowledge mechanisms has not yet adopted the features perspective. This paper focuses on transforming the unit of analysis, aiming to address some problems in the domain of factual knowledge.

7 Conclusion

We investigate the mechanism of factual knowledge in LMs and propose a shift from neuron-based to feature-based analysis. Drawing inspiration from feature extraction methods, we first validate the effectiveness of SAE in extracting features for factual knowledge. Based on this new analysis unit, we make several key findings: Features exhibit greater influence on knowledge expressing than neurons, offer enhanced interpretability, and demonstrate superior monosemanticity. Additionally, our feature-based approach demonstrates better performance in erasing privacy-sensitive knowledge from LMs.

8 Limitations

A significant challenge lies in the increased complexity of feature-based methods compared to neuron-based approaches, as features do not constitute natural units of analysis. While neurons are inherent components of LLM architecture, features require additional training of SAE for extraction. Consequently, although we can derive deeper insights using features, translating these insights into model performance improvements presents considerable challenges. For instance, while feature ablation is straightforward, such operations do not modify model parameters. Mapping external features back to model parameters may require an additional mapping layer to establish correspondence between features and weights. Our preliminary approach is based on reconstruction methods, consistent with activation ablation, but this is suboptimal. Therefore, mapping back to weights likely requires further algorithmic innovation.

Additionally, we observe an intriguing clustering phenomenon of SAE features that merits further investigation. While our current analysis confirms that features undergo progressive decomposition with increasing N while maintaining their cluster structure, more fundamental questions remain unexplored. Specifically, we aim to investigate whether the SAE feature space possesses an inherent stable structure and whether it is truly insensitive to N. Such investigations could provide deeper insights into the nature of SAE feature representations.

Furthermore, our proposed FeatureErase method has several specific limitations. First, it is only applicable when SAEs are trained on MLP layers, and does not work when SAEs are trained on residual streams or attention layers. Second, feature-based methods like FeatureErase are computationally more intensive than neuron-based approaches, requiring additional resources for feature extraction and analysis. Third, our current implementation only supports knowledge erasure operations. Extending this approach to knowledge modification would require additional research into mapping mechanisms between features and specific weights.

9 Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. U24A20335, No. 62176257, No. 62406321). This work is also supported by the Youth Innovation Promotion Association CAS and the China Postdoctoral Science Foundation under Grant Number 2024M753500.

References

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

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. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Vi'egas, and Martin Wattenberg. 2021. An interpretability illusion for bert. *ArXiv*, abs/2104.07143.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Pengfei Cao, Yuheng Chen, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. 2024. One mind, many tongues: A deep dive into language-agnostic knowledge neurons in large language models. *arXiv preprint arXiv:2411.17401*.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. 2024. Improving steering vectors by targeting sparse autoencoder features. *Preprint*, arXiv:2411.02193.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024a. 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.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024b. Knowledge localization: Mission not accomplished? enter query localization! *Preprint*, arXiv:2405.14117.

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

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *ArXiv*, abs/2309.08600.

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, Dublin, Ireland. Association for Computational Linguistics.

Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, Oliver Rausch, Saffron Huang, Sam Bowman, Stuart Ritchie, Tom Henighan, and Deep Ganguli. 2024. Evaluating feature steering: A case study in mitigating social biases.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

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

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*.

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

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

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada. Association for Computational Linguistics.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Wilke: Wise-layer knowledge editor for lifelong knowledge editing. *arXiv preprint arXiv:2402.10987*.

Chenhui Hu, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Knowledge in superposition: Unveiling the failures of lifelong knowledge editing for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24086–24094.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024. Unveiling the pitfalls of knowledge editing for large language models. In *The Twelfth International Conference on Learning Representations*.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, 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, Miami, Florida, US. Association for Computational Linguistics.

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

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

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? In *The Twelfth International Conference on Learning Representations*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Yuval Pinter and Michael Elhadad. 2023. Emptying the ocean with a spoon: Should we edit models? In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024. Improving dictionary learning with gated sparse autoencoders. *arXiv preprint arXiv:2404.16014*.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.

Gemma Team, Morgane Riviere, 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 Rogozińska, 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-Plucińska, 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 Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. 2024. Steering language models with activation engineering. *Preprint*, arXiv:2308.10248.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. Jailbroken: How does llm safety training fail? In *Advances in Neural Information Processing Systems*, volume 36, pages 80079–80110. Curran Associates, Inc.

Jiakuan Xie, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Revealing the deceptiveness of knowledge editing: A mechanistic analysis of superficial editing. *Preprint*, arXiv:2505.12636.

Jiakuan Xie, Pengfei Cao, Yuheng Chen, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Memla: Enhancing multilingual knowledge editing with neuron-masked low-rank adaptation. *arXiv preprint arXiv:2406.11566*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024. Knowledge circuits in pretrained transformers. *arXiv preprint arXiv:2405.17969*.

Zhihao Zhang, Jun Zhao, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Unveiling linguistic regions in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6228–6247, Bangkok, Thailand. Association for Computational Linguistics.

