CIRCUIT INSIGHTS: TOWARDS INTERPRETABILITY BEYOND ACTIVATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

The fields of explainable AI and mechanistic interpretability aim to uncover the internal structure of neural networks, with circuit discovery as a central tool for understanding model computations. Existing approaches, however, rely on manual inspection and remain limited to toy tasks. Automated interpretability offers scalability by analyzing isolated features and their activations, but it often misses interactions between features and depends strongly on external LLMs and dataset quality. Transcoders have recently made it possible to separate feature attributions into input-dependent and input-invariant components, providing a foundation for more systematic circuit analysis. Building on this, we propose WeightLens and CircuitLens, two complementary methods that go beyond activation-based analysis. WeightLens interprets features directly from their learned weights, removing the need for explainer models or datasets while matching or exceeding the performance of existing methods on context-independent features. CircuitLens captures how feature activations arise from interactions between components, revealing circuit-level dynamics that activation-only approaches cannot identify. Together, these methods increase interpretability robustness and enhance scalable mechanistic analysis of circuits while maintaining efficiency and quality.

1 INTRODUCTION

Large language models (LLMs) have seen rapid adoption in recent years, including in sensitive domains such as medical analysis (Singhal et al., 2023). Despite their remarkable capabilities, we still understand very little about their internal mechanisms, which is crucial for safe and reliable deployment (Olah et al., 2018; Sharkey et al., 2025; Lapuschkin et al., 2019). Several methods have emerged in the fields of mechanistic interpretability and explainable AI to understand how models encode and utilize mechanisms that influence outputs (Olah et al., 2020; Achtibat et al., 2024; Dreyer et al., 2025). Much of the existing work focuses on circuit discovery, identifying subgraphs responsible for specific tasks (Conmy et al., 2023). However, these studies are mostly limited to toy tasks, and understanding the roles of individual neurons and attention heads still requires extensive manual analysis (Elhage et al., 2022; Wang et al., 2022; Bricken et al., 2023).

Automated interpretability methods have been proposed to address these limitations. Initial work, such as Bills et al. (2023), leveraged larger LLMs to analyze activation patterns of MLP neurons and generate natural language descriptions. Although promising, these approaches face the fundamental challenge of polysemanticity of MLP neurons, making them inherently difficult to interpret. This bottleneck prompted the development of sparse autoencoders (SAEs), which decompose activations into more monosemantic features (Bricken et al., 2023), advancing scalable interpretability pipelines (Templeton et al., 2024; Paulo et al., 2025a). More recently, Transcoders were introduced by (Dunefsky et al., 2024; Ge et al., 2024) as another alternative approach for extracting sparse features. Unlike SAEs, which reconstruct activations, Transcoders sparsely approximate entire MLP layers while maintaining a clear separation between input-dependent and weight-dependent contributions. This architecture enables efficient circuit discovery and provides direct attributions to other features, attention heads and vocabulary.

Despite these advances in sparse feature space construction, automated interpretability remains heavily dependent on explainer LLMs, which shifts the black box problem to another black box LLM, introducing notable safety risks which may produce unfaithful or unreliable explanations

(Lermen et al., 2025). Its effectiveness is influenced by the prompt, fine-tuning strategy, and the dataset used for generating explanations. Furthermore, sparse features can still be challenging to interpret (Puri et al., 2025), as they may activate on highly specific patterns that are not easily captured by analyzing activations alone, or may be polysemantic.

In this work, we focus on automated interpretability grounded in model weights and circuit structure, making the following contributions:

- We introduce *WeightLens*, a framework for interpreting models using only their weights and the weights of their Transcoders, reducing dependence on both the underlying dataset and explainer LLMs. Descriptions obtained via WeightLens match or exceed activation-based descriptions for context-independent features.
- We introduce *CircuitLens*, a framework for circuit-based analysis of feature activations, extending interpretability to context-dependent features by (i) isolating *input patterns* triggering feature activations and (ii) identifying which model outputs are influenced by specific features.

Our approach uncovers complex patterns invisible to activation-only methods, addresses the large dataset requirements and explainer LLM dependence of autointerpretability pipelines Choi et al. (2024); Puri et al. (2025). Additionally, it handles polysemanticity through circuit-based clustering and combining their interpretations into unified feature descriptions. Code for both the frameworks will be available on Github with the camera ready version.

2 Related Work

Recently, a series of works has focused on building automated interpretability pipelines for language models Choi et al. (2024); Paulo et al. (2025a); Puri et al. (2025); Gur-Arieh et al. (2025). Most approaches follow the framework introduced by Bills et al. (2023), which consists of running a large dataset through a model, collecting maximally activating samples for each neuron or SAE feature, applying various sampling strategies, and then passing the samples to a larger LLM to generate natural language descriptions.

Several studies have refined this pipeline by focusing on prompt construction and description evaluation. For instance, Choi et al. (2024), and Puri et al. (2025) examined how factors such as the number of samples and the presentation of token activations affect description quality. Choi et al. (2024) further fine-tuned an explainer model specifically to produce descriptions conditioned on a feature's activations. Beyond input-based evidence, Gur-Arieh et al. (2025) incorporated output-side information, analyzing not only what inputs trigger a feature but also how that feature influences the model's logits.

These pipelines have been applied to different representational units. Early work focused on MLP neurons (Bills et al., 2023; Choi et al., 2024), while later studies extended them to SAE features, which are generally more interpretable and often monosemantic (Templeton et al., 2024; Gur-Arieh et al., 2025; Paulo et al., 2025a; Puri et al., 2025).

As an alternative to SAEs, Dunefsky et al. (2024) and Ge et al. (2024) introduced transcoders, a sparse approximation of MLP layers that decomposes attributions into input-dependent and input-invariant components. Variants such as skip-transcoders (Paulo et al., 2025b) and cross-layer transcoders (CLTs) (Ameisen et al., 2025) have also been explored, demonstrating through qualitative and quantitative analysis that transcoders match or exceed interpretability of SAEs. Through case studies, Dunefsky et al. (2024) showed that transcoder circuits can be used for interpreting a feature's function, although based mainly on manual analysis.

