Towards Explaining Classification Models in Security with Sparse Autoencoders

Nils Ole Breuer
TU Wien

Lorenz Linhardt
BIFOLD & TU Berlin

Philipp Normann
TU Wien

Daniel Arp
TU Wien

***Abstract*—Sparse Autoencoders (SAEs) offer a promising unsupervised interpretability approach for extracting human-interpretable concepts from large language models. Yet, their use in the security domain remains underexplored. Security-related classification tasks typically rely on smaller models than those commonly studied with SAEs. In this paper, we examine how SAEs can be used to interpret classification models fine-tuned for security tasks. We apply an interpretability framework that combines established techniques for foundation models to generate concept explanations, focusing on two widely studied problems in safety and security: hate speech and deepfake detection. We demonstrate its ability to produce meaningful concept explanations while identifying critical challenges for the effective deployment of SAEs in security contexts. Our findings suggest that while SAEs offer a promising unsupervised technique for generating concept explanations, addressing the identified challenges is necessary for their useful application in security interpretability.**

***Index Terms*—XAI for Security, Sparse Autoencoders, Explainable AI**

1. Introduction

Machine learning (ML) has become a key component of modern security research, with widespread adoption across domains such as malware detection [1], [2], deepfake detection [3], and vulnerability discovery [4], [5]. In these settings, ML models are leveraged to identify malicious or deceptive behaviors, uncover previously unknown attack patterns, and improve the efficiency of security analysis at scale. These models are often deployed in black-box settings, limiting visibility into their decision-making processes, complicating their deployment in security contexts, as practitioners may not trust model outputs without interpretable reasoning [6]. This challenge has been amplified by the recent shift toward larger and more complex transformer-based architectures [7]–[9].

To elucidate the decision strategies of such complex nonlinear models, explainable AI methods have been employed for over a decade [10], [11]. While early works have produced widely used attribution methods, mapping classification outputs to input dimensions [10], [12], [13], attention has since shifted to concept-based explanations, uncovering higher-level concepts used by machine learning models [14]–[16]. In security, such explanation methods have found application in various settings [17]–[21].

In recent years, Sparse Autoencoders (SAEs) [22] have emerged as a promising unsupervised and concept-based interpretability approach. This technique has garnered attention for its ability to extract human-interpretable concepts from large language models [23], [24]. SAEs, rooted in dictionary learning [25], cast internal representations of neural network-based machine learning models into a large, sparse representation space, where each dimension potentially corresponds to a concept represented within the model. This requires that they are used as part of a larger interpretability pipeline, where promising dimensions are identified and matched to a human-understandable concept [26]. Considerations of the exact composition of this pipeline are task-dependent, e.g., mapping all concepts used by a large foundation model [22], [24], or investigating specific capabilities [27]. While other explainability methods have successfully been applied in security use-cases, SAEs have so far remained underexplored. This may be because many security-related applications, such as deepfake-, hate speech-, malware-, or intrusion detection, are fundamentally classification tasks, requiring much narrower models than the large foundation models SAEs are typically deployed for.

In this paper, we systematically examine the capabilities of SAEs for interpreting classification models fine-tuned on security-related downstream tasks. In particular, we show how techniques commonly used to apply SAEs to understand foundation models can be combined to explain downstream security models. Using this approach, we study models trained for two security-related tasks, namely hate speech detection and deepfake detection. In particular, we make the following contributions:

1) We provide a framework for generating concept explanations with SAEs for classification models, illustrating how several components from natural language interpretability can be combined to explain machine learning models in security[1].
2) We empirically demonstrate the ability of SAEs to identify interpretable concepts represented within classification models in the domains of deepfake and hate speech detection.
3) We identify three challenges for the effective use of SAEs for classifiers in security settings: concept redundancy, classification-irrelevant concepts, and inconsistent concept annotations.

2. SAE-Based Analysis Framework

The recent popularity of SAEs stems from their successful application to language and other foundation models [23], [24], [28], [29]. Yet in many security applications, the key challenge is interpreting how classification models that are optimized for specific tasks arrive at their decisions. So far, the use of SAEs to interpret such task-specific classifiers has been underexplored.

We address this gap by introducing a framework that combines three established interpretability approaches [30]–[32]: each component was originally developed within the context of foundation model interpretability. By combining these components, it is possible to generate

1. Code is available at: https://github.com/sailab-vienna/sae-for-cybersec

Accepted author manuscript (postprint). Final version published in 2026 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 278–289, doi: 10.1109/EuroSPW72509.2026.00041.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

concept explanations for task-specific classification models in the security setting.

2.1. SAE Training

Our goal is to identify concepts important for a classification task at hand. To achieve this, we first disentangle the feature space of a fine-tuned classification model into candidate concepts by training an SAE on its internal representations. For this purpose, we leverage established libraries [28], [33], which offer flexibility in model architecture and training data.

Accordingly, we extract the `[CLS]`-token activations from the same training data used to fine-tune the classification model, taken from the layer immediately preceding the classification layer, and use these activations to train BatchTopK SAEs [30], an established training method that yields stable, well-distributed sparse features:

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

$$\tilde{\mathbf{x}}(f(\mathbf{x})) := f(\mathbf{x})\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}} \tag{2}$$

Here, $f(\mathbf{x}) \in \mathbb{R}^m$ denotes a sparse latent code, where each entry potentially corresponds to an interpretable concept, and $\tilde{\mathbf{x}}(f(\mathbf{x})) \in \mathbb{R}^n$ denotes the reconstructed input from this representation. $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{n\times m}$ denotes the encoder weight matrix of the SAE, and $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{m\times n}$ the decoder weight matrix. σ denotes the batch top-k function, setting all but the top k activations per batch to zero. This sparsification is replaced by a learned activation threshold during inference, alleviating the need to adhere to the training batch size. Hence, inference-time sparsity may deviate. The SAE is trained by minimizing

$$\mathcal{L}(\mathbf{x}) := \underbrace{\|\mathbf{x} - \tilde{\mathbf{x}}(f(\mathbf{x}))\|_2^2}_{\mathcal{L}_{\text{reconstruction}}} + \alpha\,\mathcal{L}_{\text{aux}}, \tag{3}$$

where $\mathcal{L}_{\text{aux}}$ is an auxiliary loss to reactivate *dead latents*, i.e., latents that are not active for multiple batches and thus lead to wasteful training by occupying model capacity without contributing to the learned representations (see Bussmann et al. [30]).

We select the dictionary size m, i.e., the number of SAE latent dimensions, and the sparsity level k via grid search. Appendix A provides more details on the SAE architecture and the training process. We evaluate SAEs using metrics from an established benchmarking suite [34]:

- L_0 *norm:* The mean number of features activated per input instance, serving as a direct measure of latent sparsity. Lower values indicate more compact and interpretable representations.
- *Fraction of Variance Explained (FVE):* Measures reconstruction quality by capturing how much input variability is preserved. Higher values indicate better information preservation.
- *Fraction of Active Features (Frac-Alive):*. The percentage of features that maintain non-zero activations across the entire dataset. This reflects how effectively a model utilizes its available feature capacity.

