Unveiling the Black Box: Causal Inference and Feature Analysis in Fine-Tuned Language Models Using Sparse Autoencoders

Sean Sica, Rini Gupta

4 August 2024

Abstract

3 to 5 sentences Concisely describe your problem, how you solved it, and what you found

1 Introduction

The field of artificial intelligence has witnessed remarkable advancements, with language models like GPT-3 and PaLM demonstrating human-level proficiency in English communication. However, a significant challenge persists: despite creating these powerful systems, we lack a comprehensive understanding of their internal mechanisms. This gap in knowledge not only limits our ability to replicate such systems but also raises concerns about their interpretability and controllability.

Mechanistic interpretability (MI) has emerged as a promising approach to address this challenge. Recent work, particularly by Anthropic, has shown that sparse dictionary learning algorithms, specifically sparse autoencoder (SAE) model architectures, offer a potential path forward in making neural networks more interpretable [2]. Their demonstration of controlling language model outputs using identified features, such as the "Golden Gate Bridge" demo [8], has inspired further research in this direction.

Our work aims to contribute to this growing field by:

1. Proposing a novel and effective process for evaluating large quantities of SAE features.

2. Testing the hypothesis that fine-tuned models manifest higher quantities of relevant features and sharper, more cohesive features.

3. Increasing awareness of mechanistic interpretability through accessible feature analysis and causal intervention techniques.

Specifically, we hypothesize that when we fine-tune a model, compared to the base model, we will observe both a higher quantity of features relevant to the fine-tuning training dataset and higher quality features. In testing this hypothesis, we also investigate tangential questions such as the relationship between an SAE's training dataset and the resultant features, whether fine-tuning models on specific datasets enhances their

interpretability, and if fine-tuned models are easier to steer than their foundational counterparts. These insights are particularly relevant given the widespread use of fine-tuned models in the tech industry.

Our approach also addresses a key criticism of current MI work: the potential misinterpretation of SAE features. By employing causal inference techniques, we aim to provide stronger evidence that these features not only are interpretable but also demonstrably represent what we claim they do.

The primary contributions of this paper are:

1. Novel research evaluating sparse autoencoders with respect to fine-tuned models, using controlled experiments to better understand fundamental properties of SAE features.

2. Reusable code for practitioners and researchers to easily extract and analyze features from trained sparse autoencoders.

3. A framework for applying causal interventions to steer model behavior conditioned on features of interest, replicating and extending Anthropic's demonstration.

In the following sections, we present our methodology, results, and analysis, providing evidence to substantiate our claims and contribute to the growing body of knowledge in mechanistic interpretability of language models. Through our work, we aim to not only validate our hypothesis regarding the impact of fine-tuning on feature quantity and quality but also to provide a framework for future research in this critical area of AI interpretability.

2 Background

The foundation of our research lies in Anthropic's work on superposition, polysemanticity, and toy models in neural networks [2, 8, 5, 4]. Their research asserts that neurons in neural networks are polysemantic, responding to mixtures of seemingly unrelated inputs. When inputs are fed into a transformer model, a substantial subset of neurons typically show non-negligible activation. This phenomenon is analogous to observing widespread brain activation in response to specific stimuli during neurological studies.

Anthropic hypothesizes that this behavior results from superposition, where a neural network represents more independent "features" of the data than it has neurons by assigning each feature its own linear combination of neurons. This suggests that we may be able to recover the network's features by finding a set of directions in activation space such that each activation vector can be reconstructed from a sparse linear combination of these directions. This approach is equivalent to the well-known problem of sparse dictionary learning [3]. Through experiments with toy models, Anthropic demonstrated that sparse dictionary learning can be employed to causally reduce polysemanticity, making inference activations far more sparse and enabling more effective causal inference for neuron labeling.

Scaling the neural network to have more neurons than features is not a viable solution due to incidental polysemanticity. Lecomte et al. [7] suggest that polysemanticity can manifest for multiple reasons, including regularization and neural noise. They propose that incidental polysemanticity might be mitigated by adjusting the learning trajectory without necessarily altering the neural architecture.