The interpretability of transcoder weights has been studied by Ameisen et al. (2025), who showed that while some connections appear meaningful, interference is a major challenge. They proposed target-weighted expected residual attribution (TWERA), which averages attributions across a dataset. However, they found that TWERA weights often diverge substantially from the raw transcoder weights, making the method sensitive to the distribution of the evaluation dataset.

Finally, Puri et al. (2025) highlighted a persistent challenge: even though sparse features, specifically in SAEs, are generally more monosemantic than MLP neurons, they can be highly fine-grained and activate only on certain patterns. This specificity makes them difficult to interpret; either the

explainer LLM fails to identify the correct trigger, or the resulting description is too vague to be useful.

3 METHODOLOGY

As introduced by Dunefsky et al. (2024), given a transcoder structure, the contribution of transcoder feature i' in transcoder layer l' to feature i in layer $l > l'$ on token t can be expressed as:

$$\underbrace{\text{activation}^{(l',i')}[t]}_{\text{input-dependent}} \underbrace{(f_{\text{dec}}^{(l',i')} \cdot f_{\text{enc}}^{(l,i)})}_{\text{input-invariant}} \tag{1}$$

where $f_{\text{enc}}^{(l,i)} \in \mathbb{R}^{d_{\text{model}}}$ denotes the i-th column of the encoder matrix $W_{\text{enc}}^{(l)} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{features}}}$, and $f_{\text{dec}}^{(l',i')} \in \mathbb{R}^{d_{\text{model}}}$ denotes the i'-th row of the decoder matrix $W_{\text{dec}}^{(l')} \in \mathbb{R}^{d_{\text{features}} \times d_{\text{model}}}$, where d_{features} is the dimension of the transcoder, d_{model} is the dimension of the model, and $d_{\text{features}} \gg d_{\text{model}}$.

This formulation cleanly separates an input-dependent scalar activation from a fixed, input-invariant connectivity term between features across layers.

3.1 INPUT-INVARIANT ANALYSIS

The input-invariant connections from Equation (1) provide a useful foundation for interpretation of transcoder features, as demonstrated in the case studies of (Dunefsky et al., 2024). To build on this idea, we make the following assumptions:

Assumption 1: *Input-invariant connections indicate meaningful structural relationships only if their magnitude significantly exceeds that of other connections, making them statistical outliers.*

Since many features are context-dependent, relying solely on weight-based analysis can produce misleading results. To address this, we introduce a validation step to determine whether a feature is truly token-dependent, i.e., whether it consistently activates on specific tokens regardless of context. Formally, we state the following assumption:

Assumption 2: *If a token is strongly supported by input-invariant connections (weights) and semantically aligned with the concept encoded by the feature, then the feature should activate on this token regardless of context.*

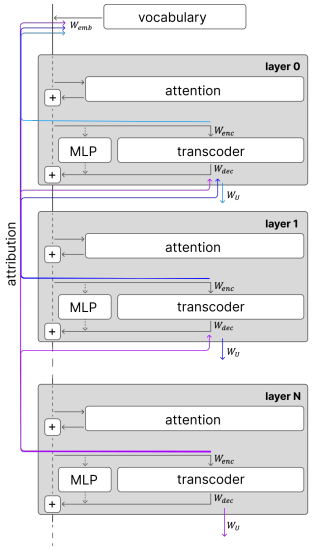
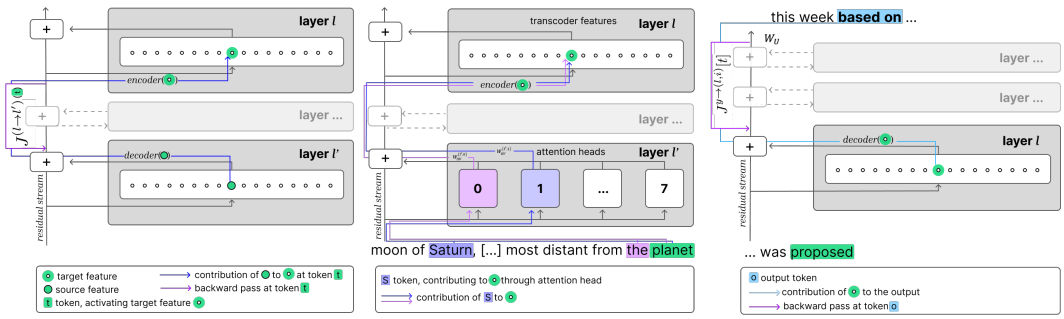


Figure 1: Process of layer-wise input-invariant analysis via WeightLens.

We generate a feature description – a set of tokens associated with the feature – by processing the model layer-by-layer, starting from layer 0 as presented on Figure 1. For this we are performing the following steps:

1. **Extract candidate tokens from vocabulary and previous layers features:** Project the feature encoder vector f_{enc} into the input vocabulary embedding space via the embedding matrix W_E as $W_E \cdot f_{\text{enc}}$, and identify candidate tokens as statistical outliers based on their z-scores (Barnett & Lewis, 1994), retaining only highly distinctive tokens. For each earlier layer $l' < l$, compute $W_{\text{dec}}^{(l')} \cdot f_{\text{enc}}^{(l)}$, identifying top contributing (outlier) features using the same criterion, and inherit their token descriptions.
2. **Validate tokens:** Retain only tokens that activate the feature in a forward pass.
3. **Analyze output effects:** Project the feature decoder vector f_{dec} into vocabulary logits via $f_{\text{dec}} \cdot W_U$, and identify outlier tokens (strongly promoted) via z-score.

Token-based features often respond to multiple forms of the same word. To process the obtained set of tokens and produce a coherent feature description, we apply lemmatization (Bird et al., 2009) to both the promoted tokens and the generated descriptions. This step consolidates different inflected forms into a single canonical form and can be considered a lightweight alternative to LLM-based postprocessing for cleaning and standardizing the descriptions.



