Universiteit Leiden

Master Computer Science

Is Chain-of-Thought Faithfully Reasoning in Large Language Models?

Name:	Xi Chen
Student ID:	3863522
Date:	10/06/2025
Specialisation:	Data Science
1st supervisor:	Prof. dr. Aske Plaat
2nd supervisor:	Dr. Niki van Stein

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Large language models (LLMs) have recently achieved strong performance on complex reasoning tasks, but their internal reasoning processes remain opaque and lack interpretability. To improve both reasoning performance and interpretability, researchers have proposed the Chain-of-Thought (CoT) prompting method, which guides models to generate intermediate reasoning steps, thereby improving performance on multi-step reasoning tasks. However, there is ongoing debate about whether these intermediate chains produced by CoT prompts truly reflect the model's internal decision-making process—that is, whether they are faithful. Based on this, our study poses a central question: does CoT prompting increase the faithfulness of the internal reasoning process in LLMs?

To investigate this, we adopt a mechanistic interpretability approach, combining sparse autoencoders (SAEs) with activation patching techniques. We design comparative experiments on the GSM8K math question-answering dataset to analyze model behavior under both CoT and NoCoT prompting conditions. Our experiments cover models of different sizes, including the large Pythia-2.8B and the small Pythia-70M, and assess the impact of CoT prompts from three perspectives: semantic feature consistency, causal influence on output, and activation sparsity.

Results show that for large models, CoT prompts significantly enhance internal consistency, increase causal influence on outputs, and produce more structured, sparse activations. These effects are weaker and less stable in smaller models. This indicates that CoT can induce more interpretable internal structures in high-capacity LLMs, validating its role as a structured prompting method—though its ability to improve reasoning faithfulness is constrained by model capacity.

Contents

1 Introduction **1**
1.1 Topic . . . 1
1.2 Problem Statement . . . 2
1.3 Method . . . 2
1.4 Overview . . . 3

2 Related Work **4**
2.1 Background . . . 4
2.2 Reasoning and CoT . . . 6
2.3 Faithfulness and Causal Interpretability . . . 7
2.4 SAE and Interpretation . . . 9
2.5 Chapter Summary . . . 10

3 Methodology **12**
3.1 Sparse Autoencoding . . . 13
3.2 Activation Patching . . . 15
3.3 Activation Sparsity . . . 20
3.4 Feature Interpretation . . . 21
3.5 Experiment Setup . . . 24

4 Results **27**
4.1 Effect of CoT on Feature Interpretability . . . 27
4.2 Causal Effects of CoT Features via Activation Patching . . . 29
4.3 Activation Sparsity under CoT and NoCoT . . . 35

5 Conclusion **41**

5.1 Research Questions 41

5.2 Limitations 42

5.3 Future Work 45

5.3.1 Token-level Causal Path Analysis 45

5.3.2 Activation-grounded Explanation Methods 46

5.3.3 Scaling to Larger Models and Diverse Architectures 47

5.3.4 Subspace Patching and Circuit Discovery 47

Chapter 1

Introduction

1.1 Topic

While Large Language Models (LLMs) have shown exceptional performance in reasoning tasks [Plaat et al., 2024], their internal decision-making often remains a black box, making it hard for people to understand how the models reach their conclusions. This challenge is similar to what biologists face when studying complex systems: we may know the basic training or evolutionary principles, but the internal processes are hard to observe directly.

In parallel, the Chain-of-Thought (CoT) prompting technique has received growing attention for its ability to improve LLM performance on complex, multi-step reasoning tasks [Wei et al., 2022]. CoT encourages models to explicitly generate intermediate steps, effectively guiding them to break down problems into subgoals. Follow-up work showed that even minimal cues like "Let's think step by step" can trigger emergent reasoning behaviors in zero-shot settings [Kojima et al., 2022].

Although CoT improves performance, an open question remains: Do these intermediate reasoning steps reflect the model's true internal decision-making process? Or are they simply a plausible surface-level explanation that helps organize output better? In other words, does CoT increase the faithfulness of reasoning—that is, the degree to which the reasoning chain aligns with the internal mechanism used to reach the answer? While a few recent studies have begun

exploring this question from a representational similarity perspective [Lin et al., 2025], it has received little attention from a mechanistic, causality-based viewpoint.

Motivated by this gap, our research adopts a mechanistic interpretability perspective to address the question of whether CoT truly enhances the internal faithfulness of LLM reasoning processes. Rather than relying on external proxies or symbolic translations, we directly probe the internal semantic representations to explore how and to what extent CoT prompts shape the actual reasoning mechanisms within LLMs. Through this mechanistic lens, our study aims to clarify whether the improvement in reasoning performance associated with CoT is accompanied by a genuine increase in interpretability and internal consistency, thereby deepening our understanding of the internal logic behind LLM reasoning.

1.2 Problem Statement

To comprehensively explore the faithfulness of reasoning in LLMs, our central question is: can mechanistic interpretability reveal whether CoT enhances reasoning faithfulness? The following research questions will be addressed:

RQ1. Does CoT encourage the model to learn internal features that are more semantically consistent and easier to interpret?

RQ2. Does CoT enhance the causal relevance of internal features, as measured via activation patching?

RQ3. Can CoT promote sparser feature activations, a property commonly associated with enhanced interpretability?

1.3 Method

To address the unresolved question of whether CoT genuinely enhances the internal faithfulness of LLM reasoning as outlined in the problem statement above, this study combines sparse autoencoder (SAE) and activation patching to analyze the semantic features underlying LLM reasoning under CoT and NoCoT settings. We conduct experiments on the Grade School Math 8K (GSM8K) dataset [Cobbe

et al., 2021], a benchmark of math word problems, by (1) training separate SAEs on CoT and NoCoT activations to extract dictionary features, and (2) performing activation patching to swap these features between reasoning conditions. This causal intervention allows us to test whether features elicited by CoT prompts have a genuine influence on model output. To further investigate the semantic alignment of these internal features, we also perform a lightweight interpretation procedure that maps selected features to natural language descriptions. In doing so, we aim to go beyond attributional and symbolic methods and gain deeper insight into whether CoT actually enhances the internal consistency and transparency of LLM reasoning.

1.4 Overview

This thesis is organized as follows.

In the next chapter, we provide conceptual background and review related work on CoT prompting, reasoning faithfulness, and mechanistic interpretability, with a focus on sparse feature extraction using SAEs.

Then, Chapter 3 introduces the methodology of our study, including the model setup, the SAE training process, and the activation patching procedure.

Chapter 4 presents the experimental results, analyzing the semantic structure, causal influence, and sparsity of internal features under CoT and NoCoT prompts.

Finally, Chapter 5 addresses the research questions, discusses the limitations of our approach and outlines potential directions for future work.

Chapter 2

Related Work

Building on the motivation outlined above, this chapter reviews prior work relevant to the components of our approach. We begin with a conceptual background that motivates the use of mechanistic interpretability over more traditional methods. This is followed by an overview of CoT prompting and its role in improving reasoning performance in LLMs, a discussion on faithfulness and causal interpretability, and finally, recent advances in sparse feature extraction using SAEs.

2.1 Background

In recent years, many studies have tried to explain how LLMs reason. One line of work focuses on attributional methods, which analyze the importance of input tokens or features for the model's output. For example, Chuang et al. proposed FaithLM [Chuang et al., 2024], which aims to produce more faithful explanations through attribution techniques. However, such methods often fail to guarantee a causal relationship between highlighted input features and the model's actual decision-making, making them limited to rough correlational insights.

Another group of methods leverages knowledge-augmented reasoning, incorporating external structured knowledge—such as knowledge graphs—to increase reasoning faithfulness. Notable examples include Reasoning on Graphs (RoG) [Luo et al., 2023] and Decoding on Graphs (DoG) [Li, Zhang, Wu, Luo, Glass and Meng, 2024]. These methods provide external evidence to support the reasoning

process, yet still treat the LLM as a black box without directly probing its internal computational path.

A third direction emphasizes logic-based and symbol-based reasoning, combining LLMs with external symbolic solvers to ensure reliable reasoning. One research proposed SymbCoT [Xu et al., 2024], while another introduced LINA [Li, Li, Liu, Zeng, Cheng, Huang and Liu, 2024]. These approaches translate natural language into logical symbols and apply symbolic inference, improving precision in logical tasks. Nonetheless, challenges remain in bridging the semantic gap between symbolic representations and the subtleties of natural language, which can lead to information loss or mismatch.

In contrast to these methods, Faithful CoT [Lyu et al., 2023] introduces a distinct output-level strategy. It proposes a two-stage framework where a language model generates a hybrid natural-language and symbolic reasoning chain, and a deterministic solver executes this chain to derive the final answer. This guarantees that the reasoning chain is a faithful explanation in terms of execution. However, the generation of this chain remains fully handled by the model, without access to or control over its internal computations, rendering the approach fundamentally black box.

Despite their progress, these methods share a common limitation: they do not directly analyze the internal mechanisms of LLMs. Instead, they rely on external attribution, augmentation, or symbolic conversion to approximate or enhance model faithfulness. In other words, they still treat the model as a black box and fall short of uncovering the model's internal feature activations and computational processes at a fine-grained level.

In contrast to the above approaches, mechanistic interpretability seeks to analyze LLMs from the inside out. Instead of relying on external proxies, it investigates how specific features, neurons, or internal circuits contribute to reasoning. One research study introduced the idea of circuit-level analysis, emphasizing the role of individual components and their connections in model computation [Olah et al., 2020].

A key obstacle to understanding LLMs from a mechanistic perspective is polysemanticity—the tendency for individual neurons to respond to multiple, unrelated features [Bricken et al., 2023]. This phenomenon is believed to result from super-

position, where the model must encode far more internal features than it has available neurons. As a result, multiple features are "stacked" onto the same activation directions, especially within components like the residual stream in Transformer models. This makes it inherently difficult to interpret neurons as atomic units of meaning. Traditional mechanistic approaches often struggle to untangle such entangled representations.

A promising advance in this area is the use of SAEs [Cunningham et al., 2023]. SAEs learn a set of sparse activation directions from model internals—so-called dictionary features—which provide a finer-grained and more interpretable decomposition than raw neurons. Each feature is designed to activate strongly in only a small number of specific contexts, significantly reducing superposition and making the learned representations more monosemantic. Studies show that these features offer improved interpretability over neuron-based analysis, and crucially, they allow for causal interventions without altering the model's weights or architecture. For example, Cunningham et al. demonstrated that certain SAE-derived features can precisely identify the internal causes of model behavior in tasks like indirect object resolution, outperforming earlier attribution-based methods.

Overall, while many approaches attempt to explain the reasoning process of LLMs, most remain external to the model and fail to uncover its internal mechanisms. This work therefore adopts the emerging direction of mechanistic interpretability as the foundation for subsequent methods.

2.2 Reasoning and CoT

Recent work has extensively focused on improving and analyzing reasoning capabilities in LLMs. Among these, the CoT prompting approach has repeatedly proven effective. It was first shown that inserting intermediate reasoning steps into prompts allows large pretrained models to significantly outperform previous baselines on complex tasks such as arithmetic and symbolic reasoning [Wei et al., 2022]. For example, on the GSM8K math dataset, the accuracy of PaLM 540B increased from 17% to over 80% with CoT prompting, even surpassing some fine-tuned models.

Subsequent work introduced Zero-shot CoT, demonstrating that simply ap-

pending a phrase like "Let's think step by step." without few-shot examples is sufficient to elicit coherent intermediate reasoning, substantially boosting zero-shot performance for models such as GPT-3 [Kojima et al., 2022]. Earlier work also showed that allowing models to write out intermediate computations on a "scratchpad" can significantly enhance their performance on long arithmetic or multi-step logical reasoning tasks [Nye et al., 2021].