To ensure that all relevant classification concepts are captured in the dictionary, we select the SAE achieving the highest FVE on the validation set and use it for the subsequent explanatory steps of the framework.

2.2. SAE Feature Selection

By design, the trained SAEs have a large latent dimension. Yet, for concept-based explanations to be useful, they should consist of a small number of unique concepts [35]. Our framework's second component selects a small subset of features from the trained SAEs that have a potential high relevance for the downstream task. This selection follows the evaluation approach of Kantamneni et al. [31] to ensure the selected SAE features are both meaningful for interpretation and functional for classification.

2.2.1. Class-Wise Difference. For a binary classification task, the most important SAE features are those exhibiting the largest differential activation between the two classes. Specifically, we rank features by the magnitude of their mean activation differences across the training set. The training set is first partitioned into two subsets, $\mathbf{X}_0$ and $\mathbf{X}_1$, based on class labels. These inputs are then processed by the original model and the SAE to extract the latent activation matrices $\mathbf{Z}_0$ and $\mathbf{Z}_1$, where $\mathbf{Z}_c$ represents the set of SAE feature activations corresponding to class $c \in \{0, 1\}$. The activation matrix $\mathbf{Z}_c$ is of dimension $N_c \times m$, where N_c denotes the number of samples in class c and m represents the latent dimensionality of the SAE. We then calculate the class-conditional mean vector $\bar{\mathbf{z}}_c$ for all classes c:

$$\bar{\mathbf{z}}_c = \frac{1}{N_c}\sum_{i=1}^{N_c} \mathbf{Z}_{c,i} \tag{4}$$

Finally, the discriminative feature importance is given as the ranked difference vector $\Delta\bar{\mathbf{z}}$:

$$\Delta\bar{\mathbf{z}} = |\bar{\mathbf{z}}_1 - \bar{\mathbf{z}}_0|. \tag{5}$$

Ultimately, a subset of SAE features is selected based on the ranking of $\Delta\bar{\mathbf{z}}$. To ensure relevance to the downstream task, Kantamneni et al. [31] propose an iterative procedure in which a linear probe is trained using the activations of the top i ranked SAE features from $\Delta\bar{\mathbf{z}}$. The value of i that yields the highest classification performance determines the subset of SAE features considered most important for the downstream task.

2.3. Explanation Generation

In the final component of our framework, we generate concept explanations by assigning human-interpretable concept annotations to the top i SAE features. A typical but labour-intensive approach is to manually assign concept labels based on input samples, where the feature is most highly activated. To generate explanations at scale, we adopt the LLM-based automated interpretability framework [26], [32]: For each of the top i SAE features, we identify the top-activating samples within the training dataset, which serve as a context for the subsequent automated interpretability process. An LLM is then prompted to infer the concept common to these highly activating examples, generating a descriptive hypothesis for the concept encoded in the SAE feature.

3. SAEs for Security Tasks

Using the framework discussed in Section 2, we examine the potential of SAEs to generate post-hoc concept explanations for fine-tuned transformer-based models in

security. To this end, we present case studies in which we examine state-of-the-art models in two widely studied domains in security and safety: deepfake [36] and hate speech detection [37].

Deepfake Detection. Recent generative models can create images that are difficult to distinguish from authentic ones [38]. This development poses significant risks for online fraud and potential harm to individuals, as illustrated by recent incidents involving the chatbot `Grok` [39]. Deepfake detection has long been an active area of research and has evolved in parallel with advances in generative models. Whereas earlier approaches focused on low-level features based on frequency and spectral analysis [40], current approaches take into account high-level concepts [41].

Hate Speech Detection. Insults and hatred towards various marginalized groups have been and continue to be a widespread problem on the internet and constitute a significant source of online harm [42]. In response, transformer-based language models have been widely explored as a means of addressing the challenge to detect online hate speech [43]–[45].

3.1. Models and Datasets

Deepfake Detection. For deepfake detection, we employ two recent approaches. Firstly, we use the model of Park et al. [46], combining a ViT-S backbone [47] with a linear classification head. It is trained on 2.7M fake images from 4,803 generative models and 2.7M real images from existing datasets, achieving a mean average precision of 0.994. We refer to this model as $\mathcal{M}_{\text{comm}}$.

Secondly, we make use of the approach by Cozzolino et al. [8], which consists of a linear SVM on CLIP-ViT activations [48]. The classifier is trained on 10k fake images generated by a latent diffusion model [49] and 10k real images from the MS-COCO dataset [50], achieving an accuracy of 88.5%. We refer to this model as $\mathcal{M}_{\text{coco}}$.

Hate Speech Detection. For hate speech, we analyze `HateBERT` [7], a BERT-base-uncased model [51] pre-trained on 1,478,348 messages from the RAL-E dataset and fine-tuned for hate speech classification on the HatEval dataset [52], which contains 13,000 tweets. The fine-tuned model achieves an F1-score of 0.645 on the evaluation set. We refer to this model as $\mathcal{M}_{\text{hate}}$.

3.2. SAE Training

Deepfake Detection. For SAE training, we evaluate sparsity level $k \in \{10, 20, 50\}$ and set the dictionary size m via an expansion factor e. For $\mathcal{M}_{\text{coco}}$, we use $e \in \{8, 16\}$, yielding $m \in \{8192, 16384\}$ and train on data from Cozzolino et al. [8]. For $\mathcal{M}_{\text{comm}}$, we use $e \in \{16, 32\}$, yielding $m \in \{6144, 12288\}$, and train on a subset of the data from Park et al. [46]. In the following, we refer to the trained SAEs as $\mathcal{S}_{\text{coco}}$ and $\mathcal{S}_{\text{comm}}$, respectively.

Hate Speech Detection. For training the SAE to analyze $\mathcal{M}_{\text{hate}}$, we evaluate sparsity levels $k \in \{10, 20, 50\}$ and set the dictionary size m directly, considering $m \in \{2^{14}, 2^{16}, 2^{18}\}$. We train the SAE on data from Caselli et al. [7]. We refer to the selected SAE as $\mathcal{S}_{\text{hate}}$.

SAE Selection. Following Section 2.1, we select SAEs with the highest FVE on the validation set. As shown in Table 1 $\mathcal{S}_{\text{comm}}$ and $\mathcal{S}_{\text{hate}}$ achieve FVEs of 0.83 and 0.78, while $\mathcal{S}_{\text{coco}}$ reaches 0.6. The L_0-norm indicates strong sparsity for $\mathcal{S}_{\text{coco}}$ and $\mathcal{S}_{\text{comm}}$, and moderate sparsity for $\mathcal{S}_{\text{hate}}$, likely due to its large dictionary and limited data. All SAEs have a high fraction of active features (≥ 0.9).

TABLE 1: Evaluation results of best-performing SAEs.