(a) Attributions between transcoder features. (b) Attributions from transcoder features to attention heads. (c) Attributions from model logits to transcoder features.

Figure 2: Types of attributions in transcoders.

3.2 Circuit-Based Analysis

To account for interference between layers, Ameisen et al. (2025) propose incorporating a Jacobian term into the attribution formulation, which improves the reliability of feature attributions. With this adjustment, Equation (1) can be redefined as

$$\text{activation}^{(l',i')}[t] \left(f_{\text{dec}}^{(l',i')} \cdot J^{(l \to l')}[t] \cdot f_{\text{enc}}^{(l,i)} \right), \tag{2}$$

where the Jacobian is given by

$$J^{(l \to l')}[t] \;:=\; \left. \frac{\partial\, r_{\text{mid}}^{(l)}[t]}{\partial\, r_{\text{post}}^{(l')}[t]} \right|_{\substack{\text{stop-grad on MLP outputs,} \\ \text{attention patterns, norm denominators}}}, \tag{3}$$

and $r_{\text{mid}}^{(l)}[t]$ and $r_{\text{post}}^{(l')}[t]$ denote the residual streams before and after the transcoder at layers l and l', respectively (see Figure 2a).

Similarly, we can measure, how much previous tokens influenced the activation of our analyzed feature (l, i) on token t through attribution to attention heads, as presented in Figure 2b (Dunefsky et al., 2024). For an attention head h at layer l' with $l' \leq l$, the contribution of token s through (l', h) to feature (l, i) at token t can be expressed as

$$\underbrace{\text{score}^{(l',h)}(r_{\text{pre}}^{(l')}[t], r_{\text{pre}}^{(l')}[s])}_{\text{attention score from } s \text{ to } t} \underbrace{\left(\left((W_{\text{OV}}^{(l',h)})^\top f_{\text{enc}}^{(l,i)} \right) \cdot r_{\text{pre}}^{(l')}[s] \right)}_{\text{projection of feature onto head output}}, \tag{4}$$

where $r_{\text{pre}}^{(l')}[s]$ denotes the residual stream at token s before the attention block in layer l', $W_{\text{OV}}^{(l',h)}$ is the output-value matrix of head h, and $f_{\text{enc}}^{(l,i)}$ is the encoder vector of feature (l, i).

The contribution of our target feature (l, i) to the output logit $y[t]$ at token t is demonstrated on Figure 2c, and can be expressed as

$$\text{activation}^{(l,i)}[t] \left(f_{\text{dec}}^{(l,i)} \cdot J^{y \to (l,i)}[t] \cdot W_U[:, y[t]] \right), \tag{5}$$

where $f_{\text{dec}}^{(l,i)}$ is the decoder vector of feature (l, i), $J^{y \to (l,i)}[t]$ is the Jacobian from the final residual stream to the post-residual of feature (l, i) at token t (calculated as before with nonlinearities and attention patterns considered as a constant on a given input), and $W_U[:, y[t]]$ is the unembedding vector for token $y[t]$.

Interpretability Beyond Activations A central challenge in interpreting feature activations is that raw activation values do not always reveal what triggered an activation of a given feature. Simply highlighting token activations and prompting language models often yields vague or generic explanations such as "variety of words on variety of topics."[1]

[1] https://www.neuronpedia.org/gemma-2-2b/4-gemmascope-transcoder-16k/13598

To address this, we focus on identifying patterns in the data that both lead to a feature's activation and determine how the feature influences the output.

- **Input-centric focus**: Using the attribution formulation in Equation (4), we extract attention head–token pairs that provide strong contribution for a feature's activation. Outlier pairs are selected based on their z-score (Barnett & Lewis, 1994) relative to the distribution of contributions, ensuring that only the strongest connections are retained. We then mask the original input sequence, keeping only tokens that either directly activated the feature or contributed significantly through attention. This procedure isolates interpretable token patterns underlying the activation of a feature, as illustrated in Figure 2b, where the feature activates on references to already mentioned entities.
- **Output-centric analysis**: Using Equation (5), we evaluate whether the analyzed feature contributed to the prediction of the generated tokens after being activated. This highlights which output tokens were influenced by the feature and thus provides an estimate of its downstream impact, as shown in Figure 2c.

Circuit-Based Clustering A single feature often responds to multiple concepts, which may be entangled and hard to interpret. Semantic clustering in embedding space is insufficient, as it ignores the causal, circuit-level mechanisms.

We propose *circuit-based clustering*: for each input, we collect contributing elements, such as transcoder features and token/attention head pairs via Equation (2) and Equation (4) respectively, including significant transcoder features (l', i') and attention head contributions (l, h, Δ), where Δ is the relative token position. Sparse activations ensure that each input has only a few contributors.

To reduce noise, we apply a frequency filter, retaining a feature or head only if it appears in at least a fraction ρ of inputs: $\frac{|\{j : f \in S_j\}|}{N} \geq \rho$, where S_j is the contribution set for input j. This step removes features and heads that contribute only for isolated inputs and are unlikely to reflect consistent circuit-level behavior relevant to the feature activation, as well as reduces the size of the set, making subsequent clustering more robust and computationally efficient.

We then compute pairwise Jaccard similarities $J(\mathcal{A}, \mathcal{B}) = |S_{\mathcal{A}} \cap S_{\mathcal{B}}|/|S_{\mathcal{A}} \cup S_{\mathcal{B}}|$, forming an $n \times n$ distance matrix. Clusters are extracted via DBSCAN (Ester et al., 1996) on this similarity matrix, which is robust to noise and does not require a predefined number of clusters.

Sampling Strategy Most prior works (Bills et al., 2023; Choi et al., 2024; Puri et al., 2025; Gur-Arieh et al., 2025) focus on analyzing the most highly activating examples of a feature. This approach is especially sensible for MLP neurons, whose activations are often noisy and unstructured. In contrast, both SAEs and transcoders are explicitly designed to yield monosemantic features. For this reason, we aim to analyze the entire distribution of a feature's activations, in order to capture the broader concept(s) that drive its behavior.