In a different approach, researchers at OpenAI have proposed LLM-driven solutions to tackle the interpretability issue [1]. Their method uses prompt engineering to interpret individual neurons. While we approach this method with some skepticism regarding its practicality and reliability, it provides inspiration for our research. We aim to leverage prompt engineering to aid in analyzing and labeling sparse features extracted from sparse autoencoders.

Sparse autoencoders (SAEs) have emerged as a powerful tool for addressing the challenges of superposition and polysemanticity in neural networks. An SAE is a type of neural network designed to learn a sparse representation of input data, typically with more hidden units than input units. The key insight behind SAEs is that by enforcing sparsity in the hidden layer, we can encourage the network to learn more interpretable and disentangled features. Conceptually, the encoder component of an SAE detects features by finding optimal directions in the activation space to project onto, minimizing interference with other similar features. The decoder, on the other hand, attempts to represent these features, approximating their "true" directions regardless of interference. This division of labor allows SAEs to potentially recover individual features that are superposed in the original network. By applying an L1 regularization penalty during training, SAEs are incentivized to activate only a small subset of hidden units for any given input, leading to more focused and interpretable representations. This approach has shown promise in decomposing the complex, polysemantic representations found in large language models into more semantically coherent and manipulable features. [2]

Our research seeks to synthesize Anthropic's approach of decomposing neuron activations into sparse, interpretable features with OpenAI's neuron-centric automated interpretability solution. By combining these methodologies, we aim to enhance the interpretability of fine-tuned language models and provide a more robust framework for understanding their internal mechanisms.

Our research seeks to synthesize Anthropic's approach of decomposing neuron activations into sparse, interpretable features with OpenAI's neuron-centric automated interpretability solution. By combining these methodologies, we aim to enhance the interpretability of fine-tuned language models and provide a more robust framework for understanding their internal mechanisms.

To verify that our learned features represent a semantically meaningful decomposition of the activation space, we employ several techniques. First, we demonstrate that our features are, on average, more interpretable than neurons and other matrix decomposition techniques, as measured by coherence scores. We then confirm these findings through both a causal intervention "smoke test" and a human evaluation. These methods collectively provide strong evidence for the semantic significance of our extracted features.

3 Methods

We designed an experiment involving a control group and a treatment group to isolate the effects of fine-tuning on a language model's internal representations. The treatment effect is the application of fine-tuned data to the control/baseline model. Our aim is to isolate the causal effects of fine-tuning on the resultant features extracted from the model's sparse autoencoder, allowing us to observe how features change with respect to model training. Specifically, we extract features and analyze their sharpness using an empirical coherence score and verify their supposed identities using a rigorous causal

intervention smoke test.

3.1 Experimental Design

Our experimental approach is designed to use causal inference to directly observe the impact of the fine-tuning process on the model's learned features while holding all other variables constant. The key components of our experimental design are:

1. Control Group: A baseline GPT-2 model with a pretrained sparse autoencoder (SAE).
2. Treatment Group: A fine-tuned GPT-2 model (based on medical patient transcripts) with a custom-trained SAE.
3. Feature Extraction: A process to extract interpretable features from both SAEs.
4. Feature Analysis: Methods to assess feature sharpness and verify feature identities.

The causal intervention smoke test is a key component of our methodology. It involves applying a steering vector to the transformer's residual stream using a hook point positioned at the "pre" stage of the residual application. This steering vector allows us to control the activation strength of a candidate feature using a coefficient multiplier and a temperature value (ranging from 0 to 1.0).

For each feature that passes our coherence and human evaluation criteria, we conduct an experiment wherein next token prediction/inference is triggered in response to a relevant prompt. We determine the feature to be a match (i.e., the feature is what we suppose it to be based on its top activating tokens) if the steering vector causes the model output to drastically shift in the direction of the candidate feature label.