A Experimental Dataset Introduction

In our experiments, we selected the ParaRel dataset Elazar et al. (2021), a high-quality resource of cloze-style query English paraphrases. It contains a total of 328 paraphrases for 38 relations. We further conducted a basic filtering, excluding 2 relations that had no paraphrases. Table 1 displays these relations and corresponding example data.

B Experimental Hardware Specification and Environment

All experiments are conducted using a high-performance computing system with an Intel(R) Xeon(R) CPU E5-2680 v4 (2.40GHz, 56 cores) processor and 10 NVIDIA GeForce RTX 3090 GPUs, each equipped with 24576 MiB of memory. The software environment consists of Python 3.10.10 and PyTorch 2.0.0+cu117 for deep learning implementations.

C Feature Ablation Process and Autointerpretation Protocol

Here we will introduce in detail how we obtain $\Delta Prob$ and IS.

C.1 Feature Ablation Process

Let $\mathbf{h} \in \mathbb{R}^{d_m}$ denote the original component activation (e.g., MLP activation) at a specific layer. Through SAE, we obtain the encoding matrix $\mathbf{W}_e \in \mathbb{R}^{d_f \times d_m}$ and feature vector $\mathbf{f} = \sigma(\mathbf{W}_e \mathbf{h}) \in \mathbb{R}^{d_f}$, where d_f is the number of features and σ is the activation function. The feature ablation process follows these steps:

1. Given a set of target features to ablate S, we create a masked feature vector $\mathbf{f}'$:

$$\mathbf{f}'_i = \begin{cases} 0 & \text{if } i \in S \\ \mathbf{f}_i & \text{otherwise} \end{cases} \tag{9}$$

2. We reconstruct the activation using the decoder matrix $\mathbf{W}_e^T$:

$$\mathbf{h}' = \mathbf{W}_e^T \mathbf{f}' \in \mathbb{R}^{d_m} \tag{10}$$

3. Replace the original activation $\mathbf{h}$ with the reconstructed activation $\mathbf{h}'$ in the model's forward computation to obtain the modified probability $Prob_a$. This process allows us to measure how specific features influence the model's knowledge expression by comparing the original probability $Prob_b$ (using $\mathbf{h}$) with the modified probability $Prob_a$ (using $\mathbf{h}'$) through the $\Delta Prob$ metric.

C.2 Autointerpretation Protocol

We adapt the interpretability evaluation method from Bills et al. (2023) for our factual knowledge dataset, which consists of triples in various domains. This method is applied to features extracted by Sparse Autoencoders (SAE) from LLMs' post-MLP residual flow (this paper uses Gemma 2 2B and Gemma 2 9B). The process for each feature is as follows:

1. We select 20 diverse samples from our dataset of factual knowledge triples. Each sample is run through LLMs, measuring the feature's activation (range 0-1).

2. We identify the top 3 samples with highest feature activation. These high-activation samples are provided to a large language model. (we use gpt-4o-mini here[5].)

3. Based on this interpretation, we ask gpt-4o-mini to predict activation levels for 6 new samples: 3 high-activation and 3 random samples from our dataset.

4. We calculate the correlation between these predictions and the actual Gemma 2 2B activations, yielding an interpretability score for the feature.

D Details of SAE, PCA, ICA and FeatureEdit

This section details the four methods used for extracting interpretable features: Sparse Autoencoders (SAE), Principal Component Analysis

[5]Any large language model can be used, but it is required that this LLMs can output logprobs.

Relation	Example data	
	Example Query	**Answer**
P39	Adrian IV has the position of	pope
P264	Purple Hearts is represented by music label	Sunshine
P37	The official language of Republic of Ingushetia is	Russian
P108	Henry Swanzy works for	BBC
P131	Heaton Park is located in	Manchester
P103	The native language of Francis Ponge is	French
P176	Fiat Grande Punto is produced by	Fiat
P30	Somalia is located in	Africa
P178	Gain Ground is developed by	Sega
P138	International Day for Biological Diversity is named after	biodiversity
P47	Ukraine shares border with	Poland
P17	Media Development Authority is located in	Singapore
P413	Joe Torre plays in [MASK] position.	catcher
P27	Edward Wollstonecraft is [MASK] citizen.	Australia
P463	Chuck Schuldiner is a member of	Death
P364	The original language of NU.nl is	Dutch
P495	The Creepshow was created in	Canada
P449	Yes Minister was originally aired on	BBC
P20	Margaret Cavendish, Duchess of Newcastle-upon-Tyne died in	England
P1376	Rumbek is the capital of	Lakes
P1001	Minister for Foreign Affairs is a legal term in	Australia
P361	propellant is part of	cartridge
P36	The capital of Flanders is	Brussels
P1303	Ludovico Einaudi plays	piano
P530	Brunei maintains diplomatic relations with	Australia
P19	Lopo Soares de Albergaria was born in	Lisbon
P190	Bratislava and [MASK] are twin cities.	Dublin
P740	Shirehorses was founded in	Manchester
P136	Frank Mantooth plays [MASK] music.	jazz
P127	AVCHD is owned by	Sony
P1412	Karl Bodmer used to communicate in	French
P407	Zarez was written in	Croatian
P140	Leo IX is affiliated with the [MASK] religion.	Christianity
P279	quinquina is a subclass of	wine
P276	Al-Rifa'i Mosque is located in	Cairo
P159	The headquarter of Allied Command Transformation is in	Norfolk
P106	Giuseppe Saracco is a [MASK] by profession.	politician
P101	Aleksei N. Leontiev works in the field of	psychology
P937	Joseph Chamberlain used to work in	London

Table 1: Example data of the ParaRel dataset (Elazar et al., 2021).