Because activation distributions are typically highly skewed toward zero, we adopt inverse-frequency quantile sampling (Li et al., 2019; Han et al., 2022) to ensure sufficient coverage of rare but strongly activating cases. Specifically, activations are partitioned into $B = 20$ quantile bins. If n_b denotes the number of activations in bin b, then each activation i in b is assigned weight $w_i = 1/n_b^{\alpha}$ with $\alpha = 0.9$, and corresponding normalized probability $p_i = w_i / \sum_j w_j$.

Finally, we sample $N = 100$ activations without replacement, producing a diverse set of contexts that up-samples tail cases while still maintaining broad overall coverage.

Automated Interpretability Sampled inputs are first analyzed from input- and/or output-centric perspectives, then grouped into clusters according to the detected circuits. Each cluster is interpreted independently using an explainer LLM (GPT-4o-mini; see Appendix C.2). Rather than providing full inputs with highlighted activations or token–activation pairs, we supply only the detected pattern, marking the single most activating token. Finally, for each feature, the explainer LLM synthesizes a unified description from the individual cluster-level interpretations.

3.3 Evaluation

All evaluations were performed on input-centric metrics Clarity, Responsiveness, Purity and the output-centric metric Faithfulness from the FADE framework (Puri et al., 2025), where Clarity mea-

	Layer 0				Layer 7				Layer 12				Layer 21			
	C	R	P	F	C	R	P	F	C	R	P	F	C	R	P	F
WeightLens	**0.56**	0.24	0.03	0.00	**0.62**	0.22	0.02	0.01	**0.47**	0.13	0.02	0.01	**0.71**	0.86	0.65	0.02
WeightLens+Out	0.51	0.26	0.03	0.00	0.55	**0.24**	0.02	0.01	0.39	**0.15**	0.02	**0.01**	0.63	**0.87**	0.63	**0.04**
WeigthLens+Out+LLM	0.52	**0.27**	0.04	0.01	0.58	0.20	0.03	0.02	0.41	0.13	0.03	0.00	0.68	0.84	0.63	0.03
Neuronpedia	0.53	0.22	0.07	0.03	0.51	0.17	**0.10**	0.03	0.28	0.10	**0.08**	0.00	0.50	0.80	0.73	0.02
MaxAct*	0.50	0.21	**0.08**	**0.04**	0.54	0.15	0.08	**0.05**	0.33	0.11	0.7	0.00	0.53	0.80	**0.74**	0.03

Table 1: Evaluation of Gemma-2-2b transcoder descriptions (C = Clarity, R = Responsiveness, P = Purity, F = Faithfulness; metrics in range 0–1, higher is better). Best results marked in bold. Methods: WeightLens variants; Circuit-Based; Neuronpedia and MaxAct* (activation-based).

sures if the concept is expressed clearly enough to generate synthetic data, that would activate a feature. Responsiveness measures that a feature activating on the given concept is significantly-higher than the baseline activation distribution of the feature. Purity measures if the feature only strongly activates on the described concept and also on unrelated concepts. Finally, Faithfulness measures the extent to which steering the feature influences the model output in the direction of the described concept.

4 Weight-Based Interpretability Results

We analyze the interpretability of transcoders for GPT-2 Small (Dunefsky et al., 2024), Gemma-2-2b (Lieberum et al., 2024), and Llama-3.2-1B (Paulo et al., 2025b) via WeightLens, and we focus on Gemma-2-2b transcoders for qualitative analysis. We evaluate and compare:

- **WeightLens (Weights-based descriptions):** descriptions composed solely of the tokens that activate the feature, postprocessed via lemmatization;
- **WeightLens+Out (Weight-based with logits-based analysis):** weight-based descriptions augmented with tokens derived from the feature's unembedding vectors;
- **WeightLens+Output+LLM (Weight-based with logits and LLM refinement):** descriptions further refined using a secondary LLM (gpt-4o-mini-2024-07-18) to generate a concise, descriptive single-line summary based on both activating and promoted tokens (see Appendix C.1).

They are further compared to activation-based methods, specifically descriptions from Neuronpedia (Lin, 2023) and MaxAct* method (Puri et al., 2025).

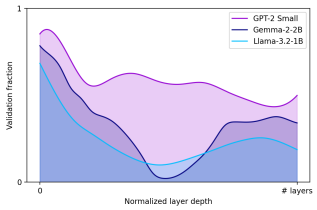


Figure 3: Percentage of validated feature descriptions per layer obtained via WeightLens.

WeightLens matches or exceeds activation-based methods Across ∼250 features per layer, our weight-based method performs on par with or better than activation maximization methods. It achieves higher scores on *Clarity* and *Responsiveness* (Table 1), while activation-based methods tend to overgeneralize, leading to lower scores. However, activation maximization attains higher *Purity*, highlighting that many features might be context-dependent which is not discovered through WeightLens.

Layer-wise trends in token-based feature interpretability Figure 3 shows that token-based interpretability varies strongly with depth.

Early layers exhibit clear token-level structure, making them well-suited for weight-based analysis, with many features activating reliably on specific tokens. In Gemma-2-2B, however, layers 0 and 7 perform poorly in terms of descriptions quality (Table 1), reflecting their high activation count ($\ell_0 = 76, 70$)[2], which also explains the weak activation-based baselines.

Number of token-based features, as well as the quality of their descriptions, drops sharply in the middle layers of Llama and Gemma, as presented on Figure 3, consistent with prior interpretability analysis (Choi et al., 2024), but not in GPT-2, similarly to Bills et al. (2023). A likely explanation is the use of RoPE in Llama and Gemma, which introduces additional non-linearities. Within Gemma-

[2]https://huggingface.co/google/gemma-scope-2b-pt-transcoders

2-2B, layer 12 is the least interpretable based on weights: despite high sparsity ($\ell_0 = 6$), it contains few validated token-based features. Majority of the features in this layer are extremely context-dependent and encode very specific patterns.

