Exploring Text Classification Models with Sparse Autoencoders

Daniel Kerrigan* Capital One
Brian Barr† Capital One
Enrico Bertini‡ Northeastern University

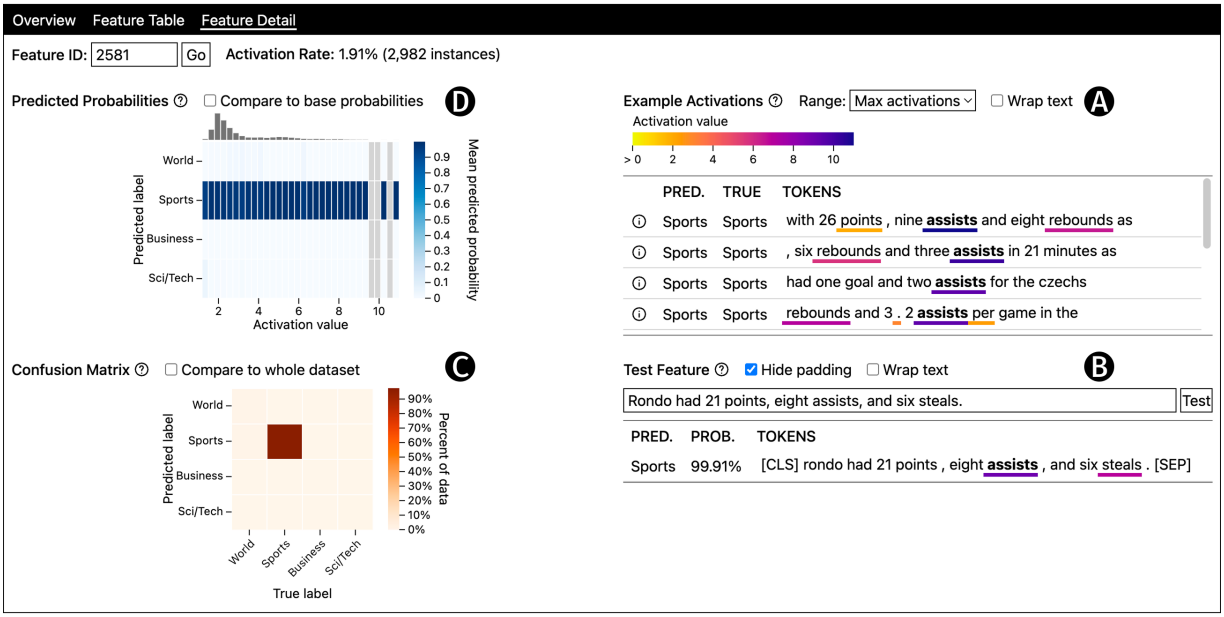

Figure 1: The Feature Detail tab in SAEfarer for a model that classifies the topic of news articles. This feature activates on tokens relating to sports statistics. A) The example activations shows snippets from instances that activate the feature. B) The user can enter their own piece of text to see if it activates the feature. C) A confusion matrix calculated from the instances that activate the feature. D) The relationship between the feature's activation value and the model's predicted probabilities for each class.

Abstract

As language models (LMs) rise in prominence, there is interest in making them more transparent in order to better understand their internal behavior. Recent interpretability work has focused on using sparse autoencoders (SAEs) to break down neuron activations at a given layer in the LM into human-understandable features, where each feature represents a concept that the model has learned. In this paper, we share work on using SAEs to analyze the behavior of text classification LMs. We present techniques for exploring the relationships between the SAE's features and the model's predictions and errors. We integrate these techniques into SAEfarer, a tool for analyzing concepts learned by text classification LMs. We assess SAEfarer in an expert pilot evaluation with five Ph.D. students.

Index Terms: Sparse autoencoder, dictionary learning, mechanistic interpretability, text classification, visual analytics.

1 Introduction

When interpreting and explaining the behavior of machine learning (ML) models trained on tabular data, much of the focus is on analyzing the relationships between the models' features and predictions (e.g., [9, 18]). One difficulty in analyzing the behavior of models trained on unstructured data like images or text is that they lack interpretable features in the traditional sense. Individual pixels or tokens that form the inputs to the model may be too granular to promote a higher-level conceptual understanding of the model from analyzing them [15]. To address this, researchers have developed techniques to extract concepts learned by image and language models (LMs) (e.g., [11, 3]). These approaches can be viewed as ways to identify higher-level "features" of the models.