(PCA), Independent Component Analysis (ICA), and random directions. Assume that the input is MLP activation. Other inputs are similar.

D.1 JumpReLU Sparse Autoencoders (SAEs)

JumpReLU SAEs are neural networks that learn sparse representations through a threshold-based activation mechanism (Lieberum et al., 2024). Given MLP activations $\mathbf{h} \in \mathbb{R}^{d_m}$, the encoder and decoder functions are defined by:

$$\mathbf{f}(\mathbf{h}) := \sigma(W_{enc}\mathbf{h} + \mathbf{b}_{enc}) \tag{11}$$

$$\hat{\mathbf{h}}(\mathbf{f}) := W_{dec}\mathbf{f} + \mathbf{b}_{dec} \tag{12}$$

where $W_{enc} \in \mathbb{R}^{d_f \times d_m}$, $W_{dec} \in \mathbb{R}^{d_m \times d_f}$, $\mathbf{b}_{enc} \in \mathbb{R}^{d_f}$, $\mathbf{b}_{dec} \in \mathbb{R}^{d_m}$.

The JumpReLU activation σ is defined as:

$$\sigma(\mathbf{z}) = \text{JumpReLU}_\theta(\mathbf{z}) := \mathbf{z} \odot H(\mathbf{z} - \theta) \tag{13}$$

where $\theta > 0$ is the learnable threshold parameter and H is the Heaviside step function.

The loss function combines reconstruction error with an L0 sparsity penalty:

$$\mathcal{L} = \|\mathbf{h} - \hat{\mathbf{h}}(\mathbf{f}(\mathbf{h}))\|_2^2 + \lambda\|\mathbf{f}(\mathbf{h})\|_0 \tag{14}$$

where λ controls the sparsity penalty weight.

Features are obtained through:

$$\mathbf{F} = \{\mathbf{f}(\mathbf{h}) := \sigma(W_{enc}\mathbf{h} + \mathbf{b}_{enc})\} \tag{15}$$

Selected features are identified using:

$$\mathbf{F_a} = \{f \in \mathbf{F} \mid a(f) > \tau_1 \cdot \max_{f \in \mathbf{F}} a(f)\} \tag{16}$$

In this equation, $a(f)$ represents the activation value of feature f_i, while τ_1 serves as the threshold parameter controlling feature selection sensitivity. The term $\max_{f \in \mathbf{F}} a(f)$ denotes the maximum activation value across all features.

D.2 Principal Component Analysis (PCA)

PCA finds orthogonal directions that capture maximum variance in the data. For MLP activations $\mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_{d_m}]^T$, the process involves several key steps. First, we center the data by computing $\mathbf{H}_c = \mathbf{H} - \mathbb{E}[\mathbf{H}]$. Next, we compute the covariance matrix $\mathbf{C} = \frac{1}{n}\mathbf{H}_c^T\mathbf{H}_c$. We then perform eigendecomposition $\mathbf{C} = \mathbf{V}\Lambda\mathbf{V}^T$, where $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{d_m}]$ contains eigenvectors. Finally, we project the data using $\mathbf{F} = \mathbf{H}_c\mathbf{V}_{d_f}$, where $\mathbf{V}_{d_f}$ contains top d_f eigenvectors.

Features are obtained through:

$$\mathbf{F} = \{\mathbf{f}(\mathbf{h}) := \mathbf{h}^T\mathbf{V}_{d_f}\} \tag{17}$$

Selected features are identified using Equation 16.

D.3 Independent Component Analysis (ICA)

ICA seeks to find statistically independent components by maximizing non-Gaussianity. The process begins with whitening, where we transform the data to have unit variance in all directions:

$$\mathbf{H}_w = \mathbf{H}_c\mathbf{V}\Lambda^{-1/2} \tag{18}$$

We then find the unmixing matrix $\mathbf{W} \in \mathbb{R}^{d_f \times d_m}$ that maximizes non-Gaussianity:

$$\mathbf{F} = \mathbf{H}_w\mathbf{W} \tag{19}$$

The optimization typically uses approximations of negentropy:

$$J(\mathbf{w}) = [E\{G(\mathbf{w}^T\mathbf{h}_w)\} - E\{G(\nu)\}]^2 \tag{20}$$

where G is a non-quadratic function and ν is a standard Gaussian variable.

Features are obtained through:

$$\mathbf{F} = \{\mathbf{f}(\mathbf{h}) := \mathbf{h}_w^T\mathbf{W}\} \tag{21}$$

Selected features are identified using Equation 16.