Higher layers partially recover in terms of presence of token-based features, though still below early-layer levels. For instance, Gemma-2-2B layer 21 ($\ell_0 = 13$) shows strong interpretability, with token-based features often acting as key–value pairs that map input tokens to predictable collocations (e.g., *"apologize for"*, *"will be"*).

Faithfulness is consistently low across layers and methods, likely due to the transcoder architecture: unlike SAEs, which decompose the full residual stream, transcoders write into it like MLPs, so modifying a single feature rarely produces large effects because similar concepts are distributed across layers and features.

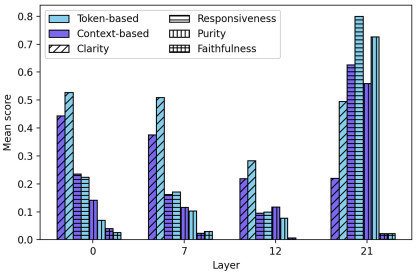


Figure 4: Comparison of Neuronpedia feature descriptions for token-based features (i.e., those for which descriptions were successfully generated via WeightLens) and context-based features, whose activations are strongly influenced by context.

Most interpretable features are token-based Only a subset of features receive validated input-invariant descriptions: 32.7% for Gemma, 58.8% for GPT-2, and 25.4% for Llama (Figure 3). However, when the weight-based descriptions fail, activation-based ones also perform poorly (see Figure 4). This is especially visible on layer 21, which overall demonstrated the best results in terms of interpretability.

LLM postprocessing is not essential Although an LLM can produce more general results and filter out noise in the output logits, it's utilization is not mandatory in this analysis, since the results are comparable, as seen in Table 1. This is a promising step to reducing reliance on the explainer models.

5 Circuit-Based Interpretability Results

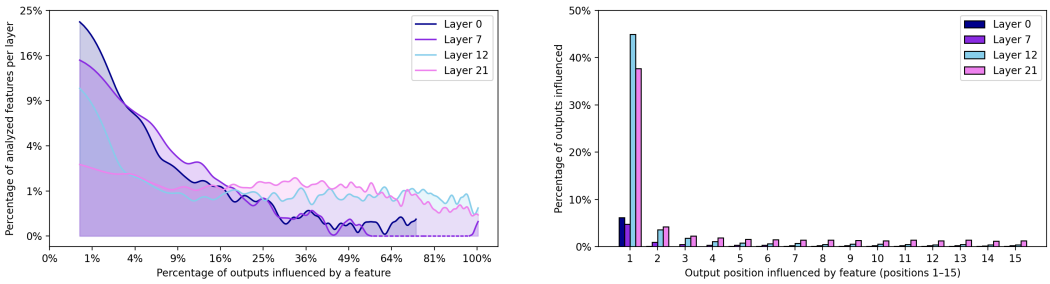
For the input-dependent analysis, we evaluate the following approaches, implemented within the CircuitLens framework:

- **CircuitLens-Input (Circuit-based descriptions):** descriptions derived from the *activation patterns* of the feature, obtained via attribution to attention heads;
- **CircuitLens-Full (Circuit-based full descriptions):** descriptions based on activation patterns obtained through *attributions to attention heads*, augmented with tokens, which generation was influenced by the feature.
- **WeightLens (WL) + CircuitLens-Full (Circuit-based full descriptions integrated with WeigthLens results):** circuit-based descriptions are *enriched with weight-based tokens* obtained via WeightLens, which are incorporated when merging cluster-level descriptions into a full feature description.

For generating circuit-based descriptions, we use sampling, described in the Subsection 3.2, on a relatively small dataset of 24M tokens (see Appendix B). In addition, to eliminate the factor of the dataset influence, we compare the circuit-based analysis performed on the same data, as was used in MaxAct* baseline, i.e. on a large dataset with sampling from the top, as presented in Appendix B. These results will be marked by (top), i.e. CB-Input (top).

As baselines, we consider Neuronpedia (Lin, 2023) and MaxAct* (Puri et al., 2025) feature desciprions. We do not analyze output-based patterns separately, as they are more computationally expensive to obtain and, in most cases, provide little additional information without the activation patterns that originally triggered the feature.

Activations alone are not sufficient Figure 5 shows three circuit-based clusters of activating input samples for feature 619 in layer 12 (L12F619). Within each cluster, some commonality in activating concepts exists, but no clear general pattern emerges from activations alone. Isolating the tokens that contribute most through attention heads reveals that each activating entity is either explicitly mentioned or marked as definite or demonstrative references such as "the," "this," or "that," as well as "former" or "latter," where contributing tokens align semantically.



(a) Smoothed histogram of the fraction of features per layer influencing a given percentage of outputs, with both axes square-root transformed to highlight skewed distributions.

(b) Fraction of features in each layer that contribute to generated outputs, plotted by output position relative to the activating token.

Figure 7: Influence of features on the new 15 generated tokens from the position of the maximally activating token on a given feature.

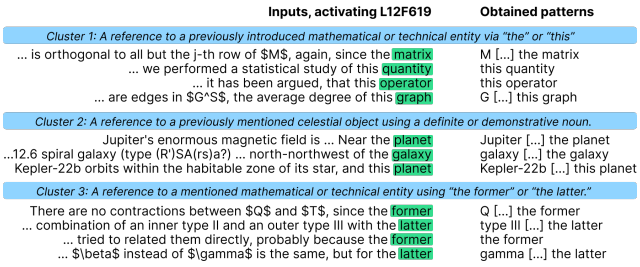


Figure 5: Clusters of activating inputs for L12F619 and their patterns, obtained through attribution to attention heads.

Extending to the output, feature L21F91 shows that functional roles are not fully captured by input activations: while it activates on tokens like "on" or certain verbs, its main effect is generating output phrases such as "the basis of" or "based on" (Figure 6). These contributions, detectable via attributions from output logits, illustrate how features influence output patterns beyond input activations.

INPUT: Subsequent **work** in this direction shall be **conducted** on
OUTPUT: **the basis** of the results of the present study.
PATTERN: [...] work [...] conducted on the basis [...]