These findings establish CoT as a critical paradigm for reasoning in LLMs. By explicitly breaking down tasks, CoT reduces the burden of one-shot complex reasoning.

However, concerns have been raised regarding the reliability of CoT. It was found that models sometimes arrive at the correct answer despite producing incorrect intermediate steps during CoT reasoning [Yee et al., 2024]. This phenomenon of unfaithful reasoning suggests that not all CoT steps contribute causally to the final prediction. Some may be irrelevant or even incorrect, and yet the model still derives the correct answer through other latent mechanisms. For instance, a CoT response to a math problem may contain a mistaken calculation step that is later corrected, leading to a valid final answer.

This highlights the fact that CoT reasoning chains do not always reflect the model's actual decision process and may contain elements that are merely hallucinated or superficially plausible. As a result, it has sparked discussion around hallucination and faithfulness in CoT: how can we ensure that the model's outputted reasoning path aligns with its internal computation? The present study addresses this question by applying mechanistic interpretability methods to evaluate the credibility of CoT reasoning.

2.3 Faithfulness and Causal Interpretability

Faithfulness is typically defined as the extent to which an explanation accurately reflects the model's true decision-making process [Agarwal et al., 2024]. Unlike accuracy, which measures the correctness of the final output, faithfulness demands that the intermediate reasoning steps actually drive the model's decisions, rather than being post hoc justifications intended for human consumption.

In the case of LLMs, CoT reasoning represents a form of self-explanation.

Thus, its faithfulness is crucial. A fully faithful CoT chain should correspond to the actual internal computation path the model follows. If CoT is unfaithful, we cannot trust the model's reasoning process even when its answers are correct.

To evaluate faithfulness, researchers have introduced various causal analysis tools to investigate the relationship between internal activations and outputs. A key approach involves counterfactual intervention, where parts of the model's intermediate representations or parameters are selectively modified to observe their effect on the output. If intervening on a component alters the result, that component is considered causally relevant. This includes techniques like activation patching and feature ablation.

One method introduced layer-wise activation replacement to identify locations responsible for factual associations in models like GPT [Meng et al., 2022]. It was shown that mid-layer feedforward networks carry structured stages of computation, where activations during subject token processing are crucial for generating factual outputs. By identifying and editing these activations (e.g., via Rank-One Model Editing), the model's internal memory of facts was successfully altered without harming unrelated behaviors. These results support the use of counterfactual intervention to localize internal components responsible for specific reasoning functions.

Another related method is interchange intervention, in which activations from one input at a given layer are swapped with activations from another input, to measure how this substitution affects the final output. This technique has been proposed as a more robust measure of causal influence [Geiger et al., 2021]. Specifically, when swapping activations from a control input into a target input at a given node changes the output, it indicates that the node encodes contrastive information relevant to the difference between inputs. Compared to setting activations to zero or injecting noise, interchange interventions maintain the realism of activation distributions and reduce unintended statistical disruptions in the model. This method has been used in circuit discovery to identify substructures responsible for specific tasks.

In addition, studies on small-scale models have demonstrated the integration of mechanistic interpretability and causal validation. For example, in one experiment, researchers fully reverse-engineered a transformer layer solving modular arithmetic

and confirmed the causal function of its identified circuits by performing activation patching and ablation [Nanda et al., 2023].

Altogether, these findings show that testing for faithfulness requires a shift from correlation to causation. An explanation—whether in terms of neurons, hidden states, or CoT steps—can only be deemed faithful if intervening on it leads to predictable and interpretable changes in the model's behavior [Meng et al., 2022]. This study adopts this perspective by using activation patching to test whether internal representations corresponding to CoT steps have causal effects on model outputs, thereby assessing the faithfulness of CoT reasoning.

2.4 SAE and Interpretation

SAE offer a promising new approach for extracting dictionary features from LLMs. Cunningham et al. [2023] proposed training sparsity-regularized autoencoders on large-scale model activation data to learn more semantically meaningful bases than raw neurons. These features are essentially directions in the model's activation space, where each direction corresponds to a semantically coherent internal pattern. When the model processes text, a high projection onto one of these directions indicates that the corresponding semantic feature has been "triggered." Thanks to the sparsity constraint, each feature remains mostly inactive and only fires in specific contexts, substantially reducing interference due to superposition.

Although SAE-extracted features show significantly better semantic interpretability than traditional methods such as Principal Component Analysis (PCA) or Independent Component Analysis (ICA), assigning human-understandable labels to these features remains a major challenge. Several interpretation strategies have been proposed:

- Max activation set analysis: This method collects input samples that strongly activate a specific feature. Humans read these samples to find shared patterns. For example, if a feature activates for text containing math formulas, it may relate to "mathematical reasoning." Bills et al. [2023] used this approach in early neuron studies. It is simple and intuitive but relies heavily on human judgment. This makes it hard to scale to thousands of features.

- Probing classifiers: These train simple classifiers or regressors to predict predefined human-interpretable labels (e.g., negation, part-of-speech tags) based on feature activations. Probes are objective and quantifiable but limited by the availability and scope of human-defined labels. They may miss or mischaracterize emergent internal features of the model [Belinkov, 2022].

- Activation maximization: This technique optimizes the input to maximize a feature's activation, theoretically revealing the "preferred" pattern of that feature. While widely used in vision models [Erhan et al., 2009], it is less effective in language models, where direct optimization may produce meaningless token sequences.

To overcome the limitations of these methods, Cunningham et al. [2023] adopted an automatic language model-based explanation approach, originally introduced by OpenAI in 'Language Models Can Explain Neurons in Language Models' [Bills et al., 2023]. This method uses a powerful language model (e.g., GPT-4) to generate natural language descriptions of each feature based on its top-activating examples. In addition, the same or another GPT model is used to evaluate the quality of each description. This technique allows for high-throughput, scalable interpretation of thousands of features, greatly improving efficiency.

Our study follows this approach. We adopt the same GPT-based automated explanation pipeline to assign preliminary semantic labels to SAE-derived dictionary features. This choice is not novel, but rather a continuation of prior work, aimed at maximizing interpretability and scalability in support of subsequent causal analysis. At the same time, we acknowledge the risk of hallucination—that is, the possibility that generated descriptions may sound plausible while deviating from the true meaning of the features. To mitigate this, our experiments include additional quantitative metrics and counterfactual validation steps to assess the faithfulness of these automatically generated labels.

2.5 Chapter Summary

This chapter reviewed the background and related literature relevant to this study. We first outlined the limitations of traditional approaches in improving inter-

pretability, such as attribution, knowledge augmentation, and symbolic reasoning, and highlighted the potential of mechanistic interpretability for uncovering internal model mechanisms. We then discussed the role of CoT prompting in enhancing reasoning performance, along with concerns about inconsistencies in its internal reasoning paths. Additionally, we reviewed the concept of faithfulness and tools for causal analysis, and introduced SAEs as a promising method for feature extraction and semantic interpretation.

Together, these areas of research form the methodological foundation of this work. The following chapters describe how we integrate these techniques to systematically examine whether CoT prompts enhance the internal consistency and interpretability of LLM reasoning.

Chapter 3

Methodology

To investigate the internal behavior of the language model, we adopt a structured, multi-step methodology. Our approach consists of four key components:

1. Sparse autoencoding, used to extract salient latent features from the model's hidden representations by learning a sparse dictionary of activation directions;

2. Activation patching, a technique for causal intervention, where selected activations are swapped or modified between CoT and NoCoT runs to test the influence of individual features on the model's output;

3. Activation sparsity analysis, which compares the sparsity of internal representations under CoT and NoCoT conditions to assess whether CoT induces more selective or focused feature use;

4. Feature interpretation, where we use an automated explanation module to generate natural language descriptions of extracted features and assess their relevance to reasoning steps.

Together, these components form a unified framework for examining and explaining the model's internal reasoning. This allows us to identify key features, measure their activation patterns, test their causal roles, and interpret their function in a human-readable way.

3.1 Sparse Autoencoding

To understand the design of a SAE, we first introduce the basic concept of an Autoencoder (AE). An AE is an unsupervised learning method that uses a neural network to map input data to a low-dimensional latent representation, and then reconstructs the original input from this representation. Its core idea is to let the network automatically learn the most important features in the data. An AE typically consists of two parts: an encoder and a decoder.

The encoder maps an input vector $x \in \mathbb{R}^{d_{input}}$ to a lower-dimensional latent representation $h \in \mathbb{R}^{d_{hidden}}$:

$$h = f_{encoder}(x) = \sigma(W_{enc}x + b_{enc})$$

The decoder reconstructs the latent representation h back to the input space $\hat{x} \in \mathbb{R}^{d_{input}}$:

$$\hat{x} = g_{decoder}(h) = \sigma(W_{dec}h + b_{dec})$$

Where $\sigma(\cdot)$ is the activation function, typically ReLU or Sigmoid and W_{enc}, b_{enc}, W_{dec}, b_{dec} are the trainable weight matrices and biases.

The training objective of the AE is to minimize the reconstruction error between the input and the reconstructed output (commonly measured by mean squared error, MSE):

$$\mathcal{L}_{AE} = \frac{1}{N}\sum_{i=1}^{N} \|x_i - \hat{x}_i\|_2^2$$

While AE is effective at extracting features, its latent representations are often dense and difficult to interpret. To improve the interpretability of features, we introduce a sparsity constraint, forming the SAE. SAE encourages most units in the latent space to remain inactive, preserving only a few highly informative activations. This enables the model to learn clearer and more distinct features.

The training objective of the SAE extends the standard autoencoder loss by incorporating an $L1$ sparsity penalty on the hidden representations, resulting in a combined loss of reconstruction error and sparsity regularization:

$$\mathcal{L}_{recon} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \hat{x}_i\|_2^2$$

Where the reconstructed output is defined as:

$$\hat{x}_i = W_{dec}(h_i) = W_{dec}\left(ReLU(W_{enc}x_i + b_{enc})\right)$$

Here, we explicitly tie the weights of the encoder and decoder, such that $W_{dec} = {W_{enc}}^T$. This weight tying improves training efficiency and enhances interpretability.

To enforce sparse activations in the hidden layer, we apply L1 regularization:

$$\mathcal{L}_{sparse} = \frac{1}{N} \sum_{i=1}^{N} \|h_i\|_1$$

Where the hidden representation is defined as:

$$h_i = ReLU(W_{enc}x_i + b_{enc})$$

The overall SAE objective combines the two losses:

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \lambda \mathcal{L}_{sparse}$$

Here, λ is the sparsity coefficient controlling the degree of sparsity. A higher λ leads to more units in the hidden layer being suppressed. An overview of the SAE structure is illustrated in Figure 3.1.

In this study, we specifically apply the SAE method described above to residual stream activations extracted from a Transformer model. Concretely, activations from the l-th layer residual stream are treated as input vectors $x^{(l)}$. Our SAE framework leverages these activations to obtain two primary outcomes:

- Dictionary matrix (D): derived directly from the decoder weights, each column of D represents a sparse feature. These features serve as semantically meaningful components of the model's internal representation.