D.4 Random Directions (RD)

Random directions serve as a baseline method through a three-step process. Initially, we generate a random matrix $\mathbf{R} \in \mathbb{R}^{d_m \times d_f}$ with entries drawn from $\mathcal{N}(0, 1/\sqrt{d_m})$. We then apply QR decomposition to obtain an orthonormal basis:

$$\mathbf{R} = \mathbf{Q}\mathbf{R}_{upper} \tag{22}$$

where $\mathbf{Q} \in \mathbb{R}^{d_m \times d_f}$ is an orthonormal matrix and $\mathbf{R}_{upper} \in \mathbb{R}^{d_f \times d_f}$ is an upper triangular matrix. Finally, we project the data using the orthonormal matrix: $\mathbf{F} = \mathbf{H}\mathbf{Q}$.

Features are obtained through:

$$\mathbf{F} = \{\mathbf{f}(\mathbf{h}) := \mathbf{h}^T\mathbf{Q}\} \tag{23}$$

Selected features are identified using Equation 16.

D.5 Method-specific Parameters

The implementation of each method involves specific parameter settings. For SAE, we use $\beta = 3$, $\rho = 0.05$, sigmoid activation, and feature dimension $d_f = n \times d_m$, with $n = 4$ in this paper. The training process employs the Adam optimizer with learning rate $1e^{-3}$, batch size 256, and runs for 100 epochs. Early stopping is triggered if validation

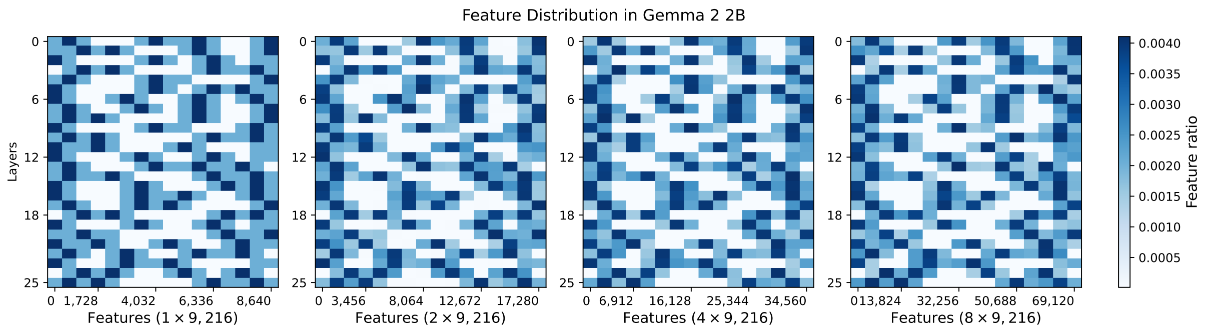

Figure 10: Feature cluster Results for Gemma 2 2B.

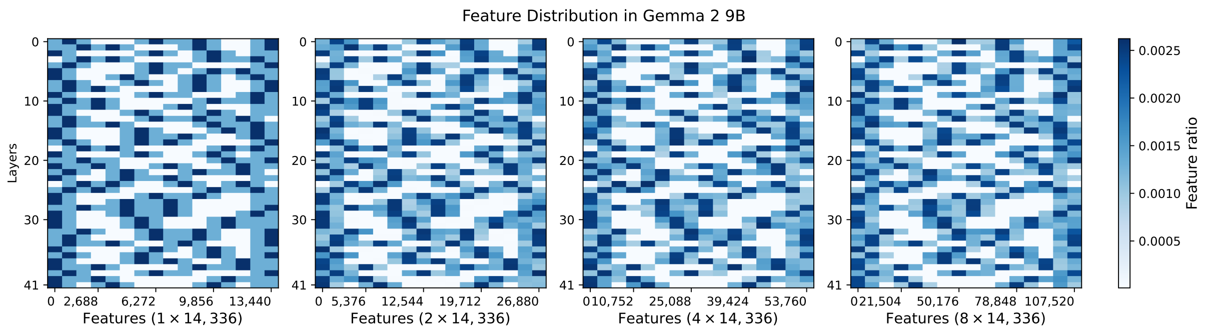

Figure 11: Feature cluster Results for Gemma 2 9B.

loss does not improve for 10 consecutive epochs. Input activations are standardized to zero mean and unit variance before training.

PCA employs an explained variance ratio threshold of 0.95, which determines the resulting d_f features. The input data is centered but not scaled, as variance information is crucial for principal component identification. ICA utilizes the FastICA algorithm with cubic G function and feature dimension $d_f = 4d_m$, with input data whitened during preprocessing. The Random method uses Gaussian initialization with variance scaling and maintains a feature dimension of $d_f = 4d_m$.

For feature selection across all methods, we employ a threshold $\tau_1 = 0.3$ to identify significant features, striking a balance between feature coverage and selectivity. This threshold was determined through preliminary experiments examining the distribution of feature activations across different knowledge categories. Specifically, $\tau_1 = 0.3$ ensures capture of features that demonstrate substantial activation (at least 30% of maximum activation) while filtering out noise and weakly activated features.

For FeatureEdit, we set the reconstruction threshold $\tau_2 = 0.1$ to identify significant weight positions. This threshold was chosen based on the empirical observation of weight contribution distributions in the reconstructed activation space, ensuring that we capture meaningful feature influences while maintaining editing precision. The relatively small threshold value allows us to identify subtle but important feature contributions in the distributed representations.