For example, if we identify a feature that appears to represent (or "fire for") words like "doctor" and "nurse", we would expect a steering vector applied to this feature to overrepresent those words in its next token predictions. This approach allows us to verify the semantic meaning of the extracted features and assess how fine-tuning affects the model's internal representations.

3.2 Control Group: Baseline Model and Pretrained SAE

The control group consists of our baseline model, gpt2-small, with a pretrained sparse autoencoder (SAE) model trained on the token activations extracted from the residual stream of the transformer model. Key specifications of the baseline model are as follows:

Our pretrained sparse autoencoder was developed by Joseph Bloom and is available through the SAELens library [6]. The SAE consists of three layers: an encoder, MLP, and decoder. It accepts batches of internal activations from the residual stream of our transformer model. Notably, we apply a bias on the input layer for performance purposes, as suggested by Bricken et al. [2]. The SAE is trained using the Adam optimizer to reconstruct the MLP activations of the gpt2-small model, with an MSE loss and an L1 penalty to encourage feature sparsity.

The architecture of the SAE is defined by the dimensions of its weights and biases. The encoder weight matrix (W_{enc}) has dimensions (768, 24576), transforming the input from the model's hidden state dimension (768) to the SAE's expanded feature space (24576). Conversely, the decoder weight matrix (W_{dec}) has dimensions (24576, 768), projecting

Parameter	Value
Model Name	gpt2-small
Number of Parameters	85M
Number of Layers	12
Model Dimension (d_model)	768
Number of Attention Heads	12
Activation Function	GELU
Context Window Size	1024
Vocabulary Size	50257
Dimension per Head	64
MLP Dimension	3072

Table 1: Specifications of the gpt2-small model

back to the original hidden state dimension. The encoder and decoder bias vectors (b_{enc} and b_{dec}) have 24576 and 768 dimensions respectively, corresponding to their respective output spaces.

This architecture implies a significant expansion in the feature space, with the SAE operating on a space more than 32 times larger than the original model's hidden state ($24576/768 \approx 32$). This expansion allows the SAE to potentially capture and isolate a much richer set of features than are directly represented in the original model's activations.

The SAE was trained on activations extracted from multiple layers of the gpt2-small model. Specifically, it processes the residual stream activations (hook_resid_pre) from all 12 layers of the model, as well as the post-residual activations of the final layer (hook_resid_post). For each layer, the SAE dimension (d_{sae}) is consistently 24576, and a context size of 128 tokens was used. The training data was sourced from the "Skylion007/openwebtext" dataset, and no normalization was applied to the activations during training.

This comprehensive approach to extracting and processing activations from multiple layers allows the SAE to capture features that may be represented differently or at different levels of abstraction throughout the depth of the transformer model.

3.3 Treatment Group: Fine-tuned Model and Custom SAE

Our treatment group consisted of a fine-tuned variant of the control group model. The transformer architecture remains the same gpt2-small model, but we fine-tuned it on a specialized dataset related to medical patient transcripts. The sparse autoencoder for this group was custom trained on the activation vectors extracted from the transformer's residual stream, mirroring the approach used in the control model.

The Sparse Autoencoder (SAE) for the treatment group was trained using a carefully configured SAE training pipeline. The training process was designed to run for 30,000 steps with a batch size of 1024 tokens, resulting in a total of 30,720,000 training tokens. The learning rate scheduler was set to maintain a constant learning rate of 5e-5 throughout the training process, with Adam optimizer parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The SAE was configured to work with GPT-2's architecture, specifically targeting the output of the first MLP layer (blocks.0.hook_mlp_out) (TODO VERIFY THIS) with an input dimension of 768. The autoencoder employed an expansion factor of 16, resulting in a hidden layer size of 12,288 (768 * 16). This expansion allows the SAE to potentially

capture a richer set of features than those directly represented in the original model's activations.

To encourage sparsity in the learned features, an L1 regularization term was applied with a coefficient of 5. The L1 penalty was gradually introduced over the first 1,500 training steps (total_training_steps // 20) to allow the model to initially learn without sparsity constraints.