INPUT: Therefore, we have divided subpopulations of RGB stars in NGC 5286 on
OUTPUT: **the basis of** their color and the color of their host galaxies.
PATTERN: [...] on the basis of [...]

INPUT: The **targets**, which are likely members of the cluster NGC, **were identified**
OUTPUT: using the 2MASS survey. The targets were selected **from** the ...
PATTERN: [...] targets [...] were identified [...] from [...]

INPUT: A new **mechanism** of tunnelling at macroscopic distances **is proposed**
OUTPUT: The mechanism is **based on** the existence of a new type of [...]
PATTERN: [...] mechanism [...] is proposed [...] based on [...]

Figure 6: Input patterns for feature L21F91, obtained via attributions to attention heads, and output patterns, via attributions from logits to the feature. Tokens isolated through attribution are shown in **bold**, and activating tokens are highlighted in green.

Downstream effect of a feature Output-based analysis is computationally expensive, as each generated token requires both a forward and backward pass. We generate 15 new tokens per sample to assess how many are needed for reliable results. As expected, early layers (e.g., layers 0 and 7) rarely contribute directly to the output (Figure 7a), and when they do, the effect is usually limited to the token immediately following the activating one (Figure 7b). Extending the analysis to additional tokens adds only about 1.4% of samples for layer 0, a negligible gain given the computational cost. In deeper layers, most influence remains on the first generated token, though multi-token output patterns also appear, and might be crucial for interpreting a feature's function, particularly in later layers (e.g., Figure 6).

Circuit-based polysemanticity Layer 0 has almost no underlying circuit structure: activations are largely token-based (Figure 3), with few or no clusters beyond outliers. This aligns with its low sparsity, broad topic coverage, and context-independent behavior, yielding on average only 1.05 clusters per feature (see Figure 8). By contrast, Layer 7 exhibits clear polysemanticity, averaging 4.5 clusters per feature and showing the fewest single-cluster cases, indicating widespread circuit formation. Layer 12 presents a mixed picture. While it averages 2.8 clusters per feature, many are either single-cluster or highly clustered. Qualitative inspection suggests that circuit-based clustering

	Layer 0				Layer 7				Layer 12				Layer 21			
	C	R	P	F	C	R	P	F	C	R	P	F	C	R	P	F
CircuitLens-Input	0.51	0.20	0.11	0.01	0.39	0.14	**0.14**	0.03	0.19	0.10	0.10	0.01	0.42	0.59	0.51	0.03
CircuitLens-Full	0.52	0.22	**0.11**	0.02	0.44	0.14	0.13	0.03	0.24	0.09	0.09	0.01	0.52	0.64	0.54	0.03
CircuitLens-Input (top)	0.66	0.24	0.05	0.02	**0.62**	0.17	0.08	0.03	**0.27**	0.09	0.08	0.00	0.54	0.72	0.63	0.03
CircuitLens-Full (top)	**0.66**	**0.24**	0.04	0.02	0.61	**0.18**	0.07	**0.04**	0.26	0.10	0.07	0.01	**0.56**	**0.73**	0.62	0.03
WL + CircuitLens-Full	0.55	0.22	0.10	0.02	0.51	0.14	0.11	0.02	0.25	0.10	0.08	0.01	0.55	0.68	0.56	0.03
Neuronpedia	0.51	0.23	0.09	0.03	0.44	0.17	0.11	0.03	0.15	0.10	**0.11**	0.00	0.36	0.71	0.64	0.02
MaxAct*	0.50	0.21	0.09	**0.04**	0.48	0.15	0.09	0.03	0.17	0.10	0.10	0.00	0.39	0.71	**0.64**	0.02

Table 2: Evaluation of Gemma-2-2b transcoder descriptions (C = Clarity, R = Responsiveness, P = Purity, F = Faithfulness; metrics in range 0–1, higher is better). Best results marked in bold. Methods: Circuit-Based variants. Baselines: Neuronpedia and MaxAct*.

can capture both main circuits and sub-circuits, and more careful hyperparameter tuning could yield more generalizable results (Figure 5). It also has the largest share of outlier-only features, reflecting the difficulty of disentangling circuit-driven from semantic-driven activations. Layer 21 is similar (avg. 2.95 clusters/feature) but shows fewer extreme cases. As this layer is closely tied to output generation, we hypothesize that features flexibly participate in multiple circuits to support specific outputs (Figure 6).

Resolving the Clarity Problem As shown by Puri et al. (2025), sparse feature descriptions are often not clear enough, as they fail to specify what precisely triggers a feature's activation. In Table 2, we compare circuit-based analysis with activation-based baselines, including experiments performed on a smaller dataset sampled from the full distribution, as described in Section 3.2 (CircuitLens-Input and -Full), and on a MaxAct*-like dataset (CircuitLens (top)), which is drawn from a much larger corpus but restricted to top activations (see Appendix B)

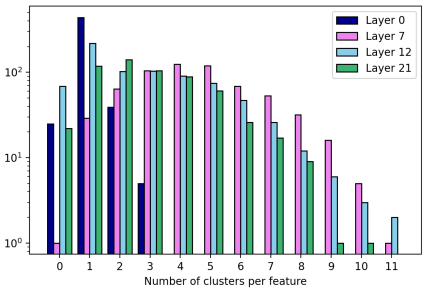


Figure 8: Histogram of number discovered clusters per layer with log scale (y-axis).

Our results show that circuit-based methods still depend on the dataset, with descriptions from the larger dataset achieving the strongest performance across layers, particularly in clarity and responsiveness. However, descriptions derived from the smaller dataset remain competitive and in some cases outperform activation-based baselines, generated on the larger dataset. Combining weight-based and circuit-based analysis further reduces sensitivity to dataset size and distribution, making interpretability more robust. Moreover, analysis of the metrics distributions reveals that circuit-based methods yield far fewer features with extremely low clarity compared to purely activation-based approaches (see Appendix D). Finally, both qualitative and quantitative evidence indicate that sampling from the full distribution, though memory-intensive, provides a more faithful picture of general feature behavior.