Visual analytics research on concept-level analysis of neural networks has focused primarily on image models [24, 14, 29, 30] and generative LMs [3, 17]. In this work, we present a concept-based visual analytics approach for exploring the behavior of text classification LMs. Our approach relies on sparse autoencoders (SAEs) to identify features learned by the model [3, 6].

One challenge in working with SAEs is the large number of features that they extract. When an ML practitioner is exploring the behavior of the model, it is impractical for them to analyze all of the identified features. It is necessary to have ways to help guide the practitioner's exploration to surface features of particular interest. When analyzing text classification LMs, of key interest are features that correlate with the model's predictions and errors. Analyzing the features that correlate with the model's predictions can help the practitioner check if the model has learned intuitive concepts that are useful for classification. Analyzing the features that correlate with the model's errors can help the practitioner identify patterns in the instances that the model makes mistakes on.

In this work, we first propose ways of ranking SAE features based on their relationships with the text classification model's predictions and errors. Second, we present SAEfarer, an interactive visualization tool for exploring SAE features for text classification

*e-mail: daniel.kerrigan2@capitalone.com
†e-mail: brian.barr@capitalone.com
‡e-mail: e.bertini@northeastern.edu

LMs. Finally, we discuss an expert pilot evaluation with five Ph.D. students whose research relates to machine learning interpretability.

2 Related Work

2.1 Identifying Concepts in Models

TCAV by Kim et al. [15] identified the direction of predefined concepts in the activation space of a neural network and measured the relationship between those concepts and the model's predictions. They demonstrated their approach on image models. Ghorbani et al. extended this work with ACE [11] in order to overcome the limitation of requiring predefined concepts. Later work by Yeh et al. [28] proposed an approach to automatically identify a set of concepts that sufficiently explain the model's behavior, which they demonstrated for both image and text classification models.

Bricken et al. [3] and Cunningham et al. [6] presented notable initial work on using SAEs to identify interpretable concepts learned by transformer-based LMs. These initial works extracted concepts from small or toy models, but later work applied SAEs to larger LMs such as GPT-4 [10], Claude 3 Sonnet [26], and Gemma 2 and 3 [16, 22]. Extracting concepts with SAEs is unsupervised and does not require labeled data or a human to predefine concepts. Despite not using labels to identify concepts, SAEs are still a useful technique to apply with labeled data.

Movva et al. [23] proposed a method called HypotheSAEs that uses SAEs to generate hypotheses about the relationship between a text dataset and a target variable. Although both SAEfarer and HypotheSAEs use SAEs to analyze labeled text data, the works have different focuses. SAEfarer is a visualization system for exploring the relationships between SAE features and a model's predictions and errors. In contrast, HypotheSAEs is a method for generating natural language hypotheses about a labeled text dataset and does not focus on analyzing the behavior of a classification model.

2.2 Concept Visualization Systems

Visual analytics research in concept-level analysis of neural networks initially focused on image models. Zhao et al. [30] proposed an active learning system called ConceptExtract for interactively identifying concepts. With NeuroCartography, Park et al. [24] created a tool to identify clusters of neurons that activate for the same concept and to analyze the relationship between concepts. Huang et al. [14] presented ConceptExplainer, a system for exploring concepts based on TCAV and ACE. Zhang et al. [29] proposed ConceptSlicer, which uses concepts to identify subsets of the data where the model performs poorly.

For LMs, researchers have developed tools for analyzing features identified by SAEs. Bricken et al. [3] is particularly influential to our work, as they developed, to our knowledge, the first visualization tool for ranking and exploring SAE features. McDougall's SAE Visualizer [21] and the SAEDashboard by Decode Research [7] are open-source tools based on the work by Bricken et al. [3]. Neuronpedia [17] is a notable SAE exploration and visualization tool that provides additional functionality such as testing feature activations on custom text and using SAE features to steer the model's output. We have taken inspiration from these works and have adopted functionality from them in SAEfarer. However, with SAEfarer, our focus is on using SAEs to analyze text classification LMs, rather than generative LMs. For classification LMs, these existing tools would not directly guide the user to features that are relevant to the classification task and would not show the features' relationships to the model's predictions. SAE Semantic Explorer by Yan et al. [27] is another SAE tool for language, but its focus is on analyzing the relationships between features for specific concepts that the user is interested in.