The training data was sourced from the "monology/pile-uncopyrighted" dataset, using a streaming approach to handle large data volumes efficiently. Each training example used a context size of 512 tokens, allowing the SAE to capture dependencies over relatively long sequences.

Several architectural choices were made to optimize the SAE's performance:

1. The decoder bias was initialized to zeros.
2. The decoder was not normalized, but the sparsity penalty was scaled by the decoder norm.
3. A heuristic initialization was used for the decoder, and the encoder was initially set as the transpose of the decoder.
4. Input activations were normalized using an "expected average only in" approach.

To address the potential issue of dead features, the training process included a resampling protocol. This protocol monitored feature activations over a window of 1000 steps, identifying and potentially resampling features that remained inactive (activation below 1e-4) throughout this period.

The training progress was logged to Weights & Biases (wandb) every 30 steps, with a more comprehensive evaluation performed every 600 steps (20 * 30). This allowed for detailed tracking of the training process and the evolving characteristics of the learned features.

By using this custom-trained SAE on our fine-tuned GPT-2 model, we aim to capture and isolate features that are specifically relevant to the patient transcript domain, potentially revealing insights into how the fine-tuning process affects the model's internal representations.

3.4 Feature Extraction and Analysis

To extract interpretable features from our sparse autoencoders, we employ a multi-step process:

1. Projection of SAE Decoder Weights: We project the SAE's decoder weights (W_{dec}) onto the unembedding matrix (W_U) of the transformer model. This projection is computed as:

$$Decoder\ Projection\ On\ W_U = W_{dec} \otimes W_U$$

   where $\otimes$ denotes matrix multiplication.

2. Feature-to-Vocabulary Mapping: This projection allows us to map the SAE's learned features directly onto the model's vocabulary space, effectively representing how each SAE feature influences the prediction of each token in the model's vocabulary.

3. Top Activated Words Identification: For each feature, we identify the top activated words based on the projection. This helps us infer the semantic meaning or function of each SAE feature in the context of the language model's vocabulary and task.

4. Sparsity Analysis: We incorporate sparsity measurements of the SAE features into our analysis, allowing us to assess how focused or distributed each feature's influence is across the model's vocabulary space.

3.5 Measuring Success

We employ several methods to measure the success of our approach:

1. Empirical Coherence Score: We use this score to analyze the sharpness of extracted features.

2. Causal Intervention Smoke Test: As described in the Experimental Design section, this test verifies the semantic meaning of extracted features.

3. Human Evaluation: We employ keyword matching to identify features containing relevant tokens/words in their top 10 activated words. This is followed by a human evaluation to verify the semantic coherence of the identified features.

4. Comparative Analysis: We compare the features extracted from the control group (baseline model) with those from the treatment group (fine-tuned model) to assess the impact of fine-tuning on feature interpretability and relevance.

This comprehensive approach allows us to not only extract meaningful, interpretable features from the complex internal representations of the transformer models but also to quantitatively and qualitatively assess the impact of fine-tuning on these features. By combining projection-based feature interpretation, sparsity analysis, and rigorous testing methods, we aim to provide a robust framework for understanding how fine-tuning affects a language model's internal representations.

4 Results and Discussion

Present your results using tables, plots, and figures as appropriate. Provide a detailed analysis of your findings, comparing them to baselines and relevant literature.

4.1 Subsection 1

Details of your first set of results or analysis.

4.2 Subsection 2

Details of your second set of results or analysis.

5 Conclusion

Summarize your main findings and their implications. Discuss any limitations of your work and potential future directions.

References

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

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

[3] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

[4] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

[5] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[6] David Channin Joseph Bloom. Saelens training, 2024. https://jbloomaus.github.io/SAELens/index.html.

[7] Victor Lecomte, Kushal Thaman, Rylan Schaeffer, Naomi Bashkansky, Trevor Chow, and Sanmi Koyejo. What causes polysemanticity? an alternative origin story of mixed selectivity from incidental causes, 2024.

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

A Appendix

Include any supplementary material, additional data, or extended proofs here.