CONCLUSION

In this work, we address a fundamental missing piece in automated interpretability pipelines by developing methods that leverage models' underlying structural information. We show that raw activations alone often fail to reveal the patterns driving feature activation, while transcoders, well-suited for circuit discovery, enable efficient incorporation of structural information. Our proposed frameworks demonstrate that structural information allow more scalable and robust interpretability. The weight-based analysis offers an efficient alternative for context-independent features – covering up to 58.8% of cases – without requiring large datasets or external LLMs. Circuit-based clustering isolates groups of activating texts into more interpretable clusters, while input- and output-based analysis further clarifies each feature's functional role. Together, these methods reduce dependence on large datasets, improve robustness, and make automated interpretability more scalable and practical for real-world applications. By bridging activation-based approaches with weight-based analysis and circuit discovery, our work opens new avenues for understanding model behavior at scale.

REFERENCES

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. Attnlrp: attention-aware layer-wise relevance propagation for transformers. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 135–168, 2024.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Vic Barnett and Toby Lewis. *Outliers in Statistical Data*. Wiley, 3rd edition, 1994.

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

Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009. ISBN 978-0-596-51649-9. URL https://www.nltk.org/book/.

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

Dami Choi, Vincent Huang, Kevin Meng, Daniel D Johnson, Jacob Steinhardt, and Sarah Schwettmann. Scaling automatic neuron description, 2024.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Maximilian Dreyer, Jim Berend, Tobias Labarta, Johanna Vielhaben, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. Mechanistic understanding and validation of large ai models with semanticlens. *Nature Machine Intelligence*, pp. 1–14, 2025.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits, 2024. URL https://arxiv.org/abs/2406.11944.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pp. 226–231. AAAI Press, 1996.

Leo Gao, Stella Biderman, Sid Black, et al. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL https://arxiv.org/abs/2101.00027.

Xuyang Ge, Fukang Zhu, Wentao Shu, Junxuan Wang, Zhengfu He, and Xipeng Qiu. Automatically identifying local and global circuits with linear computation graphs, 2024. URL https://arxiv.org/abs/2405.13868.

Yoav Gur-Arieh, Roy Mayan, Chen Agassy, Atticus Geiger, and Mor Geva. Enhancing automated interpretability with output-centric feature descriptions, 2025. URL https://arxiv.org/abs/2501.08319.

Bingyi Han, Yuan Yao, and Yisen Zhang. Frequency-aware sampling for deep learning on long-tailed data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Laura Kopf, Nils Feldhus, Kirill Bykov, Philine Lou Bommer, Anna Hedström, Marina M-C Höhne, and Oliver Eberle. Capturing polysemanticity with prism: A multi-concept feature description framework. *arXiv preprint arXiv:2506.15538*, 2025.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1096, 2019.

Simon Lermen, Mateusz Dziemian, and Natalia Pérez-Campanero Antolín. Deceptive automated interpretability: Language models coordinating to fool oversight systems, 2025. URL https://arxiv.org/abs/2504.07831.

Bo Li, Yu Wang, Yixin Liu, Zhifeng Zhang, and Shuo Li. Frequency-aware weighted loss for imbalanced classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning, 2023. URL https://arxiv.org/abs/2308.03281.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, Janos Kramar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.19. URL https://aclanthology.org/2024.blackboxnlp-1.19/.

Johnny Lin. Neuronpedia: Interactive reference and tooling for analyzing neural networks, 2023. URL https://www.neuronpedia.org. Software available from neuronpedia.org.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 2018. doi: 10.23915/distill.00010. https://distill.pub/2018/building-blocks.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

Josh OpenAI andAchiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni, et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2025a. URL https://arxiv.org/abs/2410.13928.

Gonçalo Paulo, Stepan Shabalin, and Nora Belrose. Transcoders beat sparse autoencoders for interpretability, 2025b. URL https://arxiv.org/abs/2501.18823.

Bruno Puri, Aakriti Jain, Elena Golimblevskaia, Patrick Kahardipraja, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. FADE: Why bad descriptions happen to good features. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17138–17160, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.881. URL https://aclanthology.org/2025.findings-acl.881/.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open problems in mechanistic interpretability, 2025.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

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

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

A Extended Related Work

A.1 Automated Interpretability

Most work in automated interpretability builds on the pipeline of Bills et al. (2023), where a dataset is processed through GPT-2 (Radford et al., 2019) to collect MLP neuron activations. A larger LLM (GPT-4; (OpenAI andAchiam et al., 2024)) then generates descriptions based on top-k activating sequences, and is further used as an "activations simulator" to evaluate these descriptions.

To address the polysemanticity of MLP neurons, Bricken et al. (2023) propose sparse autoencoders (SAEs). Templeton et al. (2024); Paulo et al. (2025a) extend this pipeline to SAEs and show that their features are more monosemantic and interpretable than MLP neurons.

Subsequent work refines automated interpretability in different ways (Choi et al., 2024; Templeton et al., 2024; Gur-Arieh et al., 2025; Kopf et al., 2025; Paulo et al., 2025a; Puri et al., 2025). Choi et al. (2024) fine-tune a smaller LLM (Llama-3.1-8B-Instruct (Dubey et al., 2024)) for neuron description and evaluation, systematically studying prompt design choices such as token highlighting, token–activation pairs, and number of examples. Puri et al. (2025) run a broader prompt analysis on SAEs, finding results that sometimes contradict Choi et al. (2024), especially on how token activations should be communicated. They also emphasize dependence on explainer model quality and challenges from the fine-grained specificity of SAE features.

Gur-Arieh et al. (2025) combine input- and output-based analysis, showing that descriptions improve when considering both what a feature activates on and what it promotes.

Kopf et al. (2025) address polysemanticity by clustering activating inputs. They sample from the top 1% of activations, embed sequences with gte-Qwen2-1.5B-instruct (Li et al., 2023), and apply k-means clustering into five groups, generating one description per cluster. This consistently outperforms prior work (Bills et al., 2023; Gur-Arieh et al., 2025), demonstrating the benefit of handling polysemanticity directly.