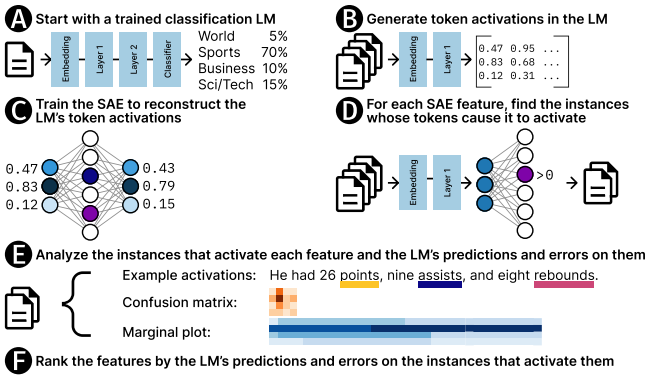

Figure 2: Our approach to training and analyzing SAEs.

3 Training and Analyzing SAEs

Figure 2 provides an overview of how we use SAEs in SAEfarer. We begin with a trained classification LM (Fig. 2A). As an example, we will use a model[1] that was trained to classify the topic of articles from the AG News dataset [13] as *World*, *Sports*, *Business*, or *Sci/Tech*. To make a classification, an instance is split into a series of tokens before being fed into the model. The model then outputs a score for each class and the class with the highest score is chosen as the predicted label for the instance.

The first step to training an SAE is to feed instances into the LM and generate activations for each token at a chosen layer in the LM (Fig. 2B). The activations for each token are represented as a high-dimensional vector. For the model in our example, each activation vector has 768 dimensions. An SAE is a neural network with one hidden layer that is trained to reconstruct these activations (Fig. 2C). The hidden layer of the SAE is larger than the activation vectors. In our example, we trained an SAE with 6144 hidden neurons. Sparsity is enforced on the hidden layer of the SAE so that only a small number of neurons in the hidden layer can fire at once to reconstruct the activations for a given token. In our case, we trained a k-sparse autoencoder [10] with k=8, meaning that only 8 of the 6144 hidden neurons can fire at the same time. The SAE is trained to minimize the error between the input activations and the output reconstructed activations.

To analyze the trained SAE, we again feed instances through the LM to generate token activations (Fig. 2D). We then feed those token activations into the SAE and observe the activations of the SAE's hidden-layer neurons. In our example, for each token that is input to the LM, 8 of the SAE's hidden-layer neurons will have positive activation values. We refer to the SAE's hidden-layer neurons as features. For each feature in the SAE, we track which instances contain tokens that cause the feature to activate.

We then generate visualizations for each feature to help us understand when it activates and how it relates to the model's predictions and errors (Fig. 2E). Ideally, each SAE feature represents a single, human-understandable concept. The concept that a feature represents can be determined by analyzing example tokens that activate the feature and their surrounding contexts. A good feature will primarily activate in contexts that are semantically similar. A poor feature may activate on tokens relating to multiple distinct concepts or not have a clear pattern in its activations. The goal of SAEfarer is to guide the user's exploration of the SAE features and how they relate to the classification model's behavior. To help identify interesting features to explore, the last step in our approach is to rank the features based on a variety of metrics calculated from the model's predictions and errors on the instances that activate them (Fig. 2F).

[1] https://huggingface.co/Kyle1668/ag-news-19200-bert-base-uncased

Table 1: Example SAE feature ranking metrics.

Pred. label	True label	Ranking Metric
World	*World*	% of instances that activate the feature that are correctly classified as *World*.
Business	*Sci/Tech*	% ... that are predicted as *Business*, but are *Sci/Tech*.
Sports	`Any`	% ... that are predicted to be *Sports*.
`Any`	*World*	% ... that have a ground-truth label of *World*.
Business	`Different`	% ... that are incorrectly predicted to be *Business*.
`Different`	*Sci/Tech*	% ... that have a ground-truth label of *Sci/Tech* and are incorrectly classified.
`Any`	`Different`	% ... that are incorrectly classified.

