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

Rini Gupta[1] and Sean Sica[1]

[1] *UC Berkeley*

4 August 2024

Abstract

This study examines the impact of fine-tuning on language model interpretability using sparse autoencoders (SAEs) and causal interventions. We hypothesized that fine-tuned models would have more relevant and sharper features than baseline models, but our results showed the baseline GPT-2 model outperformed the fine-tuned version in both feature interpretability and responsiveness to causal interventions. Specifically, the baseline model had higher average coherence scores (0.5438 vs. 0.3511) and greater vector steering receptiveness (99% vs. 83.7%). These findings suggest that fine-tuning may lead to more specialized but less coherent features. Our research advances mechanistic interpretability by introducing a novel SAE evaluation process and highlighting the nuanced effects of fine-tuning, with implications for creating more transparent AI systems.

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 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 SAE model architectures, offer a potential path forward in making large language models 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 compared to a pre-trained base model.

We hypothesize that fine-tuned models, compared to the base model, will exhibit

both a higher quantity of relevant features and higher quality features. In testing this hypothesis, we investigate the impact of fine-tuning on interpretability and the ease of steering fine-tuned models. 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 validate our hypothesis regarding the impact of fine-tuning on feature quantity and quality and provide a framework for future research in this critical area of AI interpretability.

2 Background

Our research builds upon Anthropic's work on superposition and polysemanticity in neural networks [2, 8, 5, 4]. Their studies show that neurons in neural networks are polysemantic, responding to mixtures of seemingly unrelated inputs due to superposition, where a network represents more independent "features" than it has neurons. This phenomenon is analogous to widespread brain activation in response to specific stimuli during neurological studies. Scaling the neural network to have more neurons than features is not viable due to incidental polysemanticity, which Lecomte et al. [7] suggest can manifest from factors like regularization and neural noise, proposing that adjusting the learning trajectory might mitigate this issue without altering the neural architecture.

To address this, SAEs have emerged as a powerful tool. SAEs are neural networks designed to learn sparse representations of input data, typically with more hidden units than input units. By enforcing sparsity in the hidden layer, SAEs encourage the network to learn more interpretable and disentangled features, akin to sparse dictionary learning [3]. The encoder detects features by finding optimal directions in the activation space, while the decoder represents these features, approximating their "true" directions regardless of interference. This division allows SAEs to potentially recover individual features superposed in the original network [2].

Researchers at OpenAI have proposed LLM-driven solutions using prompt engineering to interpret individual neurons [1]. While we are skeptical about its practicality and reliability, this method inspires our research. We aim to leverage prompt engineering to analyze and label sparse features from sparse autoencoders. By synthesizing Anthropic's approach of decomposing neuron activations into sparse, interpretable features with OpenAI's neuron-centric in-

terpretability solution, we seek to enhance the interpretability of fine-tuned language models and provide a robust framework for understanding their internal mechanisms. We verify our learned features' semantic significance through coherence scores, intervention tests, and human evaluation exercises, collectively providing strong evidence for their interpretability.

3 Methods

We designed an experiment involving a control and treatment model 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 intervention 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: A baseline GPT-2 model with a pretrained SAE.
2. Treatment: A GPT-2 model finedtuned on a dataset containing over 6,000 medical terms and their Wikipedia text with a custom-trained SAE.[1]
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 intervention experiment is a key component of our methodology. It involves applying a steering vector to the transformer using a hook point positioned at the "pre" stage of the residual stream. 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 Baseline Model and Pretrained SAE

The control consists of our baseline model, gpt2-small, with a pretrained SAE model[2] trained on the token activations extracted from the residual stream of the transformer model. Key specifications of the baseline model are provided in the Appendix[3].

The SAE consists of three layers: an encoder, multi-layer perceptron (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.

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 the eighth attention block of the transformer. Specifically, it processes the activations from the residual stream. The training data was sourced from the "Skylion007/openwebtext" dataset, and no normalization was applied to the activations during training.

3.3 Fine-tuned Model and Custom SAE

Our treatment consisted of a fine-tuned variant of the control model. The transformer architecture remains the same gpt2-small model, but it is fine-tuned on a specialized dataset related to Medical Wiki Text. 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 training was orchestrated using a carefully configured SAE training pipeline.[3] 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.

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. Coherence Score: For each feature, we use a pre-trained word embedding model (GloVe) to calculate the average cosine similarity between all pairs of top activating tokens for each feature. A higher average similarity indicates greater semantic coherence.