SAE	$\mathcal{S}_{\text{coco}}$	$\mathcal{S}_{\text{comm}}$	$\mathcal{S}_{\text{hate}}$
k	10	50	50
m	16,384	6,144	16,384
L_0-Norm	44.00	49.45	237.11
FVE	0.60	0.83	0.78
Frac-Alive	0.90	0.99	0.99
Linear Probe F1	0.88	0.99	0.90

3.3. SAE Feature Selection

For selecting the most important SAE features, we follow Section 2.2. Using the class-wise difference ranking, we iteratively select the first $i \in \{2^n \mid 0 \leq n \leq 7\}$ SAE features from the difference vector $\Delta\bar{\mathbf{z}}$ and train a logistic regression on the classification task. The final top i SAE features are selected based on the highest validation F1 score. The remaining hyperparameters for linear probe training are tuned via grid search (see Appendix B).

Probe performance for deepfake detection is 0.99 for deepfake detection using $\mathcal{S}_{\text{comm}}$ with $i = 64$ features, 0.88 using $\mathcal{S}_{\text{coco}}$ with $i = 128$ features, and 0.9 for hate speech detection using $\mathcal{S}_{\text{hate}}$ with $i = 128$ features (see Table 1).

3.4. Explanation Generation

As the final stage of our framework, we generate explanations for the top i SAE features by prompting `Gemini 2.5 Pro` [53] to assign concepts to SAE features based on the ten highest-activating examples in the training set. The LLM is instructed to output that no concept is present when the high-activating examples do not share a common concept. The full prompting setup for both domains and detailed model information is provided in Appendix C.

4. Feature-Level Analysis and SAE Pitfalls

After training and selecting suitable SAEs to understand the different classifiers, we can finally analyze the features identified by these SAEs. In the following, we describe the insights obtained from this analysis.

4.1. Qualitative Analysis

Table 2 depicts the three highest-ranked SAE features obtained using the feature selection procedure described in Section 2.2. Inspecting these top features reveals that SAEs frequently recover human-interpretable concepts for the analyzed deepfake models. However, the interpretability of the identified concepts varies depending on the dataset and model architecture. In particular, $\mathcal{S}_{\text{coco}}$ yields well-interpretable features that primarily reflect known generative model limitations, such as failures in face or object generation (#4450 and #13130), as well as spurious dataset artifacts like watermarks (#11494). In contrast, $\mathcal{S}_{\text{comm}}$ predominantly captures more generic generative characteristics, such as texture-related patterns (#972 and #3065),

TABLE 2: LLM-generated concept explanations for $\mathcal{S}_{\text{coco}}$ and $\mathcal{S}_{\text{comm}}$, with corresponding top-activating examples.

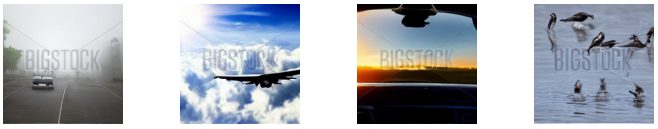
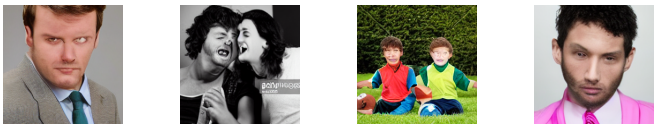
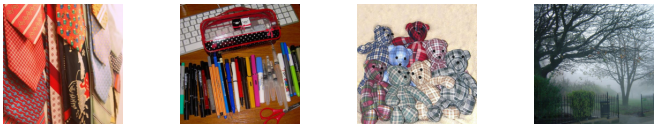
SAE	Feature ID	LLM Explanations	Top Activating Examples
$\mathcal{S}_{\text{coco}}$	#4450	Incoherent/malformed objects	
	#11494	Stock photo watermarks	
	#13130	Severe Facial Distortion	
$\mathcal{S}_{\text{comm}}$	#972	High-freq. repeating patterns	
	#1265	Structural/Geometric Artifacts	
	#3065	Synthetic High-Freq. Textures	

TABLE 3: LLM-generated concept explanations for $\mathcal{S}_{\text{hate}}$, with corresponding top-activating examples.

Feature ID	LLM Explanation	Top Activating Examples
#8767	*Misogynistic Sexual Denigration*	1. "@USER You're a vapid whore & one day you'll be ugly and begging for dick scraps" 2. "A hoe wants attention a woman wants respect." 3. "'y'all ain't shit' first of all, Idk why u think u some new bitch lmao u still that same old whore no one likes"
#3148	*Link-sharing post format*	1. ".Don't mess with these migrant dads SkimmLife URL" 2. "Trump just endorsed Kris Kobach, who made a name for himself with anti-immigrant views and allegations of voter fraud, for Kansas governor URL" 3. "I have an organic protein shake for you :face_blowing_a_kiss: URL"
#7339	*Misogynistic slurs for women*	1. "Hoe you watching I suggest you keep watching :winking_face: When a bitch ain't seen none yet :squinting_face_with_tongue:" 2. "Idgaf bout not 1 bitch lol yall hoes can die & i promise I won't b at the funeral :face_with_tears_of_joy: :hundred_points:" 3. "Ahh That's What I Get For Being A Whore :loudly_crying_face: :face_with_rolling_eyes: Now All These Bitches Wanna Talk :woman_facepalming_medium_skin_tone:"

whose associated concepts are more abstract and whose highly activating examples are harder to interpret for human analysts (e.g., #1265).

Similarly, Table 3 shows that the three top-ranked SAE features encoded in $\mathcal{S}_{\text{hate}}$ are largely human-interpretable and closely reflect properties of the training data. The identified concepts capture misogynistic language (#8767, #7339) and specific online communication patterns (#3148).

While the inspection of top-ranked SAE features yields human-interpretable concepts, the obtained concepts also highlight limitations in using SAEs for model interpretation. Closer examination of the $\mathcal{S}_{\text{coco}}$ features shows that examples activating SAE feature #4450 also frequently contain clearly visible watermarks, indicating overlap with feature #11494; a pattern also observed in $\mathcal{S}_{\text{comm}}$ (features #972 and #3065 encode closely related texture concepts) and $\mathcal{S}_{\text{hate}}$ (misogyny captured across multiple features).

Based on our qualitative analysis, we conclude that SAEs can identify meaningful concepts in deepfake and hate speech classifiers. At the same time, the presence of overlapping and redundant concepts among even the top-ranked features highlights challenges that limit the use of SAEs for model interpretation in security settings. In the following, we discuss these challenges in more detail.

4.2. Concept Redundancy

The manual inspection of the top concepts suggests that similar semantic concepts may be encoded by multiple SAE features. Ideally, an explanation of an ML model

should highlight only a small number of *unique*, human-interpretable concepts that are relevant for the downstream task [35]. Thus, redundant encodings introduce complexity in interpreting model behavior. To quantify our initial observation in greater detail, we apply the method of Li et al. [54], using the Jaccard index to assess similarity between input sets that activate individual SAE features. This analysis reveals feature pairs that co-activate in the same contexts, indicating they encode related concepts.