4 Feature Ranking

One challenge with using SAEs to analyze the behavior of models is that they contain too many features to be able to exhaustively explore all of them. It is necessary to identify and prioritize the most interesting features. For text classification, we believe that it is particularly relevant to identify features that have strong relationships with the model's predictions and errors. We propose an approach for ranking the features based on confusion matrices.

Each feature in the SAE is activated by a subset of instances in the dataset. We consider an instance to activate a feature if at least one token in the instance activates the feature. For each feature, we compute a confusion matrix for the instances that activate it. We then rank the features based on metrics that we calculate from values in the confusion matrices, which we summarize in Tab. 1.

For ranking the features, the user selects a value for the predicted class label and the true class label. For example, if the user selects the predicted label to be *Business* and the true label to be *Sci/Tech*, then this ranks the features based on the percentage of instances that activate the feature where the model predicts *Business*, but the true label is *Sci/Tech*. This would identify features that have the strongest relationship with the model making that specific error.

In addition to selecting a specific class for the predicted or true label, the user can also select from two wildcard values. The first wildcard is `Any`, which represents any class. For example, if the user selects *Sports* for the predicted label and `Any` for the true label, then this would rank the features based on the percentage of instances that activate the feature where the model predicts *Sports*. The second wildcard is `Different`, which represents any class except the other class that is selected. For example, if the user selects *Business* as the predicted label and `Different` as the true label, then `Different` represents *World*, *Sports*, and *Sci/Tech*. This configuration would rank the features based on the percentage of instances that activate the feature where the model incorrectly predicts the label to be *Business*. Finally, selecting `Any` as the predicted label and `Different` as the true label (or vice versa) would rank the features based on the percentage of instances that activate the feature where the model is incorrect.

5 User Interface

SAEfarer is a tool for exploring the concepts learned by text classification LMs. The primary goal is to enable the user to analyze the relationships between the features and the model's predictions and errors. Our aim is to help users identify features of interest, understand the concepts that those features represent, and analyze the relationship that those features have with the model's behavior. To do this, we integrated the feature rankings into an interactive widget for Jupyter notebooks [19, 20]. We organized the user interface into three tabs: Overview, Feature Detail, and Feature Table.

5.1 Overview

The Overview tab provides a high-level summary of the SAE and the model. A series of tables provides information about the dataset, model, and SAE, such as the number of tokens in the dataset, the model's error rate, and the size of the SAE. These tables are accompanied by two visualizations. First, a confusion matrix shows the performance of the classification model. Second, a histogram shows how often the features in the SAE activate. We define the activation rate of a feature as the percentage of instances in the dataset where the feature activates on at least one token.

5.2 Feature Detail

The Feature Detail tab shows information about a single feature. Figure 1 provides an example of a feature that activates on descriptions of statistics from a sporting event. At the top right (Fig. 1A), we display portions of instances that activated the feature. By default, we show instances that caused the feature to have the highest activation values. To visualize the snippets, we encode the activation value of the feature with a colored line beneath the token. Under the example activations, the user can enter their own piece of text and check how it causes the feature to activate (Fig. 1B).

In the bottom left (Fig. 1C), we show a confusion matrix of the instances that activated the feature. In this example, we can see that the feature has a strong relationship with the model accurately classifying *Sports* articles. In the top left (Fig. 1D), a marginal plot [1] shows the relationship between the feature's activation values and the model's predicted probabilities for each class. For this plot, we bin the instances that activate this feature by their activation value. If multiple tokens in the instance activate the feature, then we take the maximum value. For each bin of instances, we compute the model's mean predicted probability for each class. We then plot this as a heatmap. For the feature in Fig. 1, we see that the model assigns a high probability to the *Sports* class at all activation values. Above this, we provide a histogram that shows the number of instances in each activation value bin. The appendix contains additional example features.

5.3 Feature Table

The Feature Table tab enables the user to rank and filter the SAE features. Its purpose is to help the user identify features that are worth analyzing in the Feature Detail tab. Figure 3 shows a condensed screenshot of the Feature Table tab. At the top of the tab (Fig. 3A), there are controls that allow the user to adjust how they rank the features. The main area of the tab contains a table in which each row represents one feature. At the beginning of each row is a button that brings the user to the Feature Detail tab for the feature (Fig. 3B). Next, we show the model's error rate on instances that activate the feature (Fig. 3C) and the feature's activation rate (Fig. 3D). The remaining columns contain condensed versions of visualizations shown in the Feature Detail tab (Fig. 3E-G).