E Paired T-test Results for Preliminary Experiment

To rigorously validate the superiority of SAE features over baseline methods, we conduct paired t-tests using full dataset for each method. For both models (Gemma 2 2B and Gemma 2 9B) and both metrics ($\Delta Prob$ and IS), we compare SAE with each baseline method (PCA, ICA, and random baseline). The statistical significance of the differences is assessed using paired t-tests, as we compare different methods on the same set of instances. Table 2 presents the detailed statistical analysis results.

Notably, we include Cohen's d effect size alongside traditional significance testing because p-values alone may not reflect the practical significance of the differences, especially with large sample sizes (all p<0.001). Cohen's d measures the standardized difference between two means, where values above 0.8 indicate large effects. Our results show substantial effect sizes (Cohen's d ranging from 0.38 to 3.81, with most values exceeding 0.8), confirming not only the statistical significance but also the practical importance of SAE's improvements over baseline methods. Particularly strong effects are observed when comparing SAE with the random baseline (Cohen's d > 1.6), and in the IS metric where almost all comparisons show large effect sizes (Cohen's d > 0.7).

F Quantitative Analysis of Feature Stability Across Different N Values

To further validate our observation that the impact of the number of features (N) is less significant than anticipated, we conduct a comprehensive quantitative analysis on the entire dataset. This analysis aims to support our conclusion that our findings are stable across different values of the hyperparameter N.

F.1 Methodology

We define N as $N = n \times \text{len(MLP activation)}$, where n is a positive integer. We use $n = 1$ as the baseline for comparison. This approach yields layer $\times$ N features for each model.

Using Gemma 2 2B as an example, our methodology is as follows:

1. For a given fact, when $n = 1$, we record the positions of activated features as [layer, position].

2. For any integer $n > 1$, based on the $n = 1$ case, we expect features to fall within the range $[\text{layer}, \text{position} \times n, (\text{position} + 1) \times n - 1]$.

3. We then compare the actual positions of features for $n > 1$ with these expected positions and calculate the overlap ratio.

4. We repeat this process for the entire dataset and compute the average overlap ratio.

We apply this methodology to both Gemma 2 2B and Gemma 2 9B models, using n values of 1, 2, 4, and 8.

F.2 Results

Table 3 presents the average overlap ratios for different n values across both models. Additionally, Figure 11 complements the results shown in Figure 3 from the main text. While Figure 3 only presents the results for Gemma 2 2B, Figure 11 displays the results for both Gemma 2 2B and Gemma 2 9B.

The results in Table 3 demonstrate a high degree of overlap between the expected and actual feature positions across different n values. For both Gemma 2 2B and Gemma 2 9B, we observe that even as n increases to 8, the overlap ratio remains above 0.87, indicating a strong consistency in feature localization.

This quantitative analysis supports our earlier observation that as N increases, the original features are further decomposed but remain aggregated in consistent regions. The high overlap ratios suggest that our conclusions about feature behavior and importance are indeed stable and relatively insensitive to changes in the hyperparameter N.

These findings have important implications for future research in this area, as they suggest that the choice of N, within a reasonable range, does not significantly alter the fundamental patterns of feature activation and localization in relation to factual knowledge representation in language models.

G Knowledge Localization Method

We compare the precision of knowledge neuron localization across different research papers and select Architecture-adapted Multilingual Integrated Gradients (Chen et al., 2024a) as our baseline method, as it demonstrates superior performance in knowledge neuron localization.

Given a query q, they define the probability of the correct answer predicted by a PLMs as follows:

$$\mathrm{F}(\hat{w}_j^{(l)}) = p(y^*|q, w_j^{(l)} = \hat{w}_j^{(l)}) \tag{24}$$

Here, y^* represents the correct answer, $w_j^{(l)}$ denotes the j-th neuron in the l-th layer, and $\hat{w}_j^{(l)}$ is the specific value assigned to $w_j^{(l)}$. To calculate the attribution score for each neuron, they employ the technique of integrated gradients. To compute the attribution score of a neuron $w_j^{(l)}$, they consider the following formulation:

$$\Delta w_j^{(l)} = \overline{w}_j^{(l)} - w'^{(l)}_j \tag{25}$$

$$\mathrm{Attr}(w_j^{(l)}) = \Delta w_j^{(l)} \int_0^1 \frac{\partial\,\mathrm{F}(w'^{(l)}_j + \alpha \Delta w_j^{(l)})}{\partial w_j^{(l)}}\, d\alpha \tag{26}$$

Here, $\overline{w}_j^{(l)}$ represents the actual value of $w_j^{(l)}$, $w'^{(l)}_j$ serves as the baseline vector for $w_j^{(l)}$. The term $\frac{\partial\,\mathrm{F}(w'^{(l)}_j + \alpha \Delta w_j^{(l)})}{\partial w_j^{(l)}}$ computes the gradient with respect to $w_j^{(l)}$. Next, they aim to obtain $w'^{(l)}_j$. Starting from the sentence q, they acquire a baseline sentence and then encode this sentence as a