Our analysis identifies three pairs of SAE features in $\mathcal{S}_{\text{coco}}$ with a Jaccard index above 0.8, indicating that they frequently co-activate. Table 9 in Appendix D shows that these SAE features consistently activate for the same concepts (i.e., giraffe, tennis player, and pizza), providing quantitative evidence of concept-level redundancy within the SAE representation. While no feature pairs exceed this threshold for the other two SAEs, this does not contradict our qualitative findings. Instead, it suggests that redundancy in these models may manifest through more abstract or context-dependent activations not fully captured by simple input-set overlap. To assess whether a larger dictionary size improves the separability of concepts, we increased the size of m and observed the same pattern in $\mathcal{S}_{\text{coco}}$; the redundant concepts are documented in Appendix D.

Overall, our analysis indicates that SAEs frequently encode variations of the same semantic concept across multiple features, which limits their ability to provide concise and effective explanations of the decision-making processes of security models. Further research is therefore needed to better understand this redundancy and develop methods that reduce encoded concepts while preserving semantic distinctness. In particular, such methods must carefully balance the trade-off between reducing feature redundancy (i.e., *feature splitting* [22]) and avoiding the incorrect merging of semantically distinct features (i.e., *feature absorption* [55]).

Finding 1

SAE features may redundantly encode semantic concepts, limiting concise model interpretation.

4.3. Concept–Task Alignment

To understand the decision-making process of a classifier, the number of identified concepts should not only be small. Additionally, the concepts must be actually used by the classifier for the downstream task. Therefore, we further examine whether concepts that are selected in Section 2.2 are also relevant for the downstream task. To this end, we compare two quantities across the SAE latents: i) the class-wise SAE activation difference, which serves as the basis for the selection of the most important SAE features, and ii) the difference in the model output logit when we intervene on the latent, i.e., set the value to zero, as a measure of relevance of a latent for the classification task. Figure 1 shows the results of our analysis for $\mathcal{S}_{\text{coco}}$.

SAE features with the largest class-wise differences are not necessarily the ones that most strongly influence the model's decision-making. We observe a similar pattern for $\mathcal{S}_{\text{hate}}$, where features exhibiting high class-wise differences do not consistently produce the largest changes in the output logits when intervened upon. In contrast, for $\mathcal{S}_{\text{comm}}$, the selected features not only show big class-wise differences but also have the most impact on the model's predictions when intervened upon. The results for all SAEs are in Appendix E.

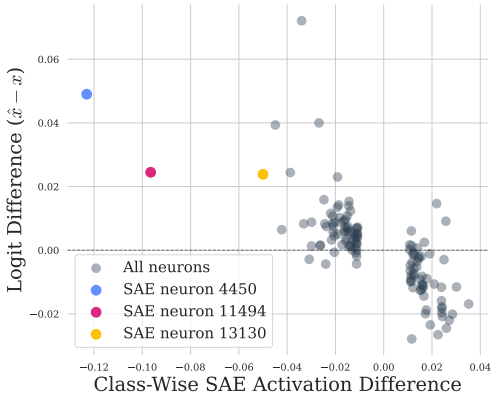


Figure 1: Relationship of class-wise activation difference and output-logit difference under intervention for $\mathcal{S}_{\text{coco}}$. Latents displayed in Table 2 are highlighted.

These findings reveal that features with the highest discriminatory power, i.e., biggest class-wise activation differences are not always classification-relevant. Therefore, the selection of important SAE features should be linked to the output of the model and not happen exclusively on the data side [56], [57].

Finding 2

SAE features that rank highly by activation are not necessarily relevant to the downstream task.

4.4. Explanation Consistency

To explain SAE concepts, a common approach is to find shared patterns in input images that strongly activate a given SAE feature. However, the large number of SAE features makes manual interpretation cumbersome and often infeasible. As a result, recent work has increasingly leveraged LLMs for this purpose [26]. Unfortunately, using LLMs to automatically generate concept explanations introduces additional risks [58]. In particular, LLMs may hallucinate and produce unreliable interpretations [59].

We investigate how the LLM choice affects the resulting concept explanations by additionally generating concept annotations using `Claude Sonnet 4.5` and comparing them to those obtained with `Gemini 2.5 Pro`. Detailed descriptions of the LLMs can be found in Appendix F. Since LLMs do not always identify common patterns in the input images, we conduct a first, coarse consistency check by measuring how often both LLMs assign a concept to the same SAE feature, compared to cases in which only one LLM produces an annotation.

Figure 2 shows the results of this analysis. Interestingly, we observe substantial disagreement between the two LLMs for $\mathcal{S}_{\text{coco}}$, whereas agreement is considerably higher for the other two SAEs. However, agreement in assignment does not imply semantic alignment. Therefore, we compute the pairwise cosine similarity between concept embeddings using `all-MiniLM-L6-v2` from the *SentenceTransformers* library [60] (Figure 5 in Appendix F shows the distribution of pairwise cosine similarity for all SAEs). Table 4 presents the three SAE features with the lowest semantic similarity between concept annotations.

TABLE 4: Explanation agreement, measured as cosine similarity between concept embeddings of explanations generated by `Gemini 2.5 Pro` and `Claude Sonnet 4.5`.

SAE	Feature ID	Agreement	`Gemini 2.5 Pro` Explanation	`Claude Sonnet 4.5` Explanation
$\mathcal{S}_{\text{coco}}$	#6891	-0.1182	Malformed or Incoherent Subjects	Horizontal linear fence structures
	#3157	0.0063	Malformed Semantic Objects	Authentic photographic crowd scenes
	#14169	0.0195	Semantic Form Breakdown	Vintage film grain texture
$\mathcal{S}_{\text{comm}}$	#1366	0.0824	Synthetic painterly surface textures	Radial motion blur artifacts
	#861	0.0956	Malformed or Fused Appendages	Overly smooth synthetic texture
	#4624	0.1078	Isolated subject, uniform background	Synthetic product photography rendering
$\mathcal{S}_{\text{hate}}$	#1293	0.0209	End of clause or sentence	Second-person direct address
	#10261	0.0489	Concluding or Aphoristic Statements	Informal conversational social media
	#6613	0.0714	In-group vs. Out-group Conflict	Anti-immigration political rhetoric

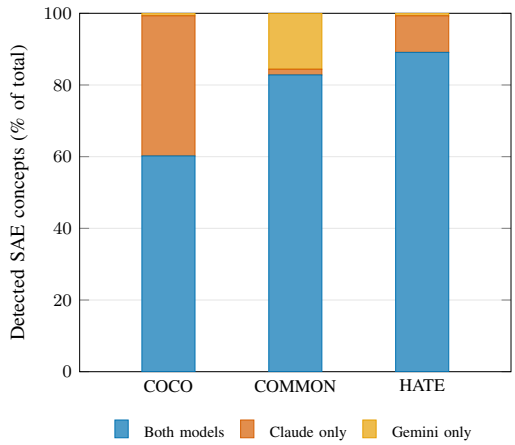


Figure 2: Overlap of concept detection for `Claude Sonnet 4.5` and `Gemini 2.5 Pro`

Although both LLMs assign a concept in these cases, the resulting explanations can differ substantially depending on the chosen model. While quantifying the semantic equivalence of annotations remains challenging, these examples illustrate that different LLMs can produce considerably different annotations.