- Sparse latent code (h): For each input activation, the SAE provides a sparse

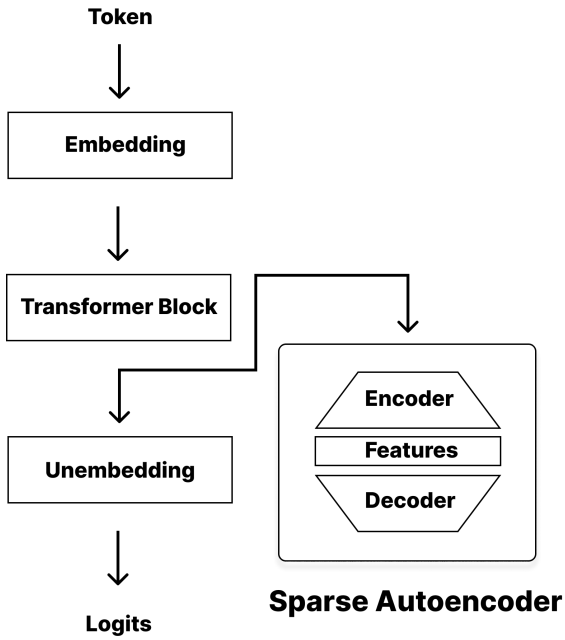


Figure 3.1: An overview of SAE. The SAE operates on the residual stream of a specific transformer layer. It receives residual activations as input and encodes them into a sparse feature representation. The decoder reconstructs the original input from this sparse code. The middle layer ("Features") captures interpretable latent dimensions, which are later analyzed or used for interventions.

encoding, indicating the presence and magnitude of key internal features.

These learned sparse features serve as essential units for subsequent analysis, interpretation, and intervention methods employed in this work.

3.2 Activation Patching

In interpretability research for modern neural networks, a central question is whether a particular internal feature genuinely contributes to the model's final output. Activation patching is a simple yet powerful causal intervention technique designed to probe such internal causal relationships.

Intuitively, activation patching can be likened to a form of machine diagnostics: if you suspect that a specific component influences a machine's behavior, you can remove it from one machine and install it in another to observe whether the

behavior changes. If the behavior of the second machine changes significantly, this suggests that the component plays a critical role.

In neural networks, activation patching is typically performed as follows: Assume we have two different input samples, denoted as A and B, passed through the same model. At layer l, the respective activations are $x_A^{(l)}$ and $x_B^{(l)}$. We construct a patched activation $x_{patch}^{(l)}$ by replacing a subset of features from $x_A^{(l)}$ with corresponding values from $x_B^{(l)}$. This can be expressed as:

$$x_{patch}^{(l)} = Patch(x_A^{(l)}, x_B^{(l)}, S)$$

where S is the subset of neurons or feature dimensions to be replaced. To the downstream layers, $x_{patch}^{(l)}$ behaves as a new hybrid state, combining features from both A and B.

We then forward-propagate the patched activation through the rest of the model to obtain the modified output $\hat{y}_{patch}$. By comparing $\hat{y}_{patch}$ with the original output y_A, we can assess the causal contribution of the replaced subset S. A significant change in the output indicates that the patched features had a meaningful impact on the model's behavior.

Formally, the procedure can be described as follows:

1. Let $x_A^{(l)}$ be the activation of input A at layer l, with original output $y_A = f(x_A^{(l)})$;
2. Let $x_B^{(l)}$ be the activation of input B, with output $y_B = f(x_B^{(l)})$;
3. Construct the patched activation by replacing a subset S of $x_A^{(l)}$ with the corresponding values from $x_B^{(l)}$:

$$x_{patch}^{(l)} = x_A^{(l)} \text{ with } x_{patch}^{(l)}[S] = x_B^{(l)}[S]$$

4. Forward-propagate $x_{patch}^{(l)}$ to obtain the patched output $\hat{y}_{patch}$;
5. Compare y_A and $\hat{y}_{patch}$. If the difference is substantial, we infer that the subset S has a significant causal effect on the model's output.

In this study, we do not perform patching directly in the original neuron activation space. Instead, we first encode the activations $x^{(l)}$ into sparse feature representations $h^{(l)}$ using a SAE. The details of SAE encoding are described earlier. Here, we focus on how patching is carried out in the sparse feature space.

For two input samples A and B, let their SAE-encoded representations be $h_A^{(l)}$ and $h_B^{(l)}$, respectively. We select a subset of sparse feature indices S and construct a patched feature vector $h_{patch}^{(l)}$ by replacing the values of $h_A^{(l)}[S]$ with those of $h_B^{(l)}[S]$:

$$h_{patch}^{(l)}[S] = h_B^{(l)}[S], \quad h_{patch}^{(l)}[\bar{S}] = h_A^{(l)}[\bar{S}]$$

We then decode this modified feature vector back into the activation space:

$$x_{patch}^{(l)} = W_{dec}\, h_{patch}^{(l)}$$

The patched activation $x_{patch}^{(l)}$ is then fed forward through the remaining layers of the model to produce the final patched output $\hat{y}_{patch}$.

Compared to patching in the original activation space, performing intervention in the sparse feature space offers several advantages. By encoding high-dimensional, entangled activations into a lower-dimensional, sparse representation, we can execute interventions in a more compact and interpretable space. Since each sparse vector contains only a small number of active dimensions, it becomes easier to isolate and observe the contribution of individual features. This structure is especially beneficial for causal analysis, as it provides a more controlled setting to test the functional impact of specific internal components. Moreover, these sparse features may align with interpretable semantics in downstream analyses, allowing for finer-grained interpretability in future work.

While our patching is consistently performed in the sparse feature space, the nature of our task requires a different evaluation strategy for measuring patching effects.

In traditional activation patching studies, such as token-level patching in NLP, researchers can typically specify a particular token position in the input sequence and evaluate the effect of patching by tracking changes in the model's prediction accuracy for that token. However, in our mathematical reasoning task, the model's output is not a single token (e.g., "fish") but a complete numerical answer or a

structured reasoning path. Furthermore, the inputs are primarily mathematical expressions or problem statements, which lack well-defined token-level alignment.

Therefore, we adopt a global output difference as the evaluation metric following patching. Specifically, let the model's output for input sample A be y_A, and the output for the patched version be $\hat{y}_{patch}$. Given the ground truth label for sample B, denoted y_B, we define the patching score as:

$$Score = |\hat{y}_{patch} - y_B| - |y_A - y_B|$$

This score quantifies whether the intervention brings the output closer to or further from the target. A negative score indicates that the model performs better after patching (i.e., the output is closer to the ground truth), while a positive score suggests a degradation in performance under the feature intervention.

To further quantify the influence of each sparse feature on the model's output, we design a progressive intervention experiment called patch curve analysis. This method builds upon feature-level activation patching by constructing a series of incremental interventions, enabling us to trace the model's sensitivity to specific internal features.

Given a pair of input samples A and B, we first encode their residual activations at layer l, denoted $x_A^{(l)}$ and $x_B^{(l)}$, into their corresponding sparse feature representations $h_A^{(l)}$ and $h_B^{(l)}$. We then compute the absolute difference for each feature dimension:

$$d_i = \left| h_A^{(l)}[i] - h_B^{(l)}[i] \right|$$

These values reflect the degree of divergence between samples A and B across different feature dimensions. We sort the indices of d_i in descending order and define the top k dimensions with the largest differences as the feature subset S_k, which will be used in the k-th stage of patching.

At each step k, we construct a patched sparse vector $h_{patch}^{(l,k)}$, in which the top k differing dimensions are taken from sample B, and the remaining dimensions from sample A:

$$h_{patch}^{(l,k)}[S_k] = h_B^{(l)}[S_k], \quad h_{patch}^{(l,k)}[\bar{S}_k] = h_A^{(l)}[\bar{S}_k]$$

This sparse vector is then decoded back into the original activation space:

$$x_{patch}^{(l,k)} = W_{dec}\, h_{patch}^{(l,k)}$$

The resulting activation $x_{patch}^{(l,k)}$ is passed through the remaining layers of the model to produce the patched output $\hat{y}_{patch}^{(k)}$. To evaluate the effect of this intervention, we define a score that measures whether the output becomes closer to the target label y_B compared to the original output y_A:

$$Score^{(k)} = \left|\hat{y}_{patch}^{(k)} - y_B\right| - |y_A - y_B|$$

A negative score indicates that adding the k-th feature improves the output (i.e., brings it closer to the target), whereas a positive score suggests a negative or counterproductive effect.

While the global output-difference score offers a coarse measure of how far the patched answer drifts from the ground truth, it is still ill-suited to mathematical-reasoning tasks. A value that is numerically "closer" to the correct answer does not guarantee that the underlying chain of reasoning is any better, and a small numerical mismatch can arise from an almost-correct derivation. In other words, surface-level distance fails to capture changes in the model's internal confidence and reasoning faithfulness.

To address this, we instead adopt a global evaluation metric based on the change in log-probability assigned to the ground-truth answer. Specifically, at each step k of the patch curve, we compute:

$$\Delta \log P^{(k)} = \log P_{patched}^{(k)}(answer) - \log P_{baseline}(answer)$$

Here, $\log P_{baseline}$ is the log-probability the model assigns to the ground-truth answer before patching, and $\log P_{patched}$ is the value after patching the selected features. This metric directly tracks how much each incremental intervention boosts—or diminishes—the model's belief in the correct solution, providing a task-appropriate, scale-invariant signal of causal influence.

We plot the log-probability gain at each step k, using the number of features patched as the x-axis and the corresponding $\Delta \log P^{(k)}$ as the y-axis. The resulting

patch curve traces the model's response to incremental interventions, revealing the cumulative causal impact of high-importance features on the final output. A steep drop in the curve in early steps suggests the presence of highly influential semantic features, whereas a flat curve may indicate feature redundancy or superposition within the model's internal representations.

These findings lay the groundwork for using sparse feature interventions to uncover interpretable internal mechanisms.

3.3 Activation Sparsity

Activation sparsity is a key indicator in understanding the internal representations of neural networks. It reflects how many units are actually "engaged" in processing a given input. When a model relies on a small subset of activations to complete a task, its internal representation can be considered more compact, focused, and potentially more interpretable and robust.

In this study, we compare the sparsity of model activations under two input conditions: inputs that include a CoT reasoning path versus standard inputs without such guidance. Our central question is whether CoT prompts lead the model to form more sparse and focused internal feature representations during reasoning.

To investigate this, we introduce two complementary sparsity analysis strategies:

First, we perform a global sparsity analysis over the residual activations of the entire input sequence. Let the residual activation at layer l be $x^{(l)} \in \mathbb{R}^{T \times d}$, where T is the number of input tokens and d is the hidden dimension. We define an activation unit as "active" if its magnitude exceeds a fixed threshold ϵ, and compute the overall sparsity as:

$$Sparsity(x^{(l)}) = 1 - \frac{1}{T \cdot d} \sum_{t=1}^{T} \sum_{j=1}^{d} \mathbb{I}\left[|x_{t,j}^{(l)}| > \epsilon\right]$$

Here, $\mathbb{I}[\cdot]$ is the indicator function, and ϵ is a small positive threshold to filter out numerical noise (e.g., 1×10^{-5}). Higher sparsity values indicate that the model activates fewer units, implying more compact and selective internal repre-

sentations.

Using this metric, we analyze the distribution of sparsity scores across all samples under both CoT and NoCoT conditions, and compare their overall trends.

Second, for computational efficiency—especially with large models—we perform a chunk-based sparsity calculation. We divide each input into non-overlapping chunks of length n, where each chunk $chunk_i$ consists of n consecutive tokens. For the residual activations in chunk i, denoted $x_i^{(l)} \in \mathbb{R}^{n \times d}$, we compute the sparsity as:

$$ChunkSparsity_i = 1 - \frac{1}{n \cdot d} \sum_{t \in chunk_i} \sum_{j=1}^{d} \mathbb{I}\left[|x_{t,j}^{(l)}| > \epsilon\right]$$

After calculating sparsity for all chunks, we aggregate these results to obtain the global sparsity distribution across the entire dataset. This chunk-based computation is purely technical, enabling efficient processing without changing the underlying global sparsity definition.

This setup enables a direct comparison of how reasoning style affects activation sparsity.