$\Delta Prob$				
Model	**Method**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B	SAE vs. PCA	85.15	< 0.001	0.70
Gemma 2 2B	SAE vs. ICA	46.33	< 0.001	0.38
Gemma 2 2B	SAE vs. Random	359.48	< 0.001	2.94
Gemma 2 9B	SAE vs. PCA	120.76	< 0.001	0.99
Gemma 2 9B	SAE vs. ICA	77.18	< 0.001	0.63
Gemma 2 9B	SAE vs. Random	466.02	< 0.001	3.81
IS				
Model	**Method**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B	SAE vs. PCA	157.64	< 0.001	1.29
Gemma 2 2B	SAE vs. ICA	87.19	< 0.001	0.71
Gemma 2 2B	SAE vs. Random	200.49	< 0.001	1.64
Gemma 2 9B	SAE vs. PCA	155.98	< 0.001	1.27
Gemma 2 9B	SAE vs. ICA	104.44	< 0.001	0.85
Gemma 2 9B	SAE vs. Random	255.96	< 0.001	2.09

Table 2: Statistical analysis of feature acquisition methods. We report t-statistics, p-values from paired t-tests, and Cohen's d effect sizes for comparing SAE with baseline methods (PCA, ICA, and random baseline) across both metrics ($\Delta Prob$ and IS).

Model	n			
	1	2	4	8
Gemma 2 2B	1.000	0.927	0.891	0.893
Gemma 2 9B	1.000	0.935	0.908	0.879

Table 3: Average overlap ratios for different n values.

vector. Let the baseline sentence corresponding to q_i be q'_i, and q'_i consists of m words, maintaining a length consistent with q, denoted as $q'_i = (q'_{i1} \cdots q'_{ik} \cdots q'_{im})$. Since they are using auto-regressive models, according to Chen et al. (2024a), $q'_{ik} = \langle\text{eos}\rangle$, where $\langle\text{eos}\rangle$ represents "end of sequence" in auto-regressive models. The attribution score $Attr_i(w_j^{(l)})$ for each neuron, given the input q_i, can be determined using Equation (26). For the computation of the integral, the Riemann approximation method is employed:

$$Attr_i(w_j^l) \approx \frac{\overline{w}_j^{(l)}}{N} \sum_{k=1}^{N} \frac{\partial F(w'^{(l)}_j + \frac{k}{N} \times \Delta w_j^{(l)}}{\partial w_j^{(l)}} \tag{27}$$

where N is the number of approximation steps. Then, the attribution scores for each word q_i are aggregated and subsequently normalized:

$$Attr(w_j^l) = \frac{\sum_{i=1}^{m} Attr_i(w_j^l)}{\sum_{j=1}^{n} \sum_{i=1}^{m} Attr_i(w_j^l)} \tag{28}$$

Let $\mathcal{N}$ be the set of neurons classified as knowledge neurons based on their attribution scores exceeding a predetermined threshold τ, for a given input q. This can be formally defined as:

$$\mathcal{N} = \left\{ w_j^{(l)} \,\middle|\, Attr(w_j^{(l)}) > \tau \right\} \tag{29}$$

where l encompassing all layers and j including all neurons within each layer.

H Paired T-test Results for Main Experiment: Features vs. Neurons

H.1 For $\Delta Prob$ and IS

To rigorously validate the comparisons between Post-MLP features and other approaches (Post-Attention features, MLP features, and neurons),

$\Delta Prob$				
Model	**Method**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B	Post-MLP F vs. Post-Att F	152.29	< 0.001	1.24
Gemma 2 2B	Post-MLP F vs. MLP F	56.95	< 0.001	0.46
Gemma 2 2B	Post-MLP F vs. Neurons	138.18	< 0.001	1.13
Gemma 2 9B	Post-MLP F vs. Post-Att F	95.91	< 0.001	0.78
Gemma 2 9B	Post-MLP F vs. MLP F	35.27	< 0.001	0.29
Gemma 2 9B	Post-MLP F vs. Neurons	146.42	< 0.001	1.20

IS				
Model	**Method**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B	Post-MLP F vs. Post-Att F	28.41	< 0.001	0.23
Gemma 2 2B	Post-MLP F vs. MLP F	12.20	< 0.001	0.10
Gemma 2 2B	Post-MLP F vs. Neurons	158.37	< 0.001	1.29
Gemma 2 9B	Post-MLP F vs. Post-Att F	12.55	< 0.001	0.10
Gemma 2 9B	Post-MLP F vs. MLP F	-12.17	< 0.001	-0.10
Gemma 2 9B	Post-MLP F vs. Neurons	162.13	< 0.001	1.32

Table 4: Statistical significance test results comparing Post-MLP features with other features or neurons. For each comparison, we report the t-statistic from paired t-tests, corresponding p-value, and Cohen's d effect size.

we conduct paired t-tests using the full dataset. For both metrics ($\Delta Prob$ and IS), we compare Post-MLP features with each alternative method across both models (Gemma 2 2B and Gemma 2 9B). We assess the statistical significance using paired t-tests, as we compare different methods on the same instances.