We note that this limitation of SAE interpretability may conceivably also extend to human annotators. Given the same limited set of highest-activating examples, two different human annotators may find different commonalities, leading to potentially incorrect concept labels.

Finding 3

LLM-generated explanations for SAE features can differ considerably across LLMs.

5. Related Work

Sparse Autoencoder for Interpretability. Based on the linear representation and superposition hypotheses [61], SAEs emerged as a promising unsupervised tool to find human-interpretable features in large models [22], [23], [62]. Since then, various variations have shown improved results, such as TopK [63], BatchTopK [30], or Matryoshka SAEs [64]. SAEs have been used to find human-understandable features [24], [63], [65] and circuits [66] in LLMs. Recently, they have also been extended to interpret large vision-language models (LVLMs) [28], [29], including domain-specific applications such as medical LVLMs [56], [67]–[69].

Sparse Autoencoder for Classification. SAEs have been used as feature extractors to build interpretable classifiers [61]. Probing downstream tasks with SAE-extracted concepts also serves as an evaluation method to assess whether learned SAE features are meaningful for downstream tasks [34], [63]. Melo et al. [70] use features of pre-trained SAEs for software bug detection, demonstrating improvements compared to fine-tuned transformer baselines. Similarly, Gallifant et al. [71] use pre-trained SAEs to evaluate SAE feature extraction strategies, reporting performance gains over linear probing approaches in toxicity detection. In contrast, Kantamneni et al. [31] found no consistent performance improvement from SAEs over baselines like linear probing across 113 binary classification tasks.

Limitations of SAEs. Although there has been ample anecdotal evidence that SAEs can isolate interpretable features in the basis directions of their feature space (e.g. [23], [24], [67], [69]), reliably monosemantic feature decompositions have not yet been achieved [22], [55], [72]. Failure modes may, on the one hand, stem from nonlinear feature representations, which violate the linear representation hypothesis, as well as from an inherent trade-off between monosemanticity and sparsity. *Feature splitting* [22] is a phenomenon in which concepts are split across multiple SAE features to increase sparsity, whereas *feature absorption* [55] refers to the case in which features coding for higher-level hierarchical concepts are partially merged into their lower-level descendant features. Furthermore, the non-deterministic training procedure can lead to different feature sets identified by the SAE [73], and different SAE latent space sizes uncover different parts of the concept hierarchy [72].

6. Conclusion

We investigated the capabilities of SAEs for interpreting fine-tuned classification models in two security-related tasks, i.e., deepfake and hate speech detection. By introducing a framework based on established interpretation techniques, we demonstrate that SAEs can successfully generate meaningful concept explanations for deepfake and hate speech classifiers. However, our evaluation reveals three challenges to SAE adoption: concept redundancy, classification-irrelevant concepts, and inconsistent annotations. Addressing these challenges is essential for the utility of SAEs in security interpretability.

Acknowledgment

This research was supported by the Vienna Science and Technology Fund (WWTF) through the BREADS project (10.47379/VRG23011).

References

[1] S. Dambra, Y. Han, S. Aonzo, P. Kotzias, A. Vitale, J. Caballero, D. Balzarotti, and L. Bilge, "Decoding the secrets of machine learning in malware classification: A deep dive into datasets, feature extraction, and model performance," in *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2023.

[2] A. S. Li, A. Iyengar, A. Kundu, and E. Bertino, "Revisiting concept drift in windows malware detection: Adaptation to real drifted malware with minimal samples," in *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2025.

[3] Z. Huang, J. Hu, X. Li, Y. He, X. Zhao, B. Peng, B. Wu, X. Huang, and G. Cheng, "SIDA: social media image deepfake detection, localization and explanation with large multimodal model," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[4] F. Yamaguchi, A. Maier, H. Gascon, and K. Rieck, "Automatic inference of search patterns for taint-style vulnerabilities," in *Proc. of IEEE Symposium on Security and Privacy (S&P)*, 2015.

[5] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong, "Vuldeepecker: A deep learning-based system for vulnerability detection," in *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2018.

[6] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[7] T. Caselli, V. Basile, J. Mitrović, and M. Granitzer, "HateBERT: Retraining BERT for abusive language detection in English," in *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, 2021.

[8] D. Cozzolino, G. Poggi, R. Corvi, M. Nießner, and L. Verdoliva, "Raising the bar of ai-generated image detection with clip," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[9] G. Zhou, X. Guo, Z. Liu, T. Li, Q. Li, and K. Xu, "Trafficformer: An efficient pre-trained model for traffic data," in *Proc. of IEEE Symposium on Security and Privacy (S&P)*, 2025.

[10] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, 2015.

[11] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.

[12] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR) Workshop Track*, 2014.

[13] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[14] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards automatic concept-based explanations," in *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[15] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar, "On completeness-aware concept-based explanations in deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[16] R. Achtibat, M. Dreyer, I. Eisenbraun, S. Bosse, T. Wiegand, W. Samek, and S. Lapuschkin, "From attribution maps to human-understandable explanations through concept relevance propagation," *Nature Machine Intelligence*, vol. 5, no. 9, pp. 1006–1019, 2023.

[17] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[18] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, "Evaluating explanation methods for deep learning in security," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[19] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer, "Sok: Explainable machine learning for computer security applications," in *IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 2023.

[20] A. Galli, V. La Gatta, V. Moscato, M. Postiglione, and G. Sperlì, "Explainability in ai-based behavioral malware detection systems," *Computers & Security*, vol. 141, p. 103842, 2024.

[21] M. Noppel and C. Wressnegger, "Sok: Explainable machine learning in adversarial environments," in *IEEE Symposium on Security and Privacy (SP)*, 2024.

[22] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. L. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023.

[23] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," in *International Conference on Representation Learning (ICLR)*, 2024.

[24] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, A. Tamkin, E. Durmus, T. Hume, F. Mosconi, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan, "Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet," *Transformer Circuits Thread*, 2024.

[25] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?," *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[26] G. S. Paulo, A. T. Mallen, C. Juang, and N. Belrose, "Automatically interpreting millions of features in large language models," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2025.

[27] B. Deng, Y. Wan, B. Yang, Y. Zhang, and F. Feng, "Unveiling language-specific features in large language models via sparse autoencoders," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

[28] M. Pach, S. Karthik, Q. Bouniot, S. Belongie, and Z. Akata, "Sparse autoencoders learn monosemantic features in vision-language models," in *Advances in Neural Information Processing Systems (NIPS)*, 2025.

[29] V. Zaigrajew, H. Baniecki, and P. Biecek, "Interpreting CLIP with hierarchical sparse autoencoders," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2025.

[30] B. Bussmann, P. Leask, and N. Nanda, "Batchtopk sparse autoencoders," in *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024.