3.4 Feature Interpretation

After extracting dictionary features under both CoT and NoCoT conditions, we aim to assign each feature an intuitive, human-understandable interpretation. In other words, we want to understand what these sparse features, learned by the SAE, actually represent. This motivation stems from the intuition that if a feature is monosemantic, there should exist a concise explanation that allows us to predict when the feature will be activated.

Feature Interpretation methods are designed to discover such natural-language explanations and evaluate how well they capture the true activation behavior of each feature.

In this study, we adopt an approach inspired by OpenAI's work in Language models can explain neurons in language models [Bills et al., 2023]. The core idea is to use a pretrained language model as a simulator to test the quality of proposed

feature explanations. Specifically, for each feature, we first generate a hypothesis in natural language describing what the feature might represent. Then, using a few-shot prompting setup, we ask the language model to simulate how the feature would behave under that explanation, and we estimate the predicted activation strengths for each token in a given text.

Intuitively, if the explanation is accurate, the simulated activations should align closely with the feature's true activations from the model.

Originally, this simulation relied on access to token-level log-probabilities to estimate activation confidence. However, due to interface restrictions (e.g., GPT-3.5-turbo not exposing prompt logprobs), we adopt a logprob-free variant introduced in subsequent work. Instead of calculating all token activations at once, we construct few-shot prompts for each token position individually and ask the language model to judge whether the feature would activate at that position.

Despite being less efficient, this approach allows feature interpretation to run on modern language models like GPT-3.5-turbo or GPT-4 and greatly increases the flexibility of the pipeline.

The full interpretation pipeline proceeds as follows.

First, we begin by collecting a set of text snippets that strongly activate a given feature, along with contrastive examples that weakly activate it. We feed these examples into a powerful LLM and prompt it to summarize what is common among the highly activating examples. The resulting natural-language description becomes a candidate explanation—e.g., "Feature X appears in math-related operations" or "Feature X activates in dialogue references."

Then using a few-shot template, we guide the language model to simulate feature activation based on the explanation. The prompt includes examples showing how to judge activation (e.g., labeling tokens as "active" or "inactive"). Given a new text input, the model produces a predicted sequence of activations according to the candidate explanation.

Formally, the simulation process can be written as:

$$SequenceSimulation(E, X) = [\hat{a}_1, \hat{a}_2, ..., \hat{a}_r]$$

where E is the explanation and $X = [x_1, x_2, ..., x_T]$ is a token sequence. Each

$\hat{a}_i \in [0, 1]$ represents the predicted activation probability at token x_i. This can be viewed as the model's belief that feature F would activate at that position under explanation E.

After that, to evaluate how well the explanation matches the true behavior of the feature, we compare the predicted activation sequence $\hat{a}_i$ with the actual activation values a_i obtained from the model (e.g., from the SAE decoder output). We compute the Pearson correlation coefficient between the two sequences:

$$InterpretationScore = Corr(\{a_i\}, \{\hat{a}_i\}$$

This score reflects how closely the simulated activations match the real ones. A high score near 1 indicates strong alignment between explanation and feature behavior; a score near 0 suggests that the explanation does not capture the true semantics of the feature.

We apply this evaluation to a set of test texts, using both top-ranked (high activation) and randomly selected examples to ensure robust assessment.

Finally, we compare the interpretation scores of features under the CoT and NoCoT conditions. Specifically, we compute the distribution of interpretation scores for features extracted from models using CoT prompting and those without it. We then conduct a comparative analysis using both statistical testing and visualization.

On the one hand, we apply an independent two-sample t-test to assess whether there is a statistically significant difference in the mean interpretation scores between the two groups. This allows us to test whether features under CoT prompting are, on average, more interpretable than those under NoCoT conditions.

On the other hand, we visualize the distribution of interpretation scores using box plots, which display key statistics such as medians, interquartile ranges, and outliers. These plots offer an intuitive view of how the interpretability of features differs between the CoT and NoCoT groups.

If CoT prompting indeed enhances the monosemanticity of internal representations, we expect the distribution of CoT interpretation scores to shift toward higher values. In the box plots, this would manifest as a higher median line and a tighter concentration of scores. Moreover, the t-test would support a significant

difference between the two conditions.

In summary, the Feature Interpretation method enables us to transform the qualitative notion of "how meaningful an explanation is" into a quantitative score, making it possible to rigorously assess the impact of CoT on the semantic interpretability of model-internal features.

3.5 Experiment Setup

In this study, we selected two pretrained language models released by EleutherAI, Pythia-70M and Pythia-2.8B, as our primary subjects of analysis. Pythia-70M is a small model with 6 Transformer layers, 512 hidden dimensions, and 8 attention heads, with a feedforward hidden size of approximately 2048. In contrast, Pythia-2.8B is a large-scale model with 32 layers, a model width of 2560, 32 attention heads, and a feedforward size of roughly 10240. Both models share the same vocabulary and tokenizer, and are trained on the Pile dataset. We used the publicly available weights from the Pythia v0 release, and all experiments were conducted on these frozen models, purely for post-hoc analysis and intervention.

We employed the GSM8K dataset as the benchmark for evaluating the reasoning capabilities of the language models. GSM8K contains grade-school level math word problems, each comprising a natural language question (typically one to two sentences) and a final numerical answer [Cobbe et al., 2021]. All our analyses and activation collection experiments were conducted on the training split, as it includes ground-truth answers, while the test split remains hidden.

To investigate the effects of CoT prompting on model behavior, we created two distinct input formats for each problem: one following the CoT format, and the other following a NoCoT format. The CoT input consists of a fixed few-shot prompt followed by the current problem. The prompt contains three representative examples, each with a detailed step-by-step solution, followed by the current question prefixed with `Q: ... \nA:`. These examples are fixed across the dataset, functioning as a hardcoded prompt template rather than a dynamic in-context learning setup. In contrast, the NoCoT input includes only the current problem with no examples or step-by-step guidance.

Importantly, we only used the question portion of each example as model input,

without providing the ground truth answer. During inference, the model must generate a solution based solely on the given problem (and prompt, if applicable). Ground truth labels were used only for downstream evaluation. Additionally, we avoided input truncation by tokenizing the full problem statement, subject only to a maximum input length (e.g., 256 tokens). For activation recording, we ensured that both CoT and NoCoT samples were handled using the same formatting pipeline to avoid introducing bias.

For training the SAE and conducting activation-based comparisons, we applied the two input formats to the full GSM8K training set. That is, the amount of training data used in both CoT and NoCoT settings was identical, with the only difference being the input formatting. This controlled setup enables a fair comparison across reasoning modes, particularly in terms of residual activation sparsity, causal response to interventions, and structure of learned features.

All model loading, tokenization, inference, and intermediate activation extraction were implemented using the HuggingFace Transformers library and the TransformerLens interpretability toolkit. We extracted activations from the residual stream of layer 2 in both models. For each forward pass, we recorded the activation at the final token position, which served as the input for feature extraction and patching experiments.

To control for dictionary sparsity and feature capacity, we trained SAE models with different dictionary ratios, specifically 4 and 8, representing lower and higher sparsity settings, respectively. For each model and layer, multiple SAE variants were trained, and a representative subset was selected for downstream interpretation and intervention experiments.

During patching and evaluation, we considered two feature-selection schemes:

1. **Top-K**: the Ksparse features with the largest absolute activation difference $|h_A^{(l)} - h_B^{(l)}|$.

2. **Random-K**: a control variant that patches K features uniformly sampled from the full dictionary.

For distributional analyses, we fix $K = 20$. For patch-curve experiments, we vary $K \in 2, 4, 8, 16, 32, 64, 128$, capping the number of patched features per

sample at 128 to balance signal strength and computational cost. We evaluate up to 1000 problem pairs per condition to ensure statistical power while maintaining feasibility.

This experimental design allows us to systematically analyze the behavior of internal features under explicit reasoning conditions, and to uncover how semantic representations are structured and recombined within the sparse activation space of pretrained language models.

In summary, this chapter outlined our methodological framework for evaluating the internal faithfulness of CoT reasoning in LLMs. By training separate SAEs on activations from CoT and NoCoT inputs, we construct sparse, interpretable feature dictionaries that capture the semantic structure of the model's internal representations. We then apply activation patching to causally test whether these features influence the model's outputs and interpret selected features using automated, language-based descriptions. This approach enables us to assess the semantic coherence, causal impact, and sparsity of internal features across different reasoning conditions. In the next chapter, we apply this framework to empirical data from GSM8K, examining how internal feature structure, causal effects, and interpretability vary under CoT and NoCoT prompts.

Chapter 4

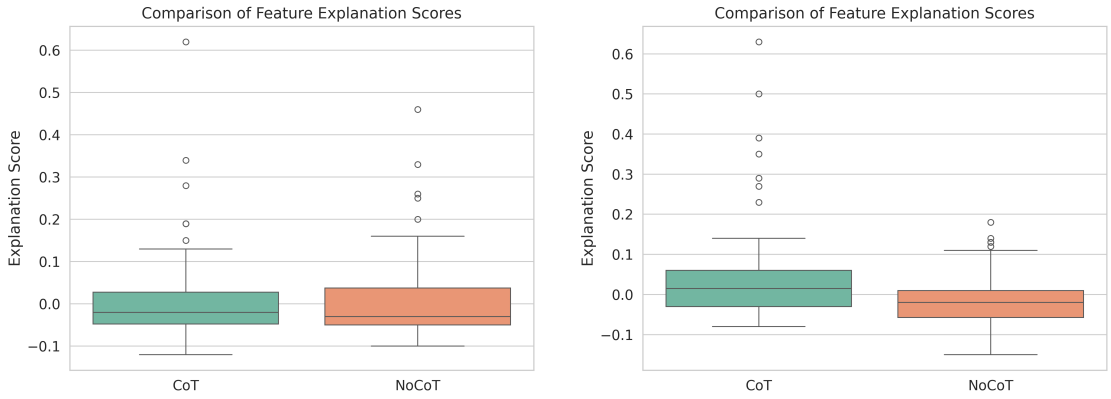
Results

This chapter applies our analysis framework to the GSM8K dataset to examine how LLMs internally represent reasoning under CoT and NoCoT prompts. We focus on three core dimensions of interpretability: the semantic coherence of sparse dictionary features, their causal influence on model outputs through activation patching, and the sparsity of internal activations. To explore how reasoning strategies interact with model capacity, we compare results across two model scales—Pythia-70M and Pythia-2.8B.

4.1 Effect of CoT on Feature Interpretability

We first compared the explanation scores of features learned under CoT versus NoCoT prompting conditions. Figures 4.1 show the score distributions for the Pythia-70M and Pythia-2.8B models, respectively, under a dictionary sparsity ratio of 4. Table 4.1 summarizes the corresponding statistical results.

For Pythia-70M, the average explanation score under the CoT condition was 0.018, compared to 0.016 under NoCoT, indicating a slight improvement. The box plot in Figure 4.1(a) further illustrates that features under the NoCoT setting performed slightly better in terms of interpretability: the median score is higher and outliers skew more positively. A t-test confirmed this observation, yielding a t-value of 0.082 and a high p-value of 0.935, suggesting that CoT may even slightly hinder interpretability in smaller models.



(a) Pythia70m (b) Pythia2.8b

Figure 4.1: Comparison of feature explanation scores under CoT and NoCoT prompts. The 2.8B model shows higher explanation scores under CoT, indicating stronger causal features are learned in the larger model when CoT prompting is applied. Each plot is based on 50 features per condition.

This result implies that for smaller models, CoT does not substantially improve the learning of semantically consistent internal features. One possible explanation is that while CoT prompts introduce more structured reasoning patterns, the limited capacity of smaller models may not be sufficient to effectively capture and utilize this added complexity. Consequently, any gains in interpretability remain minimal.