The results in Table 4 show varied effect sizes across different comparisons. For $\Delta Prob$, Post-MLP features demonstrate strong advantages over Post-Attention features (Cohen's d: 0.78-1.24) and neurons (Cohen's d > 1.1), while showing more modest advantages over MLP features (Cohen's d: 0.29-0.46). For interpretability (IS), we observe particularly strong effects when comparing Post-MLP features with neurons (Cohen's d > 1.2), while comparisons with other feature types show smaller effects ($|$Cohen's d$| \leq 0.23$). All differences are statistically significant (p < 0.001), though the practical significance varies as indicated by the effect sizes.

H.2 For Monosemanticity

To rigorously validate the separation phenomenon in activation distributions, we conduct paired t-tests on two types of comparisons: adjacent ratios (e.g., 0% vs 20%) and comparisons with the full relation-facts condition (100%).

The results in Table 5 demonstrate strong and consistent separation patterns, particularly in feature-based representations. For adjacent ratio comparisons, features show large effect sizes (Cohen's d ranging from 0.62 to 5.29) between consecutive ratios, with particularly strong separation in the middle ranges (20% to 80%). In contrast, neurons exhibit decreasing effect sizes as the ratio increases, with some comparisons showing small effects (Cohen's d < 0.8) in higher ratios.

When compared against the 100% baseline, features maintain substantial separation across all ratios (Cohen's d ranging from 1.61 to 13.03), indicating clear distinctions in activation patterns even at high ratios. Neurons, while showing strong separation at lower ratios (Cohen's d > 5.0 for 0% vs 100%), demonstrate notably smaller effects at higher ratios (Cohen's d < 1.0 for 80% vs 100%). These patterns quantitatively support the superior monosemanticity of features, as they maintain clearer separation between different proportions of relation facts.

I Synthetic Privacy Dataset Construction and Characteristics

Our synthetic privacy dataset comprises 1,500 structured entries of privacy-sensitive information, distributed equally across three relation types:

phone numbers (P001), home addresses (P002), and email addresses (P003). Each entry contains a universally unique identifier (UUID), a natural language prompt, the corresponding value, and a relation code. The dataset is specifically designed for privacy-focused machine learning research while ensuring zero risk to individual privacy through complete synthetic generation.

I.1 Dataset Components

The dataset construction employs carefully curated component lists to ensure both consistency and variability. We organize our foundational elements into three main categories: identity components, location information, and contact details.

I.2 Generation Process

The dataset generation follows a systematic process to ensure consistency and quality. Names are created by combining first and last names from predefined lists, ensuring unique combinations. Values are generated according to type-specific rules: phone numbers follow the "555-XXX-XXXX" format with random digits, addresses combine random street numbers (1-9999) with component elements, and email addresses merge usernames with random numbers (1-999) and domains.

For each privacy fact, we create six variations of natural language queries, covering both declarative statements and questions. This approach expands our 1,500 unique facts into 9,000 total query-answer pairs. The dataset maintains equal distribution across relation types (500 facts each) and ensures no duplicate entries within each type.

Our quality control process focuses on three key aspects. First, we maintain consistent formatting across all entries to ensure data uniformity. Second, we establish strong referential integrity between names and their associated information to maintain data coherence. Third, we ensure reproducibility through systematic component combination, allowing for dataset regeneration when needed.

I.3 Research Applications

This dataset supports various research objectives in privacy-preserving machine learning. It enables thorough model evaluation for information retention and leakage, facilitating the development and evaluation of privacy-protecting mechanisms. The dataset also supports analysis of natural language understanding in the context of structured personal information, while enabling assessment of format learning and consistency in generated content. The synthetic nature of the dataset eliminates privacy concerns while maintaining realistic data patterns and relationships, making it ideal for academic research in privacy-preserving technologies.

J Comparison of Knowledge Editing Methods

Table 9 provides a detailed comparison between three approaches for knowledge manipulation in language models: Neuron Editing, Activation Steering, and our proposed FeatureEdit method. As shown in Table 9, while both Neuron Editing and our FeatureEdit approach permanently modify model weights, FeatureEdit achieves this with significantly higher precision by targeting specific weight positions identified through feature reconstruction. In contrast, Activation Steering offers fine control but requires intervention at each inference, making it unsuitable for applications requiring permanent knowledge removal such as privacy protection.

Adjacent Ratio Comparisons				
Model	**Comparison**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B Feature	20 vs. 0	27.99	< 0.001	1.25
Gemma 2 2B Feature	40 vs. 20	118.21	< 0.001	5.29
Gemma 2 2B Feature	60 vs. 40	57.74	< 0.001	2.58
Gemma 2 2B Feature	80 vs. 60	67.87	< 0.001	3.04
Gemma 2 2B Feature	100 vs. 80	35.94	< 0.001	1.61
Gemma 2 2B Neuron	20 vs. 0	50.81	< 0.001	2.27
Gemma 2 2B Neuron	40 vs. 20	70.10	< 0.001	3.14
Gemma 2 2B Neuron	60 vs. 40	72.44	< 0.001	3.24
Gemma 2 2B Neuron	80 vs. 60	-2.82	0.005	-0.13
Gemma 2 2B Neuron	100 vs. 80	12.58	< 0.001	0.56
Gemma 2 9B Feature	20 vs. 0	13.83	< 0.001	0.62
Gemma 2 9B Feature	40 vs. 20	106.59	< 0.001	4.77
Gemma 2 9B Feature	60 vs. 40	67.33	< 0.001	3.01
Gemma 2 9B Feature	80 vs. 60	29.32	< 0.001	1.31
Gemma 2 9B Feature	100 vs. 80	88.83	< 0.001	3.97
Gemma 2 9B Neuron	20 vs. 0	56.45	< 0.001	2.53
Gemma 2 9B Neuron	40 vs. 20	29.68	< 0.001	1.33
Gemma 2 9B Neuron	60 vs. 40	19.10	< 0.001	0.85
Gemma 2 9B Neuron	80 vs. 60	9.25	< 0.001	0.41
Gemma 2 9B Neuron	100 vs. 80	17.24	< 0.001	0.77