[31] S. Kantamneni, J. Engels, S. Rajamanoharan, M. Tegmark, and N. Nanda, "Are sparse autoencoders useful? a case study in sparse probing," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2025.

[32] S. Bills, N. Cammarata, D. Mossing, H. Tillman, L. Gao, G. Goh, I. Sutskever, J. Leike, J. Wu, and W. Saunders, "Language models can explain neurons in language models," *OpenAI Research*, 2023.

[33] A. K. Samuel Marks and A. Mueller, "dictionary_learning." https://github.com/saprmarks/dictionary_learning, 2024.

[34] A. Karvonen, C. Rager, J. Lin, C. Tigges, J. I. Bloom, D. Chanin, Y.-T. Lau, E. Farrell, C. S. McDougall, K. Ayonrinde, D. Till, M. Wearden, A. Conmy, S. Marks, and N. Nanda, "SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2025.

[35] L. Longo, M. Brcic, F. Cabitza, J. Choi, R. Confalonieri, J. Del Ser, R. Guidotti, Y. Hayashi, F. Herrera, A. Holzinger, R. Jiang, H. Khosravi, F. Lecue, G. Malgieri, A. Páez, W. Samek, J. Schneider, T. Speith, and S. Stumpf, "Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions," *Information Fusion*, vol. 106, 2024.

[36] S. M. Abdullah, A. Cheruvu, S. Kanchi, T. Chung, P. Gao, M. Jadliwala, and B. Viswanath, "An analysis of recent advances in deepfake image detection in an evolving threat landscape," in *IEEE Symposium on Security and Privacy (SP)*, 2024.

[37] K. Thomas, D. Akhawe, M. Bailey, D. Boneh, E. Bursztein, S. Consolvo, N. Dell, Z. Durumeric, P. G. Kelley, D. Kumar, *et al.*, "Sok: Hate, harassment, and the changing landscape of online abuse," in *IEEE Symposium on Security and Privacy (SP)*, 2021.

[38] D. Velásquez-Salamanca, M. Á. Martín-Pascual, and C. Andreu-Sánchez, "Interpretation of ai-generated vs. human-made images," *Journal of Imaging*, vol. 11, no. 7, p. 227, 2025.

[39] J. Wilson, "Hundreds of nonconsensual AI images being created by Grok on X, data shows," Jan. 8, 2026. [Accessed: Jan. 23, 2026].

[40] J. Frank, T. Eisenhofer, L. Schönherr, A. Fischer, D. Kolossa, and T. Holz, "Leveraging frequency analysis for deep fake image recognition," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2020.

[41] U. Ojha, Y. Li, and Y. J. Lee, "Towards universal fake image detectors that generalize across generative models," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[42] A. C. for Technology and Society, "Online hate and harassment: The american experience 2023," 2023.

[43] P. Röttger, B. Vidgen, D. Nguyen, Z. Waseem, H. Margetts, and J. Pierrehumbert, "HateCheck: Functional tests for hate speech detection models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021.

[44] X. Shen, Y. Wu, Y. Qu, M. Backes, S. Zannettou, and Y. Zhang, "HateBench: Benchmarking Hate Speech Detectors on LLM-Generated Content and Hate Campaigns," in *Proc. of USENIX Security Symposium*, 2025.

[45] N. Vishwamitra, K. Guo, F. T. Romit, and I. Ondracek, "Moderating new waves of online hate with chain-of-thought reasoning in large language models," in *IEEE Symposium on Security and Privacy (SP)*, 2024.

[46] J. Park and A. Owens, "Community forensics: Using thousands of generators to train fake image detectors," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[47] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[48] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2021.

[49] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[50] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV: European Conference on Computer Vision*, 2014.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

[52] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, "SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter," in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019.

[53] G. Gemini Team, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities.," *arXiv preprint arXiv:2507.06261*, 2025.

[54] Y. Li, E. J. Michaud, D. D. Baek, J. Engels, X. Sun, and M. Tegmark, "The geometry of concepts: Sparse autoencoder feature structure," *Entropy*, vol. 27, no. 4, p. 344, 2025.

[55] D. Chanin, J. Wilken-Smith, T. Dulka, H. Bhatnagar, and J. Bloom, "A is for absorption: Studying feature splitting and absorption in sparse autoencoders," in *Advances in Neural Information Processing Systems (NIPS)*, 2024.

[56] M. Dreyer, L. Hufe, J. Berend, T. Wiegand, S. Lapuschkin, and W. Samek, "From what to how: Attributing clip's latent components reveals unexpected semantic reliance," *arXiv preprint arXiv:2505.20229*, 2025.

[57] D. Shu, X. Wu, H. Zhao, M. Du, and N. Liu, "Beyond input activations: Identifying influential latents by gradient sparse autoencoders," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

[58] J. Evertz, N. Risse, N. Neuer, A. Müller, P. Normann, G. Sapia, S. Gupta, D. Pape, S. Shaw, D. Srivastav, *et al.*, "Chasing shadows: Pitfalls in llm security research," in *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2025.

[59] H. Orgad, M. Toker, Z. Gekhman, R. Reichart, I. Szpektor, H. Kotek, and Y. Belinkov, "Llms know more than they show: On the intrinsic representation of llm hallucinations," in *International Conference on Learning Representations (ICLR)*, 2025.

[60] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[61] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah, "Toy models of superposition," *arXiv preprint arXiv:2209.10652*, 2022.

[62] L. Bereska and E. Gavves, "Mechanistic interpretability for ai safety a review," *Transactions on Machine Learning Research (TMLR)*, 2024.

[63] L. Gao, T. Dupré la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu, "Scaling and evaluating sparse autoencoders," in *International Conference on Learning Representations (ICLR)*, 2025.

[64] B. Bussmann, N. Nabeshima, A. Karvonen, and N. Nanda, "Learning multi-level features with matryoshka sparse autoencoders," in *Proc. of Int. Conference on Machine Learning (ICML)*, 2025.

[65] J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Pearce, N. L. Turner, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn, A. Jones, A. Persic, Z. Qi, T. B. Thompson, S. Zimmerman, K. Rivoire, T. Conerly, C. Olah, and J. Batson, "On the biology of a large language model," *Transformer Circuits Thread*, 2025.

[66] S. Marks, C. Rager, E. J. Michaud, Y. Belinkov, D. Bau, and A. Mueller, "Sparse feature circuits: Discovering and editing interpretable causal graphs in language models," in *International Conference on Learning Representations (ICLR)*.

[67] N. M. Le, C. Shen, N. Patel, C. Shah, D. Sanghavi, B. Martin, A. Eng, D. Shenker, H. Padigela, R. Biju, *et al.*, "Learning biologically relevant features in a pathology foundation model using sparse autoencoders," *arXiv preprint arXiv:2407.10785*, 2024.

[68] S. Sun, D. van Midden, G. Litjens, and C. F. Baumgartner, "Prototype-based multiple instance learning for gigapixel whole slide image classification," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2025.