	CoT mean	CoT std	NoCoT mean	NoCoT std	T-test t	T-test p
70M	0.018	0.125	0.016	0.116	0.082	0.935
2.8B	0.056	0.147	-0.013	0.071	2.96	0.004

Table 4.1: Statistical comparison of feature explanation scores under CoT and NoCoT prompts. Results are shown for Pythia-70M and Pythia-2.8B, including mean, standard deviation, and T-test statistics.

In contrast, the results for Pythia-2.8B present a different picture. Under the CoT condition, the average explanation score was 0.056, notably higher than -0.013 for NoCoT. As shown in Figure 4.1(b), features activated by CoT prompts display a broader distribution, with some reaching values around 0.6. This suggests that CoT effectively elicits semantically coherent internal features in larger models.

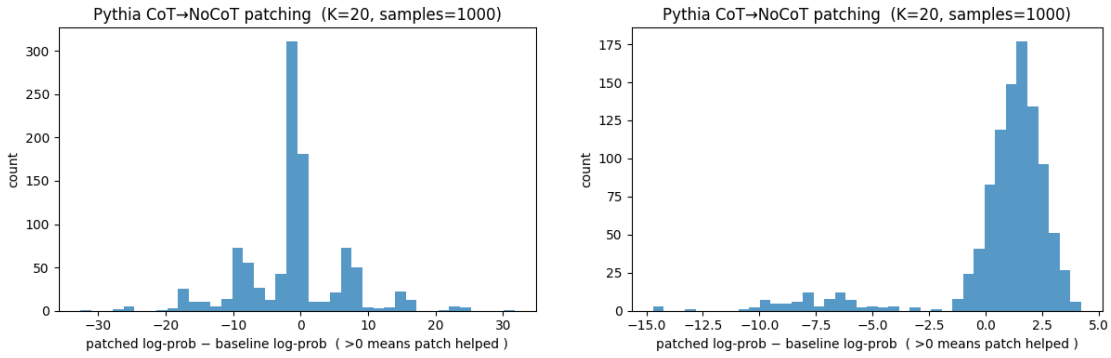
A t-test confirmed the statistical significance of this difference ($t = 2.96$, $p = 0.004$), supporting the hypothesis that CoT encourages the emergence of more interpretable representations in high-capacity models. This result underscores the potential of CoT prompting to induce structurally meaningful activations that align more closely with human-interpretable reasoning patterns.

In summary, we argue that while CoT is not a sufficient condition for inducing logically faithful reasoning chains in LLMs, it serves as an effective structural prompt—nudging models toward the activation of more semantically coherent internal features. This effect is particularly evident in larger models such as Pythia-2.8B, where CoT prompts significantly improve feature interpretability. In smaller models, however, the effect remains minimal, likely constrained by limited representational capacity. These findings are consistent with our activation patching experiments, where CoT-elicited features in larger models demonstrated causal influence on output behavior.

4.2 Causal Effects of CoT Features via Activation Patching

After establishing the semantic plausibility of SAE-derived features through interpretation, we now examine the causal role of learned sparse features through a controlled activation patching experiment. Specifically, we keep the model parameters fixed and inject the top-K most salient sparse features from a CoT forward pass into a NoCoT pass, and vice versa, in order to assess their impact on the log-probability assigned to the correct answer.

In the Pythia-2.8B model, we observe a clear directional asymmetry: CoT-to-NoCoT patching tends to improve performance, while NoCoT-to-CoT patching has minimal effect. As shown in Figures 4.2(b) and 4.3(b), this trend holds across both dictionary sparsity ratios of 4 and 8. In each case, the log-probability deltas after patching are predominantly positive, and the distribution is skewed to the right. This indicates that features activated under CoT conditions retain significant causal efficacy even when transferred to NoCoT inputs, effectively "nudging" the model toward more accurate answers.



(a) Pythia70m (b) Pythia2.8b

Figure 4.2: Distribution of log-probability changes after patching the top 20 CoT features into NoCoT runs under dictionary ratio 4. Left: Pythia-70M; Right: Pythia-2.8B. While 2.8B shows a strong positive shift indicating consistent benefit from CoT features, 70M shows highly variable effects, including large performance drops, suggesting unstable or less effective feature transfer.

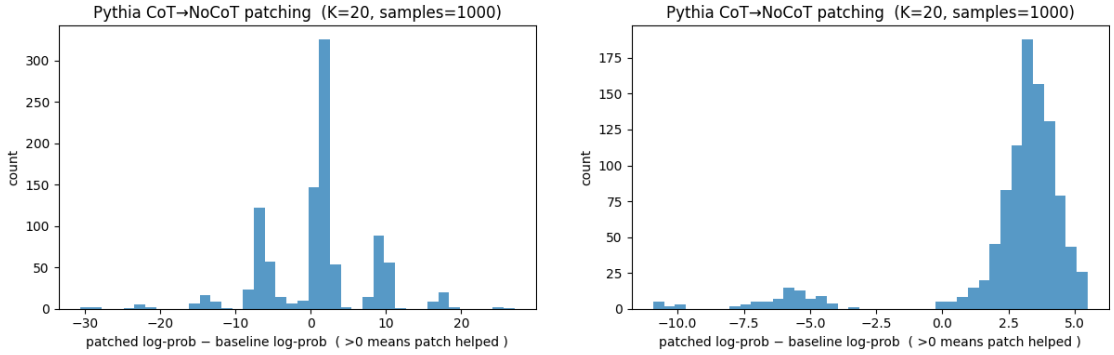
In contrast, the same patching operation in the smaller Pythia-70M model (Figures 4.2(a)) and 4.5(a) produces highly unstable results. The effect distribution is nearly symmetric around zero, with positive and negative examples occurring at roughly equal frequencies, and with extreme values (e.g., Δ log-prob reaching ± 30) prominently observed. This suggests that CoT features do not reliably transfer within the smaller model and may even interfere with the original inference trajectory in some cases.

When comparing across dictionary sparsity levels, the pattern remains consistent: both sparsity ratios yield reliable positive transfer effects in Pythia-2.8B, while Pythia-70M consistently shows no stable trend. This supports the view that the observed differences are not artifacts of a particular setup, but instead reflect a broader, capacity-dependent phenomenon—namely, that the causal utility of CoT-derived features is scale-sensitive and more robust in larger models.

To more precisely characterize the relationship between patching performance and the number of patched features K, we plot patching curves as shown in Figures 4.4 and 4.5.

We begin with the Pythia-70M model, focusing on the difference between the two patching directions: CoT $\rightarrow$ NoCoT and NoCoT $\rightarrow$ CoT.

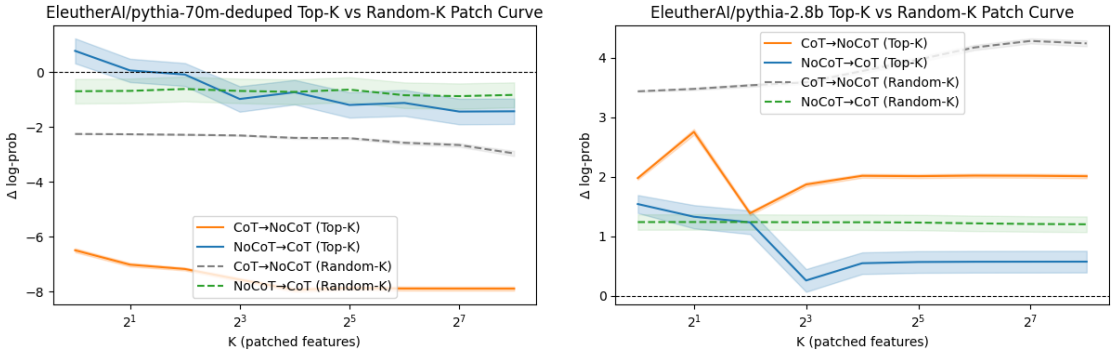
Under the dictionary sparsity ratio of 4 (Figure 4.4(a)), injecting CoT features into NoCoT trajectories (orange curve) yields no performance gain. In fact, the curve declines steadily after $K > 4$, eventually dropping to around –8 log-prob. This suggests that CoT-activated features may exhibit distributional mismatch or representational conflict in the small model, effectively disrupting the original information processing flow. Conversely, the NoCoT → CoT patching (blue curve) also leads to a decline in performance, though the drop is slightly less steep—indicating that the CoT mode may offer some robustness against perturbations.



(a) Pythia70m (b) Pythia2.8b

Figure 4.3: Distribution of log-probability changes after patching the top 20 CoT features into NoCoT runs under dictionary ratio 8. Left: Pythia-70M; Right: Pythia-2.8B. Compared to ratio 4, the distributions are similar: 2.8B continues to show consistent improvements, while 70M remains less robust, exhibiting high variance and frequent negative effects.

Under the higher sparsity setting (dictionary ratio = 8), the patching behavior of Pythia-70M continues the trend observed earlier, though the curves appear smoother (see Figure 4.5(a)). In the CoT → NoCoT direction, the patching curve remains consistently below zero, indicating that features extracted from CoT inputs fail to provide performance gains when injected into NoCoT contexts. In fact, they introduce a degree of disruption to the model's reasoning process. Although the negative impact is numerically less severe compared to the ratio 4 condition (with a minimum drop of about –3, as opposed to –6 to –8), the direction of the effect remains unchanged. This suggests that even under a more relaxed spar-



(a) Pythia70m (b) Pythia2.8b

Figure 4.4: Top-K patching performance under dictionary ratio 4. The orange line shows the effect of patching CoT features into NoCoT runs (CoT→NoCoT), while the blue line shows the reverse (NoCoT→CoT). In Pythia-2.8B, patching CoT features yields consistent performance gains, highlighting their causal importance. In contrast, for Pythia-70M, patching CoT features leads to a substantial and monotonic performance decline, suggesting that CoT-induced features are ineffective or even harmful in the smaller model.

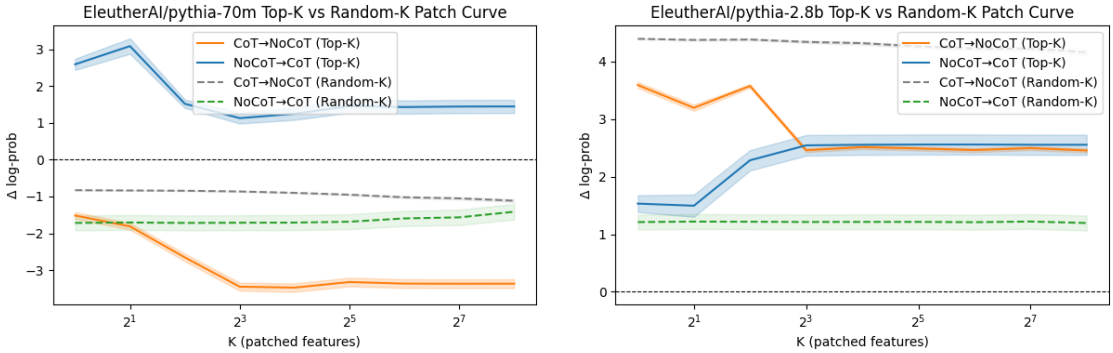
sity configuration, the small model is still unable to consistently benefit from the transfer of CoT features.

In contrast, the NoCoT → CoT direction reveals a fragile advantage of the CoT setting. At $K = 2$, the patching yields a performance boost of approximately $+3$, suggesting that the first few injected features play a meaningful role in supporting CoT-style reasoning. However, this advantage diminishes rapidly as more NoCoT features are injected, eventually stabilizing around $+1$ near $K = 128$. This trend implies that in small models, CoT-related performance gains may not be driven by a small set of dominant features, but rather distributed across a broader range of components—or that the individual utility of each feature is diluted. As a result, once these features are partially replaced or perturbed, their original advantage becomes difficult to preserve.