3.5 Measuring Success

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

1. *Empirical Coherence Score*: Analyzes the sharpness of extracted features.
2. *Causal Intervention Test*: Verifies the semantic meaning of extracted features as described in the Experimental Design section.
3. *Human Evaluation*: Uses keyword matching to identify "medical" features (at least two medical keyword matches) followed by thorough human verification to ensure semantic coherence and avoid false positives (such as synonym/subword misinterpretation).
4. *Comparative Analysis*: Compares features from the baseline model and the fine-tuned model to assess the impact of fine-tuning on feature interpretability and relevance.

This comprehensive approach allows us to extract meaningful features from transformer models and assess the impact of fine-tuning quantitatively and qualitatively. Combining projection-based feature interpretation and rigorous testing provides a robust framework for understanding how fine-tuning affects a language model's internal representations.

4 Results

Our experiment evaluated features from two sparse autoencoders using keyword matching, human evaluation, coherence scoring, and vector steering intervention. Features of interest are those with at least two top activating tokens present in our reference medical vocabulary [4].

We assess feature sharpness using a novel "semantic coherence" score, which measures the semantic relatedness of a neuron's top-K activations. This score (0-1) evaluates pairwise comparisons between embeddings of top-K activated tokens, weighted by their activation scores, capturing the neuron's semantic consistency across multiple activations.

The semantic coherence score is defined as:

$$\text{Coherence Score} = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_i w_j \cdot \text{sim}(\vec{e_i}, \vec{e_j})}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_i w_j + \epsilon} \quad (1)$$

Where n is the number of valid tokens, w_i and w_j are activation scores, $\vec{e_i}$ and $\vec{e_j}$ are word embeddings, and $\text{sim}(\vec{e_i}, \vec{e_j})$ is the cosine similarity between embeddings. ϵ is a small constant to avoid division by zero.

4.1 Intervention Test Results

For the intervention experiment, we evaluate one feature at a time by attaching a steering vector that amplifies the corresponding neuron weights and generating sequences (up to 50 tokens). We then assess the output's cosine similarity to the feature's top-K tokens. Amplifying sparse feature weights should increase the representation of these tokens proportional to the steering vector coefficient. We interpret the steered output by comparing it to the model's normal (un-steered) output.

In our baseline model, we extracted 24,576 features from the eighth attention block's residual stream and identified 101 features of interest. These were filtered through keyword matching against a reference vocabulary and human evaluation. Of these, 100 (99%) produced steered output with higher semantic similarity than the un-

steered counterpart, indicating high receptivity to vector steering using a coefficient multiplier of 50 based on basic parameter optimization techniques. For the fine-tuned model, we extracted the same number of features and identified 111 relevant features. Of these, 93 (83.7%) produced steered output with higher similarity scores than the un-steered output.

Metric	Baseline	Fine-tuned
Total medical features	101 (0.39%)	111 (0.43%)
Passing intervention	100 (99%)	93 (84%)
Avg. steered similarity	0.6118	0.5165
Avg. baseline similarity	0.4634	0.4651

Table 1: Vector Steering Results Comparison

4.2 Coherence Scores

We contrast the features of the baseline model with those of the fine-tuned model by comparing their coherence scores. When comparing the entire feature set, we observe that the mean coherence scores of the baseline features (0.5438) are higher than those of the fine-tuned features (0.3511), with a difference of 0.1927.

To contextualize this difference, we normalized it with respect to the range of coherence scores. The baseline model's coherence scores range from 0.1130 to 0.7641, while the fine-tuned model's scores range from 0.0615 to 0.6793. The normalized mean difference is 0.3037, indicating that the difference in coherence scores is approximately 30.37

Metric	Mean	Median
Baseline Coherence	0.5438	0.5648
Finetuned Coherence	0.3511	0.3196
Difference	0.1927	0.2452

Table 2: Coherence Scores Comparison

The substantial difference in coherence scores suggests that fine-tuning has significantly altered the feature representations in the model. Lower coherence scores in the fine-tuned model might indicate that features have become more specialized or task-specific, potentially at the cost of general coherence. Further investigation into these changes and their impact on model performance is needed to understand the effects of fine-tuning on feature representations.