Comparisons with 100% Baseline				
Model	**Comparison**	**t-statistic**	**p-value**	**Cohen's d**
Gemma 2 2B Feature	100 vs. 0	255.40	< 0.001	11.43
Gemma 2 2B Feature	100 vs. 20	220.35	< 0.001	9.86
Gemma 2 2B Feature	100 vs. 40	148.82	< 0.001	6.66
Gemma 2 2B Feature	100 vs. 60	104.99	< 0.001	4.70
Gemma 2 2B Feature	100 vs. 80	35.94	< 0.001	1.61
Gemma 2 2B Neuron	100 vs. 0	153.64	< 0.001	6.87
Gemma 2 2B Neuron	100 vs. 20	112.58	< 0.001	5.04
Gemma 2 2B Neuron	100 vs. 40	66.82	< 0.001	2.99
Gemma 2 2B Neuron	100 vs. 60	11.51	< 0.001	0.51
Gemma 2 2B Neuron	100 vs. 80	12.58	< 0.001	0.56
Gemma 2 9B Feature	100 vs. 0	272.33	< 0.001	12.18
Gemma 2 9B Feature	100 vs. 20	291.28	< 0.001	13.03
Gemma 2 9B Feature	100 vs. 40	162.05	< 0.001	7.25
Gemma 2 9B Feature	100 vs. 60	120.44	< 0.001	5.39
Gemma 2 9B Feature	100 vs. 80	88.83	< 0.001	3.97
Gemma 2 9B Neuron	100 vs. 0	111.78	< 0.001	5.00
Gemma 2 9B Neuron	100 vs. 20	74.64	< 0.001	3.34
Gemma 2 9B Neuron	100 vs. 40	38.81	< 0.001	1.74
Gemma 2 9B Neuron	100 vs. 60	25.39	< 0.001	1.14
Gemma 2 9B Neuron	100 vs. 80	17.24	< 0.001	0.77

Table 5: Statistical analysis of activation distribution separation. We report t-statistics, p-values, and Cohen's d effect sizes for both adjacent ratio comparisons and comparisons with the 100% condition. Adjacent ratio comparisons show the separation between consecutive ratios, while baseline comparisons demonstrate the differences from the full relation-facts condition.

Category	Component Type	Count	Examples
Identity	First Names	30	Alex, Bailey, Casey, Dana, Ellis
	Last Names	30	Smith, Johnson, Williams, Brown, Jones
Location	Street Names	30	Maple, Oak, Pine, Cedar, Elm
	Cities	30	Springfield, Rivertown, Lakeside, Hillview
	State Codes	20	AA, BB, CC, DD, EE
Contact	Email Domains	10	example.com, sample.net, test.org

Table 6: Dataset generation components.

Code	Type	Format Template	Example Query Templates
P001	Phone	555-XXX-XXXX	"[name]'s phone number is" "What is [name]'s phone number?" "How can I reach [name] by phone?"
P002	Address	[Number] [Street] St, [City], [State] [ZIP]	"[name]'s home address is" "Where does [name] live?" "What is [name]'s residential address?"
P003	Email	[firstname].[lastname] [number]@[domain]	"[name]'s email address is" "What's [name]'s email?" "How can I contact [name] by email?"

Table 7: Relation types and query templates.

Field	Type	Description	Example
uuid	string	Unique identifier	"550e8400-e29b-41d4-a716-446655440000"
sentence	string	Natural language prompt	"Casey Thompson's phone number is"
answer	string	Corresponding value	"555-234-5678"
relation	string	Relation type code	"P001"

Table 8: Dataset entry structure.

Aspect	Neuron Editing	Activation Steering	FeatureEdit (Ours)
Target of modification	Model weights (entire columns)	Activations during inference	Model weights (selective positions)
Persistence	Permanent (persists across all future inferences)	Temporary (requires intervention each inference)	Permanent (persists across all future inferences)
Implementation method	Weight zeroing of entire columns	Adding vectors to the residual stream	Selective weight zeroing based on feature reconstruction
Granularity	Modifies entire weight columns	Modifies entire activation patterns	Modifies specific weight positions
Feature utilization	Uses neurons to guide weight manipulation	Uses features to guide activation manipulation	Maps features to weight positions via reconstruction
Application focus	Knowledge erasure	Controlling model behavior during inference	Permanent knowledge erasure for privacy protection

Table 9: Comparison of Neuron Editing, Activation Steering, and FeatureEdit