Moreover, more random patching experiments also show negative or unstable results. This further supports the idea that CoT-activated features in small models do not transfer well and may cause problems when added to NoCoT trajectories.

Taken together, these observations indicate that Pythia-70M does not success-

fully encode CoT features with consistent or robust causal influence. Its activation space is more dispersed, and post-patching performance shows high variability, weak directional signal, and susceptibility to disruption.



(a) Pythia70m (b) Pythia2.8b

Figure 4.5: Top-K patching performance under dictionary ratio 8. For the 2.8B model, CoT→NoCoT patching (orange) consistently improves performance, with diminishing returns as K increases. NoCoT→CoT patching (blue) gradually degrades the CoT run, suggesting CoT features are causally significant and sparse. In contrast, for the 70M model, patching CoT features into NoCoT runs still causes a net performance drop, though less sharply than under ratio 4. Interestingly, NoCoT→CoT patching shows mild improvement.

In contrast, the 2.8B model exhibits a markedly different behavior.

Under dictionary ratio 4 (Figure 4.4(b)), the orange curve (CoT → NoCoT) jumps immediately at $K = 2$, reaching a gain of over $+2.5$ log-prob, then slowly declines to around $+1.8$—indicating that the top few CoT features carry strong causal weight. In the reverse direction, the blue curve (NoCoT → CoT) remains largely flat, showing that replacing features from the CoT pathway has little to no benefit and may even introduce slight interference.

At a higher sparsity level (ratio 8, Figure 4.5(b)), this pattern becomes even more pronounced. The orange curve surpasses $+3.2$ at $K = 2$. As K increases, performance slightly declines and then stabilizes at approximately $+2.4$, revealing a classic "saturation" effect. Meanwhile, the blue curve gradually rises, indicating that NoCoT → CoT patching progressively erodes CoT-mode performance. The 2.8B model shows clear performance improvement when transferring from CoT

to NoCoT, and significant effects can be observed even with a small number of injected features.

However, after adding Random-K controlled experiments, we find that the performance gains are not due to a specific set of "Top-K strong features." Instead, in the CoT $\rightarrow$ NoCoT direction, randomly selecting K CoT-activated features often leads to better performance than using the Top-K features. This suggests that the useful information activated by CoT prompts is not concentrated in a few highly activated features, but is more widely spread across many moderately activated ones. The Top-K strategy, which only focuses on activation strength, may overfit to local peaks and miss other supportive features that actually play a causal role. In contrast, random sampling is more likely to include these overlooked features, leading to more stable and comprehensive positive effects. We will further explain this phenomenon through an analysis of feature sparsity structure in Section 4.3.

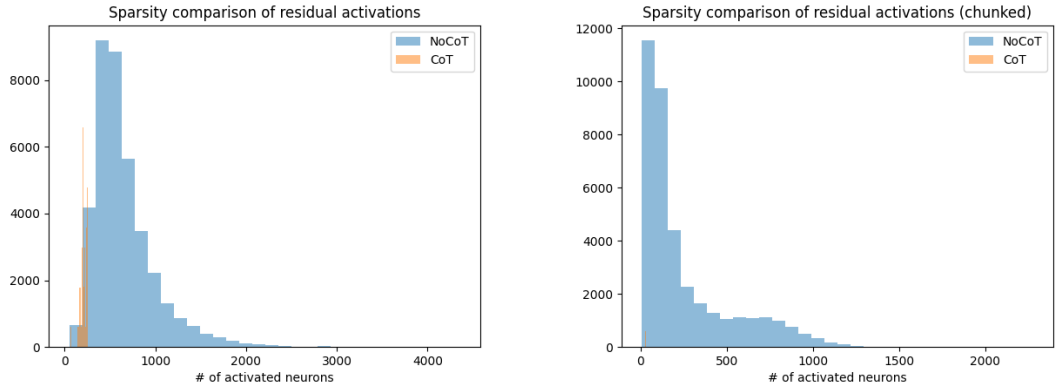
Overall, the activation patching experiments confirm that CoT-triggered features exhibit clear causal efficacy in large models: injecting even a small number of CoT-activated sparse features significantly improves model output quality. Interestingly, we find that randomly selected features often outperform top-ranked ones, suggesting that the causal signal is not concentrated in a few dominant directions but rather distributed across a broader feature space. In contrast, the CoT-activated features in small models are more scattered and fragile, lacking stable transferability and in some cases even introducing interference. All patching effects achieved statistical significance ($p < 0.001$), confirming these patterns reflect systematic differences rather than random variation.

Moreover, we observe that the sparsity ratio affects how information is distributed across features. Under higher sparsity (ratio 4), performance gains tend to occur in more abrupt "jumps" but are also more susceptible to outliers. In contrast, with lower sparsity (ratio 8), performance changes are smoother, suggesting more stable and cumulatively effective information transmission.

Together, these results support our central hypothesis: CoT prompting induces a distributed and causally meaningful internal structure, particularly in LLMs where such features are more pronounced and reliably transferable.

4.3 Activation Sparsity under CoT and NoCoT

Following the causal intervention experiments, we now turn to the structural properties of internal activations. In particular, we focus on sparsity—how CoT and NoCoT prompts affect the distribution and density of activated neurons and SAE features. Sparsity is widely associated with interpretability and generalization, and may offer additional insights into the mechanistic impact of CoT.



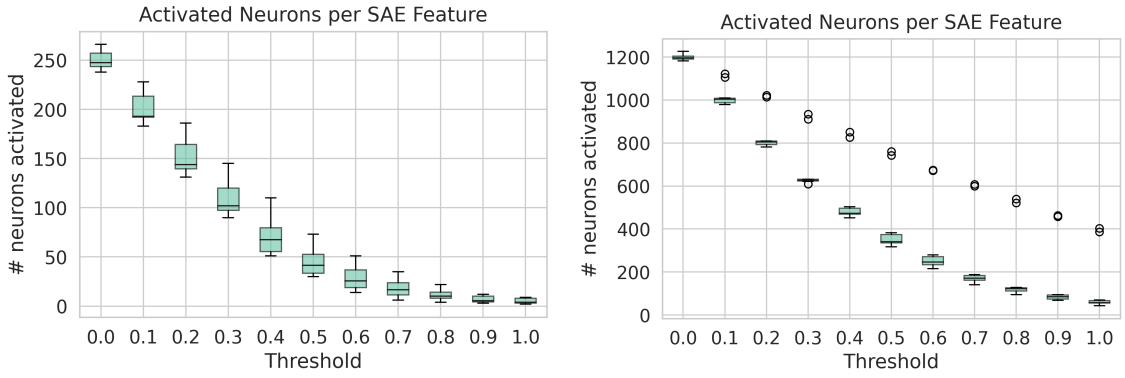
(a) Pythia70m (b) Pythia2.8b

Figure 4.6: Sparsity comparison of residual activations under CoT and NoCoT prompts. In both models, CoT leads to significantly sparser residual activations, with most neurons remaining near zero and only a small subset strongly activated. This sparsity effect is markedly more pronounced in the 2.8B model, indicating enhanced activation selectivity and structured feature usage at larger scale.

As shown in Figures 4.6, we compare the global distribution of residual activations under CoT and NoCoT prompting conditions for the 70M and 2.8B models. The results reveal that CoT prompts lead to significantly sparser residual activations compared to NoCoT prompts. Specifically, in the NoCoT condition, activation values are distributed more broadly, indicating that more neurons exhibit moderate to high activation. In contrast, under CoT prompting, most neuron activations are concentrated in a very low range, with only a few neurons showing strong activation. This sparsity trend appears in both the smaller 70M model and the larger 2.8B model, but is more pronounced in the latter. Notably, in the 2.8B model, the activation distribution under NoCoT has a heavier tail—more neurons

exhibit high activation—whereas under CoT, activations are almost entirely low, with only a small subset strongly activated, highlighting a sharper sparsity effect.

To further analyze this difference, we apply SAE to extract feature representations from the residual activations, as described in the Methods section, and count the number of significantly activated neurons per SAE feature. Figures 4.7(a) and 4.8(a) show the neuron activation distributions per SAE feature under NoCoT and CoT conditions for the 70M model. A comparison of the two reveals that under CoT, each SAE feature tends to activate only a small number of neurons, whereas under NoCoT, the same features often activate a broader set of neurons. In other words, NoCoT features are associated with more widespread neuron activations, while CoT features are more concentrated and rely on a smaller subset of neurons. This suggests that CoT leads to sparser internal representations in the 70M model, with each feature being encoded by a more compact neuronal subspace.



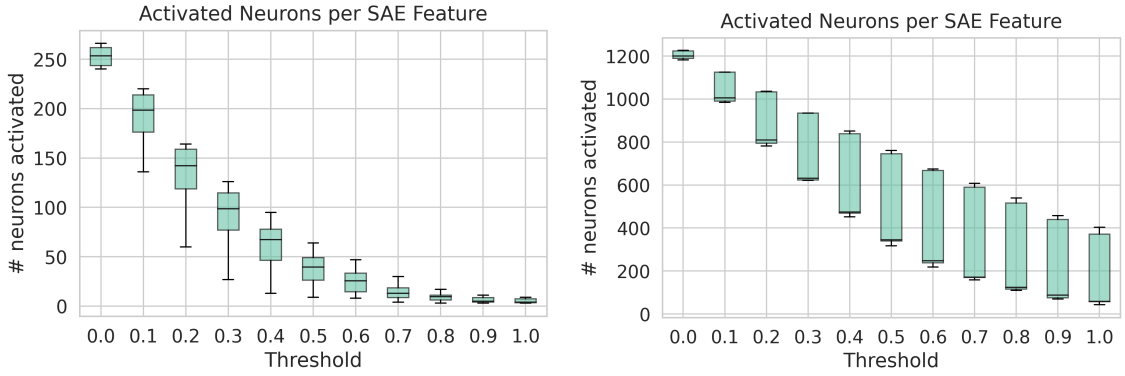
(a) Pythia70m (b) Pythia2.8b

Figure 4.7: Activated neuron counts per SAE feature under NoCoT prompting, across thresholds from 0.0 to 1.0. The large model (2.8B) activates significantly more neurons per feature at each threshold, indicating denser feature composition compared to the small model.

A similar pattern is observed in the larger 2.8B model, but to a greater extent. Figures 4.7(b) and 4.8(b) show the SAE feature activation patterns under NoCoT and CoT conditions, respectively. Under NoCoT, each feature still activates a relatively large number of neurons, while under CoT, only a very small subset is strongly activated per feature. Compared to the 70M model, the 2.8B model shows

more extreme sparsity: many features are supported by only a handful of neurons, emphasizing that larger models exhibit a more pronounced sparsity trend under CoT and may encode CoT-related features more efficiently.

This phenomenon may seem paradoxical: the CoT activations in 2.8B are globally the sparsest (Figure 4.6(b)), yet the variance in the number of activated neurons per feature is higher (Figure 4.8(b)). We interpret this as evidence of a more refined form of structured sparsity in larger models. Rather than uniformly suppressing all features, the large model under CoT appears to allocate representational resources more strategically: some features are highly focused, requiring only a few neurons, while others are more complex and involve broader neuronal collaboration. This increasing divergence in feature-level activation may underlie the superior performance of 2.8B on multi-step reasoning tasks.



(a) Pythia70m (b) Pythia2.8b

Figure 4.8: Activated neuron counts per SAE feature under CoT prompting. Compared to NoCoT, CoT prompts yield substantially sparser activations in both models, with 2.8B showing stronger sparsity and higher inter-feature variance.