6 Expert Pilot Evaluation

We conducted an expert pilot evaluation to analyze how ML researchers use SAEfarer to explore a model. We were interested in assessing the usability of the tool and how it could be improved. Our evaluation was approved by the Northeastern University IRB (#23-04-35) and all participants reviewed a participant information sheet. Our participants were five Ph.D. students whose research relates to ML interpretability or explainability. They all indicated that they were familiar with the use of SAEs for interpreting LMs. We compensated each with a $50 Amazon gift card.

6.1 Protocol

We met with each participant through a video call for two hours. Each meeting began with a presentation where we gave an overview of the study, reviewed SAEs, and provided a tutorial on how to use

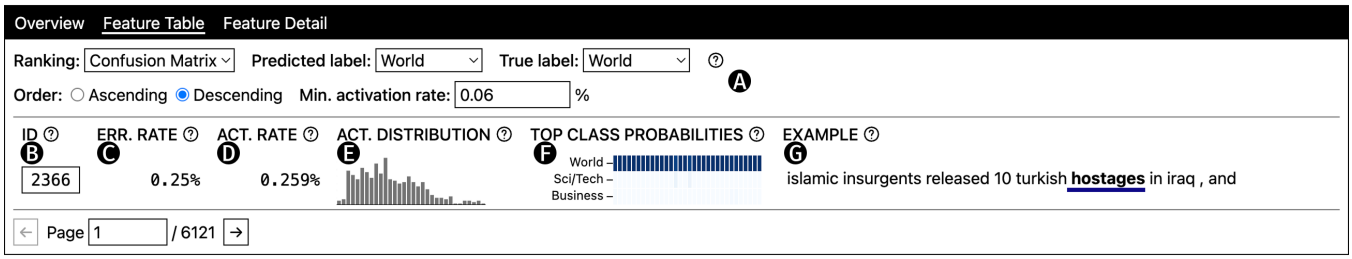

Figure 3: The Feature Table tab. A) Controls for ranking and filtering the features. B-D) The feature's index, error rate, and activation rate. E) A histogram of the feature's activation values. F) The model's probabilities for the top classes. G) The top example that activates the feature.

SAEfarer. We then gave the participant practice using SAEfarer. We provided them with a Jupyter notebook to analyze an SAE that we trained on an example model and dataset. We then asked them to answer a set of questions using the tool. These questions gave them practice with SAEfarer and allowed us to check that they could correctly use it and interpret the visualizations. We then gave the participant a few minutes to explore the model.

Next, we gave the participant a notebook that ran SAEfarer with an SAE trained on a model that classifies articles from the AG News dataset. We gave the participant 30 minutes to explore the behavior of this model. We instructed them to focus on analyzing the relationship between the SAE features and the model's predictions and errors. We also provided them with the following example questions. What features correlate with the model predicting the *Sports* label? What features correlate with the model making mistakes? Are there any surprising or unintuitive features? We chose an open-ended evaluation to allow participants to have different exploration strategies and to use the tool as they saw fit. After the 30 minutes, we reviewed aspects of their exploration with them before moving on to an interview where we elicited feedback.

6.2 Results

The meetings with the participants were recorded and transcribed by Zoom. We conducted thematic analysis [2] on the interview transcripts with the goal of understanding the strengths and weaknesses of SAEfarer and how it might be improved. Overall, participants liked SAEfarer, but pointed out several ways to improve it. Below we describe their feedback, organized by topic.

Confusion matrix rankings. Participants overall found the confusion matrix rankings to be helpful and clear to use. However, there is room for improvement to make it more obvious how to rank the features by error rate. Likewise, it was not entirely clear that ranking the features in ascending order by this metric would identify the features on whose activating instances the model had the highest accuracy. Allowing the user to adjust the rankings through the column headers could improve this.