Figure 1 shows the distribution of medically-related terms among each feature's top-K activating tokens. Medically-related features are defined as those with at least 2 of their top 10 activating tokens matching our reference medical vocabulary. While the fine-tuned model has more medically-related features, the baseline model shows a broader distribution with higher numbers of medical term matches. Notably, the baseline model has features with 7 or more medical tokens in their top-K set, which is absent in the fine-tuned model. This aligns with the finding of lower mean coherence scores for medical features in the fine-tuned model, as coherence scores are influenced by the number of medical tokens in a feature's top activating set. This explains the observed distribution pattern and provides insight into the differing representations of medical knowledge between the models.

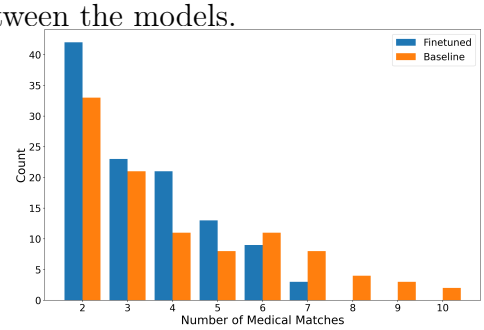

Figure 1: Distribution of medical term matches in top-K activating tokens for features in fine-tuned and baseline models

4.3 Word Analysis

Our analysis revealed significant differences in the features extracted by the sparse au-

toencoder (SAE) from the baseline and fine-tuned models. 1.1 times as many medically-related features were extracted from the fine-tuned model compared to the baseline model. Specifically, we identified 111 medical features in the fine-tuned model versus 101 in the baseline model.

The mean activation values for medically related features differed substantially between the two models. In the fine-tuned model, we observed a mean activation of $\mu = 0.794$ with a variance of $\sigma^2 = 0.018$, while the baseline model showed a mean activation of $\mu = 2.166$ with a variance of $\sigma^2 = 0.633$. These statistics indicate that the fine-tuned model's medical features have lower mean activation values and significantly reduced variance compared to the baseline model.

The activation distributions revealed notable differences: the fine-tuned model's activation values form a sharp, narrow normal curve, indicating more consistent activation of medical features. In contrast, the baseline model shows a broader normal distribution with more varied activation patterns. Comparative analysis highlights these differences: the fine-tuned model has 97.16% lower variance and 63.34% lower mean activation strength compared to the baseline model. These results suggest that fine-tuning has created a more specialized representation of medical knowledge, with consistently activated features, unlike the baseline model, which has fewer but more variable medical features.

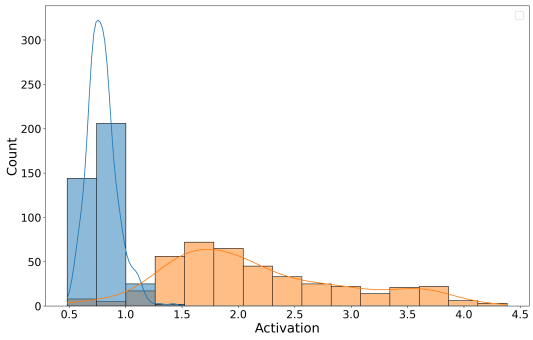

Figure 2: Distribution of Activation Values for Medical Words

4.4 Discussion

We also tested our approach on a custom fine-tuned GPT2 model, though we don't include those results here.[5] Like the other fine-tuned model, this model underperformed the baseline on the intervention test. We hypothesize that fine-tuning may make models less responsive to vector steering interventions. This could be because fine-tuning skews the data distribution, reducing the balance typically observed in pre-trained foundation models. The elevated mean activation score observed in the fine-tuned model supports this theory. It is possible that a fine-tuned model trained with more rigor may yield lower, more normally distributed activation scores. Regression analysis may be effective approach to further contextualizing the effect of activation score distributions on vector steering efficacy.

Notably, our coherence scoring algorithm is limited by the word embedding model selection. Each coherence score calculation involves computing the cosine similarity of the word embedding for each feature's top-K activating tokens. Our models were trained using a sub-word tokenizer, resulting in a fairly substantial vocabulary gap. Specifically, the features of interest (i.e., the features related to medical terms) extracted from the baseline model achieved a word embedding coverage of 84.51% (unique) and 89.99% Coverage (total). Unique coverage refers to the number of distinct words in the feature set, without considering how many times each word appears. Total coverage is total count of all words, including repetitions. Two contending word embedding models, "fasttext-wiki-news-subwords-300" and "word2vec-google-news-300", were evaluated as well. In both cases, the "glove-wiki-gigaword-100" model performs the best, despite having fewer dimensions (100) compared to the other models (300). A potential area for future research could be exploring ways to close this

coverage gap by using word embeddings trained specifically on the SAE's feature activation corpus.