Together, these experiments show that CoT prompting not only improves reasoning performance but also reshapes the internal activation patterns of the model. In both 70M and 2.8B, CoT results in fewer neurons being activated overall, indicating greater global sparsity—especially in the 2.8B model. However, this change is not limited to fewer activations: at the SAE feature level, we observe significantly greater variation in how many neurons are engaged by each feature. This suggests that CoT encourages semantic resource allocation, where some features

are represented by highly selective neurons and others mobilize a larger population for more complex reasoning. The trend is especially prominent in the 2.8B model, indicating that larger models are not only more sensitive to sparsification, but also more capable of implementing structured sparsity. We argue that this may serve as an indirect mitigation of the superposition problem: by compressing activations and increasing feature separation, CoT prompts induce a form of latent disentanglement. Although this "unsupervised disentanglement" is not explicitly optimized during training, it emerges as a byproduct of semantic prompting and plays a critical role in making internal representations more interpretable and causally effective.

Interestingly, this structured sparsity in CoT-induced representations also helps explain the surprising result from our patching experiments: in the 2.8B model, randomly sampled CoT features consistently outperform top-ranked ones when patched into NoCoT trajectories. At first glance, this seems counterintuitive—why would unranked features yield better performance than those with highest activation?

As shown earlier, CoT prompting not only makes the overall activation in both models more sparse, but also leads to stronger sparsity and higher feature-level variability in the larger model. Specifically, in the Pythia-2.8B model, under CoT conditions, most neurons have their activation values suppressed close to zero, with only a small number being strongly activated. At the same time, the number of neurons involved in different features varies much more. This means that CoT prompts in the large model lead to a form of "structured sparsity": the model does not suppress all features equally, but allocates its limited representational resources more strategically. Some features are highly concentrated and can be represented with only a few neurons, while others, which are more complex, recruit a wider set of neurons to represent them.

In other words, CoT-related information in the 2.8B model is not carried by just a few strongly activated features, but is spread across combinations of many features. Therefore, simply selecting the Top-K features based on the highest activation values may only cover local peaks in the CoT-related semantics, while missing many moderately activated but still important supporting features. These overlooked features also play a key role in final reasoning, but are not included in

the Top-K set. In contrast, when K features are selected randomly, without relying on a fixed ranking by activation strength, there is a higher chance of including these useful but less prominent features. This helps provide a more complete injection of causal information overall. This explains why, in the 2.8B model, the Random-K strategy achieves better CoT $\rightarrow$ NoCoT transfer performance than the Top-K strategy: random sampling covers a richer subset of features and avoids focusing too narrowly on local activation peaks, allowing it to capture more useful signals.

In contrast, this phenomenon does not appear in the 70M model. One possible reason is that the difference in feature distributions between CoT and NoCoT conditions is much smaller compared to the larger model. In the small model, CoT prompting does increase activation sparsity to some extent, but the overall feature activation patterns remain similar to those under NoCoT. For example, in Pythia-70M, each sparse feature under CoT typically activates only a small number of neurons, while the same feature under NoCoT might activate a wider set of neurons. However, this feature-level sparsification is much weaker than what is observed in the 2.8B model. More importantly, the limited capacity of the 70M model makes it difficult to develop new internal structures or feature organization patterns in response to CoT prompts. As discussed earlier, CoT does not significantly improve the interpretability or consistency of features in the 70M model. As a result, there are not many additional useful features emerging under CoT that the model can take advantage of. Both Top-K and Random-K strategies end up inserting features that are similarly noisy or irrelevant to the model, which naturally leads to no clear difference in performance or consistent gains. This also aligns with our earlier conclusion: smaller models, due to their limited representational power, are less capable of capturing and using the structured reasoning signals introduced by CoT prompting, and show very limited improvements in the causal relevance of their internal activations.

These results connect the earlier patching experiments with the structural analysis in this section. They show that CoT prompts create sparse, disentangled, and compositional representations in larger models, making it easier to replace features and maintain reasoning quality. In contrast, small models lack this structure, which limits their ability to benefit from CoT-style prompting. This supports the main idea that model size is critical for making CoT-induced features causally

effective and well-organized.

In summary, this chapter presented a comprehensive analysis of internal representations under CoT and NoCoT reasoning. First, we showed that dictionary features derived from CoT activations tend to be more interpretable and semantically coherent. Second, causal patching experiments revealed that CoT features in large models carry significant causal influence, exhibiting strong transferability and minimal interference. Third, we found that CoT promotes greater sparsity in both residual activations and feature composition—particularly in high-capacity models—potentially facilitating feature disentanglement. Notably, the superiority of random over top-k patching in large models further highlights the distributed and collaborative nature of CoT-induced features. Together, these findings suggest that CoT enhances the internal faithfulness of LLM reasoning by activating compact, meaningful, and causally relevant features.

Chapter 5

Conclusion

This chapter reviews the main contributions of the study and revisits the research questions introduced at the beginning. We evaluate each question based on the experimental results, then discuss the study's limitations and outline directions for future work.

5.1 Research Questions

This study investigates, from the perspective of mechanistic interpretability, whether CoT prompting improves the faithfulness of the reasoning processes within LLMs. Our experiments and analysis address the following three research questions:

RQ1: does CoT encourage the model to learn internal features that are more semantically consistent and easier to interpret?

Our results show that in the larger model, Pythia-2.8B, CoT significantly improves the semantic coherence of learned features. Several SAE features trained under CoT conditions achieve substantially higher explanation scores, indicating stronger monosemanticity and clearer interpretability. In contrast, for the smaller model Pythia-70M, CoT yields only marginal differences, with no substantial improvement over NoCoT. These findings suggest that model scale plays a critical role in enabling semantic disentanglement and the emergence of interpretable internal structure under CoT prompting.

RQ2: does CoT enhance the causal relevance of internal features, as measured

via activation patching?

Activation patching experiments show a clear effect that depends on model size. In the large model, injecting even a small number of randomly selected CoT features into NoCoT forward passes leads to a noticeable increase in output log-probabilities. This suggests that CoT features have a strong and distributed causal effect. The effect is consistent across different feature selection methods, which indicates that CoT-induced representations are not only sparse, but also semantically redundant and complementary. In contrast, similar interventions in the small model have little or even negative effect, showing that its internal features are fragile and have limited causal impact. These results suggest that CoT prompts activate meaningful mechanisms in large models, while small models do not have enough capacity to encode or make use of such structured information.

RQ3: can CoT promote sparser feature activations, a property commonly associated with enhanced interpretability?

Across both global activation distributions and the number of neurons activated per SAE feature, CoT consistently leads to greater sparsity in both models—especially in Pythia-2.8B. More importantly, the larger model exhibits structured sparsity under CoT prompting: some features are encoded by a very small number of neurons, while others engage broader neuronal collaboration. This structured allocation of representational resources likely underpins the model's success in complex reasoning tasks.

5.2 Limitations

While this study provides encouraging evidence regarding CoT's influence on internal model structure, it has several limitations.

First, our activation patching experiments only target the residual activation of the final token. They do not trace how causal effects unfold across the full reasoning process. In CoT prompting, each token—each step—may influence the final output through complex interactions. However, our analysis is limited to the last token, which makes it difficult to capture the step-by-step causal structure of reasoning. Although activation patching is a widely used tool in mechanistic interpretability [Zhang and Nanda, 2023], our current method remains coarse-

grained. We did not apply more advanced approaches like path patching, which allow interventions along multiple layers and positions, and can better reveal how information flows through the model. As a result, while we can confirm that CoT has a causal effect on the final output, we cannot explain how this effect is transmitted within the model. Our findings give a high-level view, but they do not fully capture the internal causal dynamics of CoT-based reasoning.

Second, our interpretation module relies on OpenAI's automated scoring system, which uses a LLM to evaluate explanation texts. This provides an indirect perspective on model behavior, but does not directly reflect the internal reasoning of the target model [Agarwal et al., 2024]. In effect, we are using one model to explain another, and treating the evaluator's score as a proxy for explanation faithfulness. However, this proxy may not align with the model's actual internal computations. Prior work has highlighted the difference between faithfulness and plausibility: an explanation can sound reasonable while still failing to represent how the model truly makes decisions. In our case, a high explanation score simply means that another LLM finds the explanation convincing. It does not guarantee that it reflects the real source of the activation. This "black box explains black box" setup also risks circular reasoning, since both models may share stylistic or linguistic biases. Because our explanations are not grounded in specific neurons or attention heads, and we do not test them with causal interventions, these interpretations are better seen as plausible narratives than as faithful representations of the model's internal processes. While this method offers useful insight, it lacks mechanistic precision and should be interpreted with care.

Third, due to computational limitations, our experiments were restricted to relatively small-scale models and a limited set of architectures. All analyses were performed on Pythia-2.8B and smaller variants; we did not include larger models such as LLaMA-7B or beyond. Nonetheless, our findings already show strong scale sensitivity: CoT prompts yield clear positive causal effects in Pythia-2.8B, but behave inconsistently in smaller models like Pythia-70M. Prior work has shown that reasoning traces produced by CoT-style prompting can degrade or collapse entirely as problem complexity increases, even in models explicitly designed for reasoning [Shojaee*† et al., 2025]. This suggests that interpretability at the feature level does not necessarily guarantee robust or scalable reasoning behavior and

our current conclusions may not generalize to more capable models. Moreover, the tools used in this study (e.g., TransformerLens) limit our analyses to specific architectures, mainly GPT-Neo and Pythia. We did not examine models like LLaMA or GPT-3, which differ in key design aspects [Demircan et al., 2024]. Our interventions focused solely on the residual stream at layer 2, both during SAE training and activation patching. While this provides a tractable window into sparse feature behavior, it does not capture how CoT affects representations across other layers. A broader, layer-wise analysis would thus be necessary to fully characterize CoT's structural influence within the model.

Finally, our analysis of internal features relies on SAEs to extract latent components from activation space. While this method offers a more interpretable view of model internals, it also introduces biases and uncertainties. SAEs are increasingly used in LLM interpretability to decompose activations into sparse, human-aligned features [Dooms and Wilhelm, 2025], but the quality of these features depends heavily on training assumptions and hyperparameters. In our study, we identified several features strongly associated with CoT prompts and observed increased sparsity under CoT conditions. However, this does not imply that these features are causally involved in the model's reasoning. Prior work has shown that semantically meaningful features extracted by SAEs do not always influence model outputs. For example, Menon et al. [Menon et al., 2024] found that many interpretable SAE features lack causal effect. In our experiments, only a subset of features were tested via final-token activation patching, limiting the generalizability of our causal claims. Additionally, SAEs enforce a sparse, overcomplete basis, which may fragment entangled representations or miss distributed patterns [Karvonen et al., 2024]. Given the superposition phenomenon in LLMs—where single neurons can encode multiple overlapping concepts—metrics like "number of activated neurons per feature" may not reflect true representational complexity [Bereska and Gavves, 2024]. In short, while SAE-based analysis provides a useful lens into internal structure, our conclusions about CoT-induced sparsity and interpretability are conditioned on the assumptions of this method.

Taken together, these limitations—ranging from the coarse granularity of interventions, to the indirect nature of language-based explanations, the limited generalizability across model scales, and the assumptions underlying our interpretability

tools—define important boundaries on the scope of our findings. Within current resource and tooling constraints, we have applied reasonable methods to support our core claims. Future work should trace CoT reasoning more precisely, test explanations against causal activations, expand to larger models, and cross-validate results with other methods to build a more reliable understanding of CoT's internal effects.

5.3 Future Work

Despite these limitations, our findings suggest a central insight: CoT is not just a strategy for better performance—it actively reshapes internal representations to be sparser, more structured, and more causally meaningful. This points to new directions for mechanistic interpretability. Below, we highlight several promising paths for future research.

5.3.1 Token-level Causal Path Analysis

One important extension of this work is to move beyond the final token and analyze the model's behavior across the entire reasoning trajectory at a finer granularity. Future research can explore token-level interventions within each step of a CoT sequence to map how information flows through the model and contributes to the final output.