[69] M. F. Dasdelen, H. Lim, M. Buck, K. S. Götze, C. Marr, and S. Schneider, "CytoSAE: Interpretable Cell Embeddings for Hematology," in *Proceedings of Medical Image Computing and Computer Assisted Intervention*, 2025.

[70] R. Melo, C. Mamede, A. Catarino, R. Abreu, and H. L. Cardoso, "Are sparse autoencoders useful for java function bug detection?," *arXiv preprint arXiv:2505.10375*, 2025.

[71] J. Gallifant, S. Chen, K. Sasse, H. Aerts, T. Hartvigsen, and D. Bitterman, "Sparse autoencoder features for classifications and transferability," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

[72] P. Leask, B. Bussmann, M. Pearce, J. Bloom, C. Tigges, N. Al Moubayed, L. Sharkey, and N. Nanda, "Sparse autoencoders do not find canonical units of analysis," in *International Conference on Learning Representations (ICLR)*, 2025.

[73] G. Paulo and N. Belrose, "Sparse autoencoders trained on the same data learn different features," in *Advances in Neural Information Processing Systems (NIPS)*, 2025.

Appendix A.
SAE Details

SAE architecture. The trained SAEs follow the typical encoder-decoder architecture: The encoder is a linear layer with weights of size $m \times n$, followed by an element-wise nonlinearity. The decoder is a linear layer with weights of size $n \times m$.

SAE training. All SAEs are trained to minimize the following loss:

$$\mathcal{L}(\mathbf{x}) := \underbrace{\|\mathbf{x} - \tilde{\mathbf{x}}(f(\mathbf{x}))\|_2^2}_{\mathcal{L}_{\text{reconstruction}}} + \alpha\, \mathcal{L}_{\text{aux}}, \tag{6}$$

- For preventing dead latents, we employ an auxiliary loss. We follow [63] and define αL_{aux} as the loss between the original instance $\mathbf{x}$ and its reconstruction using the top-k_{aux} dead latents. We set $k_{aux} = 512$ and $\alpha = (1/32)$. This loss enforces that latents that are not active for a long time, i.e., will also contribute to the reconstruction performance.
- We use the `dictionary_library`[33] developed by Samuel Marks et al. and the extension for CLIP models developed by Pach et al. [28].

Deepfake. For the $\mathcal{S}_{\text{comm}}$ we take the $\mathcal{M}_{\text{comm}}$ activations of layer 11. For $\mathcal{S}_{\text{coco}}$, we take the $\mathcal{M}_{\text{coco}}$ activations of layer 22 for training. We split the raw activations into training and validation (80:20) splits. In addition to the grid search for the hyperparameters m and k, we select the remaining hyperparameters as depicted in Table 5.

TABLE 5: Hyperparameter settings for $\mathcal{S}_{\text{coco}}$ and $\mathcal{S}_{\text{comm}}$ training.

Parameter	Value
Training steps	$100,000$
Batch size	$4,096$
Scale	dict_size$/2^{14}$
Learning rate	$2 \times 10^{-4}/\text{scale}^{0.5}$
AuxK α	0.03
Decay start	$10,999$
Threshold β	0.999
Remaining parameters	Default settings

Hate Spech. For $\mathcal{S}_{\text{hate}}$, we take the $\mathcal{M}_{\text{hate}}$ activations of layer 12. We split the raw activations into training and validation (90:10) splits. In addition to the grid search for the hyperparameters m and k, we select the remaining hyperparameters as depicted in Table 6.

TABLE 6: Hyperparameter settings for $\mathcal{S}_{\text{hate}}$ training.

Paramter	Value
Training steps	$30,000$
Batch size	$2,048$
AuxK α	$1/32$
Scale	dict_size$/2^{14}$
Learning rate	$2 \times 10^{-4}/\text{scale}^{0.5}$
Threshold β	0.999
Other parameters	Default settings

Appendix B.
SAE Feature Selection

Hyperparameter selection for the sparse probe is performed over the parameter grid shown in Table 7.

TABLE 7: Hyperparameter grid for linear probe.

Parameter	Values
Penalty (L1)	`l1`
Solver	`liblinear`, `saga`
C	$0.01, 0.1, 1, 10, 100$
Max iterations	$5,000, 8,000$
Penalty (L2)	`l2`
Solver	`liblinear`, `sag`, `saga`
C	$0.01, 0.1, 1, 10, 100$
Max iterations	$5,000, 8,000$
Penalty (ElasticNet)	`elasticnet`
Solver	`saga`
C	$0.01, 0.1, 1, 10, 100$
L_1 ratio	$0.1, 0.5, 0.9$
Max iterations	$5,000, 8,000$

The hyperparameters of the best-performing linear probes are depicted in Table 8.

TABLE 8: Hyperparameter configurations for the best-performing linear probes for $\mathcal{S}_{\text{coco}}$, $\mathcal{S}_{\text{comm}}$, and $\mathcal{S}_{\text{hate}}$.

Parameter	$\mathcal{S}_{\text{coco}}$	$\mathcal{S}_{\text{comm}}$	$\mathcal{S}_{\text{hate}}$
Top-k	128	64	128
C	0.1	1.0	0.01
Max iterations	$5,000$	$5,000$	$5,000$
Penalty	`l2`	`l1`	`l2`
Solver	`liblinear`	`saga`	`sag`

Appendix C.
Explanation Generation

Annotation Models. To generate explanations with an LLM, we create an API setup by using the openrouter.ai interface (https://openrouter.ai). Via the interface, we access the model named *Google: Gemini 2.5 Pro* with the model identifier `google/gemini-2.5-pro`, which was created on June 17, 2025 available under https://openrouter.ai/google/gemini-2.5-pro.

Prompting. We used the prompts shown in Figure 3 to generate the concept explanations:

Appendix D.
Concept Redundancy

Table 9 shows the redundant concepts found in $\mathcal{S}_{\text{coco}}$ with a dictionary size of $m = 16384$. Table 10 shows redundant conocepts found in $\mathcal{S}_{\text{coco}}$ with a dictionary size of $m = 24576$.

Appendix E.
Task-Feature Alignment

Figure 4 shows the relationship of per-latent relevance for SAE reconstruction and classification for $\mathcal{S}_{\text{coco}}$, $\mathcal{S}_{\text{comm}}$, and $\mathcal{S}_{\text{hate}}$.

System Prompt: Deepfake

You are an expert AI forensic analyst specializing in identifying visual features and artifacts indicative of synthetic image generation (Deepfakes) or digital manipulation.

TASK:
You are given a set of images that strongly activate a single neuron in a Sparse Autoencoder (SAE) trained for deepfake classification. Your task is to carefully analyze and describe what visual and semantic concepts these images have in common.

VALID CONCEPT TYPES:
- **Semantic Concepts:** e.g., objects (unnatural eye reflections, asymmetric facial features), actions, settings (warped background, inconsistent light sources).
- **Low-level visual features:** e.g., GAN-generated textures, diffusion blurring, checkerboard artifacts.