5 Conclusion

We initially hypothesized that fine-tuned models would exhibit more and sharper, cohesive features compared to baseline models. Our results partially supported this, showing an increase in relevant features but lower coherence scores and less responsiveness to vector steering interventions. These findings suggest that while fine-tuning increases domain-specific features, it may lead to more complex, less interpretable representations. Future work could optimize fine-tuning to enhance feature interpretability and explore more rigorous vector steering interventions, crucial for AI alignment.

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

Custom SAE Training:

1. The training process was designed to run for 30,000 steps with a batch size of 1024 tokens, resulting in a total of 61,440,000 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 eighth MLP layer (blocks.8.hook_mlp_out) with an input dimension of 768.

2. 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.

3. 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.

4. 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.

5. The transformer comprises twelve layers, any of which could have been chosen to extract activations for training the SAE. Although we opted for the eighth layer, this choice was arbitrary, as any other layer would have sufficed.

6. 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.

7. The overall loss achieved was 211.71364, with MSE loss at 66.29448 and L1 loss at 29.08383.

8. The SAE demonstrated good reconstruction performance, with a CE loss score of 0.99386 (close to 1) and an explained variance of 0.89557, suggesting it captured a large portion of the variance in the input. The L2 ratio (out/in) of 0.79866 indicates that output activations were slightly smaller than input, as expected.

9. The learned features exhibited very sparse activations, with a mean log10 feature sparsity of -3.22845. Only 66 features were identified as "dead" (never activating), which is a relatively small number considering the total number of features.

Pretrained SAE Details:

1. Our pre-trained sparse autoencoder was developed by Joseph Bloom and is available through the SAELens library [6].

2. 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 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.

Fine-tuned GPT2 Model for Medical Domain:

- For our analysis, we utilized an off-the-shelf fine-tuned variant of GPT2-small called "Med_GPT2". This model, developed by Sharathhebbar24, was specifically fine-tuned on medical terminology to enhance its performance in the medical domain. The fine-tuning process used the "gamino/wiki_medical_terms" dataset, which contains over 6,000 medical terms along with their corresponding Wikipedia text.

- The Med_GPT2 model is publicly available and can be accessed at `https://huggingface.co/Sharathhebbar24/Med_GPT2`. We chose to incorporate this model into our study to enhance the medical domain specificity of our analysis, providing a more targeted approach to identifying and evaluating medical-related features in our sparse autoencoders.

Initial Fine-tuned GPT2 Model for Medical Domain:

- Our first iteration of the fine-tuned model was one we fine-tuned ourselves using LoRA (Low-Rank Adaptation). We fine-tuned GPT2 using 10,000 random samples from an open-source dataset available on HuggingFace (`https://huggingface.co/datasets/ruslanmv/ai-medical-chatbot`). However, the output was generally less coherent than baseline GPT2. We extracted 451 features from the same residual stream hook point as reported throughout the paper. Of these, only 307 achieved a coherence score better than their unsteered variants, resulting in a 68% pass rate. This contrasts with the 311 features extracted from the baseline model using the same methods, of which 307 (98%) passed the intervention test. Unsatisfied with these results and suspecting that our fine-tuning process may have interfered with the experiment, we opted to use a similar, but more performant, pre-fine-tuned model for our final analysis. Notably, the observation holds consistent for both fine-tuned models: the baseline model was more interpretable and more responsive to the steering vector intervention.

GPT-2 Small Specifications:

- Model Name: gpt2-small
- Number of Parameters: 85 million
- Number of Layers: 12
- Model Dimension (d_model): 768
- Number of Attention Heads: 12
- Activation Function: GELU
- Context Window Size: 1024 tokens
- Vocabulary Size: 50,257 tokens
- Dimension per Attention Head: 64

- MLP Dimension: 3072

Code Repository:

- All code used in this study, including Python notebooks, scripts for feature extraction, and analysis tools, is available on GitHub at: `https://github.com/seansica/sae-vector-steering`
- The repository contains detailed documentation and instructions for reproducing our results and extending our work.