For example, a stepwise activation replacement approach could be used to gradually insert intermediate activations from a CoT run into a NoCoT run (or vice versa), in order to pinpoint where the CoT advantage first emerges. This would help clarify the causal contribution of each token or reasoning step.

More advanced techniques such as path patching—an extension of activation patching—should also be explored [Goldowsky-Dill et al., 2023]. This method allows interventions over entire paths or subcircuits across tokens and layers, enabling analysis of joint causal effects. It may reveal whether correct reasoning depends on coordinated activations across both early and late layers, rather than isolated effects.

Constructing a token-level causal graph of the reasoning process could move

us beyond asking whether CoT works, to understanding where and how it works within the model. This would significantly deepen our understanding of CoT's internal mechanisms.

5.3.2 Activation-grounded Explanation Methods

To address the second limitation, relying on black box scoring from external language models, future work should aim to better connect natural language explanations with the model's internal behavior.

One direction is to go beyond surface-level description and incorporate mechanistic validation into the explanation pipeline. Rather than relying solely on top-activating examples, future systems could evaluate whether the activations corresponding to a given explanation causally influence the model's output. For example, after generating a candidate explanation, one could perform activation patching or ablation to test whether manipulating the aligned features changes the model's prediction. If so, the explanation is more likely to be faithful; if not, it may be plausible but misleading [Geiger et al., 2023].

Another promising approach is to use the model's own activations to guide explanation generation. Techniques such as probing or clustering can identify interpretable activation patterns, which are then described using a language model [Tighidet et al., 2024]. This grounds explanations more directly in actual model behavior.

Recent work also shows that language models themselves can explain neurons. For example, GPT-4 has been used to generate and evaluate neuron-level explanations. Based on datasets like those from Bills et al., future research could build explanation sets that focus on CoT vs. NoCoT differences, and link them to model success or failure [Bills et al., 2023].

In the long run, combining transparency (by observing model activations) with interpretability (by translating them into human language) can produce explanations that are both understandable and reliable. This would make interpretability research more robust and useful in practice.

5.3.3 Scaling to Larger Models and Diverse Architectures

Regarding the third limitation, an important future direction is to apply this framework to larger models and a wider range of architectures. We expect that some patterns seen in this study—such as CoT prompting leading to sparser and more focused feature activations—may become even stronger in larger models.

Large models are better at building specialized subcircuits. They may show more "attention separation," where certain neurons or attention heads handle specific parts of the reasoning process. For example, recent work on LLaMA-2 7B shows that when using CoT prompts, the model activates several reasoning paths in parallel and shifts from processing context to step-by-step reasoning in the middle layers [Dutta et al., 2024]. Smaller models often do not show this kind of structure.

Other studies also suggest that only very large models develop complex internal behaviors. Elhage et al. found that LLaMA-70B can compute temporal-difference reward signals in RL tasks, while the 8B model cannot [Demircan et al., 2024]. This shows that models at large scales may use more advanced internal strategies, such as scratchpad reasoning or in-context learning modules.

To better understand CoT's internal impact, future work should include models of 20B parameters or more. It is also important to test different types of models. Architectures with memory or recurrence (e.g., RETRO), or models trained with instruction tuning or RLHF, may respond to CoT in different ways.

Current tools are still limited in what they support. But community progress is helping add support for architectures like LLaMA and is improving tools for sampling and patching. Solving these engineering issues will be key for scaling this interpretability framework to more advanced models.

5.3.4 Subspace Patching and Circuit Discovery

The fourth limitation concerns the use of SAEs, which depend on training choices, may fragment overlapping features, and lack clear alignment with causal mechanisms. Future work should explore more focused and reliable methods to study how internal features influence model behavior.

One useful direction is to move from full vector changes to subspace-level in-

terventions [Makelov et al., 2023]. Instead of replacing entire activation vectors, subspace activation patching changes only specific latent directions linked to interpretable concepts. These directions can come from SAEs or other tools. This method allows researchers to test whether certain semantic subspaces—not just raw activations—actually cause behaviors like CoT reasoning.

At the same time, automated methods for finding circuits offer a scalable way to avoid manual feature selection. Tools like attribution patching [Syed et al., 2023] and causal scrubbing [Conmy et al., 2023] can find small groups of components (like attention heads or neuron clusters) that carry the CoT effect. These approaches do not rely on handpicked layers and help trace how CoT-related information flows inside the model.

Together, these methods can lead to a more modular view of how LLMs reason—by identifying working parts like scratchpads, control flows, or feature composition mechanisms that are active during CoT. This can help answer whether SAE features reflect real computational units, by testing their causal role in a clearer and more scalable way.

In conclusion, this study finds that CoT prompting not only improves outputs but also changes how the model works internally. It leads to sparser, more interpretable, and more causally meaningful representations—especially in larger models. Using tools from mechanistic interpretability, we begin to understand how CoT shapes these internal changes and what that tells us about how LLMs reason. While there are still limitations, our results suggest that structured prompts like CoT affect the model's inner workings, not just its answers. We hope this work encourages further research into the mechanisms behind CoT and helps move toward models that are both powerful and easier to understand.

Bibliography

Agarwal, C., Tanneru, S. H. and Lakkaraju, H. [2024]. Faithfulness vs. plausibility: On the (un) reliability of explanations from large language models, *arXiv preprint arXiv:2402.04614* .

Belinkov, Y. [2022]. Probing classifiers: Promises, shortcomings, and advances, *Computational Linguistics* **48**(1): 207–219.

Bereska, L. and Gavves, E. [2024]. Mechanistic interpretability for ai safety–a review, *arXiv preprint arXiv:2404.14082* .

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J. and Saunders, W. [2023]. Language models can explain neurons in language models, https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A. et al. [2023]. Towards monosemanticity: Decomposing language models with dictionary learning, *Transformer Circuits Thread* **2**.

Chuang, Y.-N., Wang, G., Chang, C.-Y., Tang, R., Zhong, S., Yang, F., Du, M., Cai, X. and Hu, X. [2024]. Faithlm: Towards faithful explanations for large language models, *arXiv preprint arXiv:2402.04678* .

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. and Schulman, J. [2021]. Training verifiers to solve math word problems, *arXiv preprint arXiv:2110.14168* .

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S. and Garriga-Alonso, A. [2023]. Towards automated circuit discovery for mechanistic interpretability, *Advances in Neural Information Processing Systems* **36**: 16318–16352.

Cunningham, H., Ewart, A., Riggs, L., Huben, R. and Sharkey, L. [2023]. Sparse autoencoders find highly interpretable features in language models, *arXiv preprint arXiv:2309.08600* .

Demircan, C., Saanum, T., Jagadish, A. K., Binz, M. and Schulz, E. [2024]. Sparse autoencoders reveal temporal difference learning in large language models, *arXiv preprint arXiv:2410.01280* .

Dooms, T. and Wilhelm, D. [2025]. Tokenized saes: Disentangling sae reconstructions, *arXiv preprint arXiv:2502.17332* .

Dutta, S., Singh, J., Chakrabarti, S. and Chakraborty, T. [2024]. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning, *arXiv preprint arXiv:2402.18312* .

Erhan, D., Bengio, Y., Courville, A. and Vincent, P. [2009]. Visualizing higher-layer features of a deep network, *University of Montreal* **1341**(3): 1.

Geiger, A., Ibeling, D., Zur, A., Chaudhary, M., Chauhan, S., Huang, J., Arora, A., Wu, Z., Goodman, N., Potts, C. et al. [2023]. Causal abstraction: A theoretical foundation for mechanistic interpretability, *arXiv preprint arXiv:2301.04709* .

Geiger, A., Lu, H., Icard, T. and Potts, C. [2021]. Causal abstractions of neural networks, *Advances in Neural Information Processing Systems* **34**: 9574–9586.

Goldowsky-Dill, N., MacLeod, C., Sato, L. and Arora, A. [2023]. Localizing model behavior with path patching, *arXiv preprint arXiv:2304.05969* .

Karvonen, A., Rager, C., Marks, S. and Nanda, N. [2024]. Evaluating sparse autoencoders on targeted concept erasure tasks, *arXiv preprint arXiv:2411.18895* .

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y. and Iwasawa, Y. [2022]. Large language models are zero-shot reasoners, *Advances in neural information processing systems* **35**: 22199–22213.

Li, K., Zhang, T., Wu, X., Luo, H., Glass, J. and Meng, H. [2024]. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains, *arXiv preprint arXiv:2410.18415* .

Li, Q., Li, J., Liu, T., Zeng, Y., Cheng, M., Huang, W. and Liu, Q. [2024]. Leveraging llms for hypothetical deduction in logical inference: A neuro-symbolic approach, *arXiv preprint arXiv:2410.21779* .

Lin, T., Xie, J., Yuan, S. and Yang, D. [2025]. Implicit reasoning in transformers is reasoning through shortcuts, *arXiv preprint arXiv:2503.07604* .

Luo, L., Li, Y.-F., Haffari, G. and Pan, S. [2023]. Reasoning on graphs: Faithful and interpretable large language model reasoning, *arXiv preprint arXiv:2310.01061* .

Lyu, Q., Havaldar, S., Stein, A., Zhang, L., Rao, D., Wong, E., Apidianaki, M. and Callison-Burch, C. [2023]. Faithful chain-of-thought reasoning, *The 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL 2023)*.

Makelov, A., Lange, G. and Nanda, N. [2023]. Is this the subspace you are looking for? an interpretability illusion for subspace activation patching, *arXiv preprint arXiv:2311.17030* .

Meng, K., Bau, D., Andonian, A. and Belinkov, Y. [2022]. Locating and editing factual associations in gpt, *Advances in neural information processing systems* **35**: 17359–17372.

Menon, A., Shrivastava, M., Krueger, D. and Lubana, E. S. [2024]. Analyzing (in) abilities of saes via formal languages, *arXiv preprint arXiv:2410.11767* .

Nanda, N., Chan, L., Lieberum, T., Smith, J. and Steinhardt, J. [2023]. Progress measures for grokking via mechanistic interpretability, 2023, *URL https://arxiv.org/abs/2301.05217* .

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D. et al. [2021]. Show your work: Scratchpads for intermediate computation with language models.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M. and Carter, S. [2020]. Zoom in: An introduction to circuits, *Distill* **5**(3): e00024–001.

Plaat, A., Wong, A., Verberne, S., Broekens, J., van Stein, N. and Back, T. [2024]. Reasoning with large language models, a survey, *arXiv preprint arXiv:2407.11511* .

Shojaee*†, P., Mirzadeh*, I., Alizadeh, K., Horton, M., Bengio, S. and Farajtabar, M. [2025]. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity.
URL: *https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf*

Syed, A., Rager, C. and Conmy, A. [2023]. Attribution patching outperforms automated circuit discovery, *arXiv preprint arXiv:2310.10348* .

Tighidet, Z., Mogini, A., Mei, J., Piwowarski, B. and Gallinari, P. [2024]. Probing language models on their knowledge source, *arXiv preprint arXiv:2410.05817* .

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D. et al. [2022]. Chain-of-thought prompting elicits reasoning in large language models, *Advances in neural information processing systems* **35**: 24824–24837.

Xu, J., Fei, H., Pan, L., Liu, Q., Lee, M.-L. and Hsu, W. [2024]. Faithful logical reasoning via symbolic chain-of-thought, *arXiv preprint arXiv:2405.18357* .

Yee, E., Li, A., Tang, C., Jung, Y. H., Paturi, R. and Bergen, L. [2024]. Dissociation of faithful and unfaithful reasoning in llms, *arXiv preprint arXiv:2405.15092* .

Zhang, F. and Nanda, N. [2023]. Towards best practices of activation patching in language models: Metrics and methods, *arXiv preprint arXiv:2309.16042* .