ANALYSIS CONSTRAINTS:

- Analyze the holistic visual context, not just isolated pixel groups.
- The concept MUST be present in every described example provided.
- The domain is deepfake and synthetic media detection.
- If no single consistent concept applies, respond with: "No uniform concept is present across all examples."

OUTPUT RULES (STRICT):

- Output MUST be a structured text block following the exact labels.
- Do NOT include JSON, markdown code blocks, or conversational filler.
- Each field must start on a new line.

System Prompt: Hate Speech

You are an expert AI interpretability analyst specializing in understanding features learned by Sparse Autoencoders (SAEs) trained on Large Language Models (LLMs) for hate speech detection.

TASK:
You will be given multiple text examples that strongly activate a single SAE neuron. Your task is to identify the single underlying concept that is consistently present across ALL examples. This concept may be semantic, syntactic, or stylistic in nature.

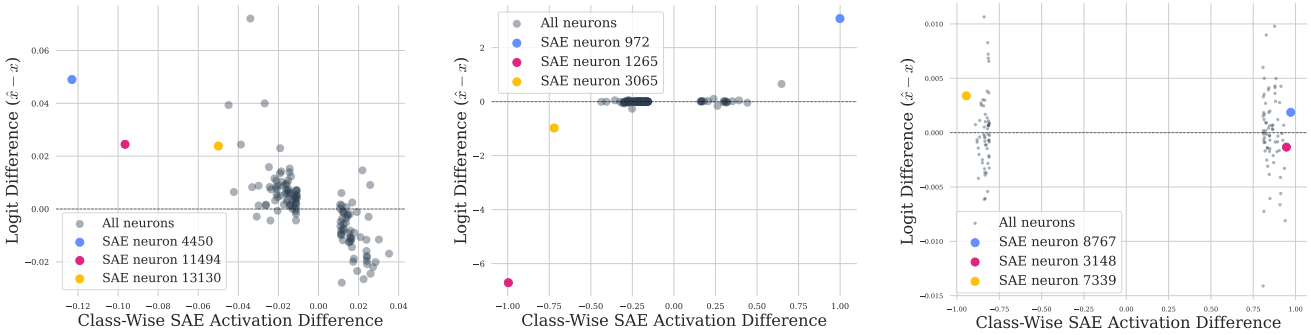
VALID CONCEPT TYPES:
- **Semantic meaning:** e.g., group denigration, threats, slurs.
- **Syntactic structure:** e.g., imperatives, rhetorical questions.
- **Textual or stylistic patterns:** e.g., news headline style, slogan-like phrasing, all-caps emphasis, telegraphic language, hashtag-driven discourse.

ANALYSIS CONSTRAINTS:
- Analyze full context, not isolated keywords.
- The concept MUST be present in every example.
- Focus on the most specific shared feature, not a vague theme.
- The domain is hate speech detection.
- If no single consistent concept applies, respond with: "No uniform semantic or syntactical concept is present across all examples."

REQUIRED OUTPUT FORMAT:

- Output MUST be a single valid JSON object.
- Do NOT include explanations, markdown, or conversational text outside the JSON.
- The `neuron_id` MUST exactly match the neuron ID provided in the user prompt.

Figure 3: System prompts for `Gemini 2.5 Pro` and `Claude Sonnet 4.5` used for automated concept annotation.



(a) $\mathcal{S}_{\text{coco}}$ (b) $\mathcal{S}_{\text{comm}}$ (c) $\mathcal{S}_{\text{hate}}$

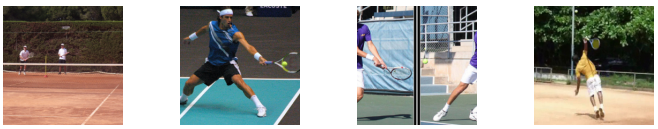
Figure 4: Relationship of class-wise activation difference and output-logit difference under intervention. Latents displayed in Table 2 and Table 3 are highlighted.

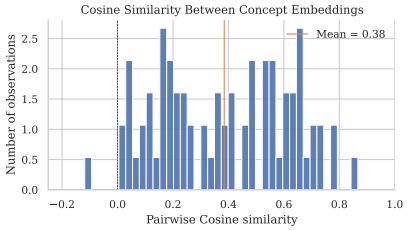
Appendix F. Explanation Consistency

Annotation Models. For the consistency experiment, we use `Claude Sonnet 4.5` as the second LLM. We create an API setup by using the openrouter.ai interface https://openrouter.ai. Via the interface, we access the model named *Anthropic: Claude Sonnet 4.5* with the model identifier `anthropic/claude-sonnet-4.5` created on September 25th, 2025, available under https://openrouter.ai/anthropic/claude-sonnet-4.5

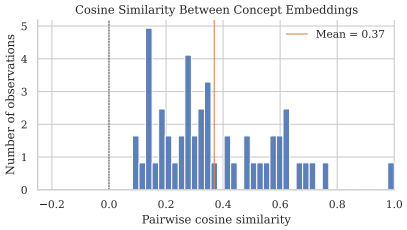
Pairwise Cosine Similarity of Concept Embeddings. Figure 5 shows the distribution of the pairwise cosine similarity for the concept explanations for $\mathcal{S}_{\text{coco}}$, $\mathcal{S}_{\text{comm}}$, and $\mathcal{S}_{\text{hate}}$.

TABLE 9: SAE feature pairs comparison of $\mathcal{S}_{\text{coco}}$ $m = 16384$ with Jaccard index ≥ 0.8

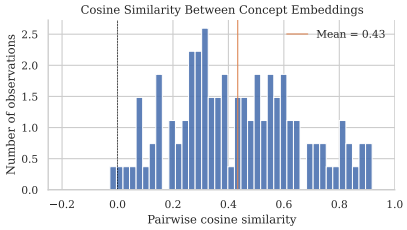
Feature ID	LLM Interpretation	Top Activating Examples
#6373	Unnatural Giraffe Anatomical Distortions.	
#1935	Malformed Giraffe Anatomy.	
#5595	Tennis match context.	
#7984	People playing tennis.	
#15833	Images featuring pizza.	
#12431	Unnatural Pizza-like Textures.	



(a) $\mathcal{S}_{\text{coco}}$



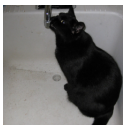
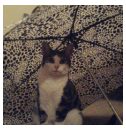
(b) $\mathcal{S}_{\text{comm}}$



(c) $\mathcal{S}_{\text{hate}}$

Figure 5: Pairwise cosine similarity of `Gemini 2.5 Pro` and `Claude Sonnet 4.5` concept embeddings across SAEs.

TABLE 10: SAE feature pairs comparison of $\mathcal{S}_{\text{coco}}$ $m = 24576$ with Jaccard index ≥ 0.8

Feature ID	LLM Interpretation	Top Activating Examples			
#8636	Fake cats.				
#15487	Fake cats.				
#1253	Fake trains.				
#17773	Fake train tracks.				
#11287	Domestic cats.				
#13242	Domestic cats.				