Providing more context. In the example activations for a feature (Fig. 1A), we show the token that has the highest activation in the instance and a few tokens surrounding it. One common request that we received was to be able to look at the entire instance and not just the provided snippet. This would be useful in a few situations. For example, some participants came across features with high error rates, where, based on the example activations, they noted that it seemed reasonable for the model to be confused (Appendix A Fig. 5). Without seeing the entire instance, it was not possible to tell whether the model was actually wrong or the instances had incorrect ground-truth labels. In other cases, the token snippets did not show enough context for the participant to fully understand the concept represented by the feature. A feature's activations on a snippet of tokens can be influenced by other tokens in the instance that are not included in the snippet. For example, if the user copies a snippet of tokens from the examples into the "Test Feature" input (Fig. 1B), the activations on their tokens may not match the example, making it difficult to reproduce the behavior. This also highlights a need for better ways for users to evaluate their understanding of what a feature represents and when it activates.

Analyzing multiple features. SAEfarer's focus is on analyzing features independently. The participants expressed interest in analyzing multiple related features at the same time. For example, most participants were interested in seeing all the features that activate on a given piece of text, such as an existing instance or their own example. P1 was curious about finding groups of related features and analyzing when they did and did not activate together. P3 was interested in exploring features that were often activated together. P5 wanted to extend that idea and identify groups of features that activate together and correlate with the model's predictions and errors. P1, P2, and P3 were all interested in being able to search for features based on the tokens they tend to activate on.

SAE quality. Some participants raised concerns about the quality of the SAEs they were analyzing. This was in part due to features that were not interpretable or whose behavior was difficult to replicate with test cases. For example, P2 stated that "it's a nice tool, for sure, but it will be limited by the quality of the SAEs", which they felt were not very good. To improve our SAEs and reduce the number of features that lack clear interpretations, we could more robustly experiment with the size and sparsity of the SAEs. We could also use more recent advancements in SAE architecture and training [25, 4, 5, 8]. Additionally, we can make improvements to the SAE analysis process, such as ignoring padding tokens.

7 Limitations and Future Work

There are several limitations with our pilot evaluation, such as having only five participants who were all Ph.D. students. Although they had relevant background knowledge, their approaches to exploring the model may not reflect those of professional ML practitioners who work with text classification models. We did not assess how practitioners would use the tool over extended periods of time to analyze their own datasets and models. We also did not evaluate SAEfarer with wider audiences that lack familiarity with SAEs.

SAEfarer's feature rankings are calculated based on the confusion matrices of the instances that cause each feature to activate. One limitation with this is that the rankings only consider whether or not a feature activates for a given instance, but not the actual activation value. A feature's low activations tend to be noisier and less important than its high activations [3, 26]. We may want to create additional rankings that give more weight to higher-value activations rather than treating all activations as equivalent.

Goyal et al. [12] note that one limitation with approaches like TCAV is that they identify concepts that are correlated with the model's predictions, but not necessarily causal. Our feature rankings suffer from a similar limitation in that they only indicate if a feature's activations are correlated with the model's predictions. They do not indicate whether the concept has a causal impact on the model's predictions.

Supplemental Materials

Code and example notebooks for SAEfarer are available at github.com/DanielKerrigan/saefarer. Pilot evaluation material is available at github.com/DanielKerrigan/saefarer-user-study. We also include a demo video and appendices that provide screenshots of the UI, feature examples, and System Usability Scale results from the pilot evaluation.

References

[1] D. W. Apley and J. Zhu. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, Sept. 2020. doi: 10.1111/rssb.12377 3

[2] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, Jan. 2006. doi: 10.1191/1478088706qp063oa 4

[3] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. 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. 1, 2, 4

[4] B. Bussmann, P. Leask, and N. Nanda. BatchTopK sparse autoencoders, 2024. doi: 10.48550/arXiv.2412.06410 4

[5] B. Bussmann, N. Nabeshima, A. Karvonen, and N. Nanda. Learning Multi-Level Features with Matryoshka Sparse Autoencoders, 2025. doi: 10.48550/arXiv.2503.17547 4

[6] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models, 2023. doi: 10.48550/arXiv.2309.08600 1, 2

[7] Decode Research. SAE Dashboard. https://github.com/jbloomAus/SAEDashboard, 2024. 2

[8] T. Fel, E. S. Lubana, J. S. Prince, M. Kowal, V. Boutin, I. Papadimitriou, B. Wang, M. Wattenberg, D. Ba, and T. Konkle. Archetypal SAE: Adaptive and Stable Dictionary Learning for Concept Extraction in Large Vision Models, 2025. doi: 10.48550/arXiv.2502.12892 4

[9] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001. doi: 10.1214/aos/1013203451 1

[10] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu. Scaling and evaluating sparse autoencoders, 2024. doi: 10.48550/arXiv.2406.04093 2

[11] A. Ghorbani, J. Wexler, J. Zou, and B. Kim. Towards Automatic Concept-based Explanations. In *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. 1, 2

[12] Y. Goyal, A. Feder, U. Shalit, and B. Kim. Explaining Classifiers with Causal Concept Effect (CaCE), 2020. doi: 10.48550/arXiv.1907.07165 4

[13] A. Gulli. AG's corpus of news articles. 2

[14] J. Huang, A. Mishra, B. C. Kwon, and C. Bryan. ConceptExplainer: Interactive Explanation for Deep Neural Networks from a Concept Perspective. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):831–841, 2023. doi: 10.1109/TVCG.2022.3209384 1, 2

[15] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In J. Dy and A. Krause, eds., *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 2668–2677. PMLR, 10–15 Jul 2018. 1, 2

[16] T. Lieberum, S. Rajamanoharan, A. Conmy, L. Smith, N. Sonnerat, V. Varma, J. Kramár, A. Dragan, R. Shah, and N. Nanda. Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2, 2024. doi: 10.48550/arXiv.2408.05147 2

[17] J. Lin. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. Software available from neuronpedia.org. 1, 2

[18] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 4768–4777. Curran Associates Inc., Red Hook, NY, USA, 2017. 1

[19] T. Manz, N. Abdennur, and N. Gehlenborg. anywidget: reusable widgets for interactive analysis and visualization in computational notebooks. *Journal of Open Source Software*, 9(102):6939, 2024. doi: 10.21105/joss.06939 3

[20] T. Manz, N. Gehlenborg, and N. Abdennur. Any notebook served: authoring and sharing reusable interactive widgets. In *Proceedings of the 23rd Python in Science Conference*, July 2024. doi: 10.25080/NRPV2311 3

[21] C. McDougall. SAE Visualizer. https://github.com/callummcdougall/sae_vis, 2024. 2

[22] C. McDougall, A. Conmy, J. Kramár, T. Lieberum, S. Rajamanoharan, and N. Nanda. Gemma Scope 2 - Technical Paper. 2

[23] R. Movva, K. Peng, N. Garg, J. Kleinberg, and E. Pierson. Sparse autoencoders for hypothesis generation. In A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu, eds., *Proceedings of the 42nd International Conference on Machine Learning*, vol. 267 of *Proceedings of Machine Learning Research*, pp. 44997–45023. PMLR, 13–19 Jul 2025. 2

[24] H. Park, N. Das, R. Duggal, A. P. Wright, O. Shaikh, F. Hohman, and D. H. P. Chau. NeuroCartography: Scalable Automatic Visual Summarization of Concepts in Deep Neural Networks. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):813–823, 2022. doi: 10.1109/TVCG.2021.3114858 1, 2

[25] S. Rajamanoharan, T. Lieberum, N. Sonnerat, A. Conmy, V. Varma, J. Kramár, and N. Nanda. Jumping Ahead: Improving Reconstruction Fidelity with JumpReLU Sparse Autoencoders, 2024. doi: 10.48550/arXiv.2407.14435 4

[26] 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, 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. 2, 4

[27] X. Yan, S. Liu, K. Thopalli, and B. W. Phillips. Visual exploration of feature relationships in sparse autoencoders with curated concepts. In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. 2

[28] 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 H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, pp. 20554–20565. Curran Associates, Inc., 2020. 2

[29] X. Zhang, J. H. Piazentin Ono, W. He, L. Gou, M. Sachan, K.-L. Ma, and L. Ren. Slicing, Chatting, and Refining: A Concept-Based Approach for Machine Learning Model Validation with ConceptSlicer. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*, IUI '24, p. 274–287. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3640543.3645163 1, 2

[30] Z. Zhao, P. Xu, C. Scheidegger, and L. Ren. Human-in-the-loop Extraction of Interpretable Concepts in Deep Learning Models . *IEEE Transactions on Visualization and Computer Graphics*, 28(1):780–790, Jan. 2022. doi: 10.1109/TVCG.2021.3114837 1, 2