A.2 Evaluation metrics

Developing automated evaluation metrics is essential for interpretability research, since manual assessment of description quality does not scale. Several main approaches have been proposed: (i) simulated activations, where an LLM predicts a feature's activation on text samples given its description (Bills et al., 2023; Choi et al., 2024); (ii) classifier-based metrics, where an LLM judges how strongly a text sample relates to a feature's description (Templeton et al., 2024; Paulo et al., 2025a; Puri et al., 2025); (iii) synthetic data approaches, where an LLM generates or labels data from a description (Gur-Arieh et al., 2025; Puri et al., 2025); and (iv) output-based metrics, which evaluate how much a feature influences model outputs (Bills et al., 2023; Gur-Arieh et al., 2025; Paulo et al., 2025a; Puri et al., 2025).

Simulated-activation metrics (Bills et al., 2023; Choi et al., 2024) are inexpensive but fail to capture many failure modes in description generation (Puri et al., 2025).

Classifier-based metrics instead ask a judge LLM to score how related a sample is to a description, often on a scale from 0 (not related) to 3 (completely related) (Templeton et al., 2024; Puri et al., 2025). Similar detection-based setups appear in Paulo et al. (2025b), where the model identifies which samples match the concept. Evaluation can then be quantified using AUROC scores (Kopf et al., 2025), or metrics such as Gini coefficient and Average Precision, which Puri et al. (2025) combine into Responsiveness and Purity scores.

Synthetic data metrics compare activating and non-activating examples, either LLM-generated or sampled uniformly from a dataset (Gur-Arieh et al., 2025; Puri et al., 2025).

Finally, output-based metrics test whether descriptions capture a feature's causal effect on model outputs. Puri et al. (2025) propose *Faithfulness*, where a judge LLM rates concept presence in steered generations. Paulo et al. (2025a) introduce *Intervention Scoring*, while Gur-Arieh et al. (2025) apply a similar approach; where the model's task is to distinguish outputs produced under feature steering from control generations.

A.3 Circuit tracing

A recent advance in mechanistic interpretability is the introduction of transcoders (Dunefsky et al., 2024; Ge et al., 2024). Unlike sparse autoencoders (SAEs), transcoders provide a structured way to trace how upstream feature activations contribute to downstream activations, enabling circuit-level analysis across layers. Their key innovation is the decomposition of a feature's activation into an input-dependent and an input-independent component. The latter depends only on transcoder weights, allowing it to be analyzed separately and efficiently.

Using GPT-2, Dunefsky et al. (2024) demonstrate cases where activating tokens identified through weight-based analysis align with those discovered via traditional activation-based methods. This indicates that transcoders can support both prompt-specific attribution graphs and global, weight-derived connectivity maps.

Extending this work to Gemma-2-2B (Riviere et al., 2024), Ameisen et al. (2025) highlight a major limitation of weight-based analysis: interference from context-dependent components of the architecture. To address this, they introduce *target-weighted expected residual attribution* (TWERA), which adjusts virtual weights using empirical coactivation statistics, effectively up-weighting connections between frequently coactivating features. However, they also show that TWERA can significantly diverge from the original transcoder weights, making it dependent on the dataset used to compute coactivations and limiting its reliability as a fully weight-based method.

B Dataset Processing

We used the uncopyrighted version of the Pile dataset Gao et al. (2020), available on Hugging Face (`https://huggingface.co/datasets/monology/pile-uncopyrighted`) with all copyrighted content removed. This version contains over 345.7 GB of training data from various sources. From this dataset, we extract two datasets of size 6GB (3.6B tokens), and 40MB (24M tokens) for generating MaxAct* descriptions and circuit based descriptions. The extracted portion from the training partition was used to collect the most activated samples based on frequency

quantile sampling for the smaller dataset and top percentile sampling for the larger dataset. For evaluations, we utilized the test partition from the same dataset, applying identical preprocessing steps as those used for the training data.

Post processing involves several steps to ensure a balanced and informative dataset. First, we used the NLTK Bird et al. (2009) sentence tokenizer to split large text chunks into individual sentences. We then filtered out sentences in the bottom and top fifth percentiles based on length, as these were typically out-of-distribution cases consisting of single words, characters, or a few outliers. This step helped achieve a more balanced distribution. Additionally, we removed sentences containing only numbers or special characters with no meaningful content. Finally, duplicate sentences were deleted.

C Descriptions Generation

C.1 Postprocessing Weight-Based Descriptions

LLM-based postprocessing of weight-based analysis enables generating smoother, more coherent feature descriptions by consolidating input- and output-centric information, specifically, the tokens that activate a feature and those that it promotes or suppresses. This approach is particularly effective at filtering out noise, as promoted or suppressed tokens often include unrelated or random terms that do not reflect the feature's true function.

```
We're studying neurons in a neural network. Each neuron has certain inputs that activate it
     and outputs that it leads to. You will receive three pieces of information about a neuron
     :

1. The top important tokens.
2. The top tokens it promotes in the output.
3. The tokens it suppresses in the output.

These will be separated into three sections: [Important Tokens], [Text Promoted], and [Text
     Suppressed]. All three are a combination of tokens. You can infer the most likely output
     or function of the neuron based on these tokens. The tokens, especially [Text Promoted]
     and [Text Suppressed], may include noise, such as unrelated terms, symbols, or
     programming jargon. If these are not coherent, you may ignore them. If the [Important
     Tokens] do not form a common theme, you may simply combine the words to form a single
     concept.

Focus on identifying a cohesive theme or concept shared by the most relevant tokens.

Your response should be a concise (1-2 sentence) explanation of the neuron, describing what
     triggers it (input) and what it does once triggered (output). If the input and output are
      related, you may mention this; otherwise, state them separately.

[Concept: <Your interpretation of the neuron, based on the tokens provided>]

Example 1

Input:
[Important Tokens]: ['on', 'pada']
[Tokens Promoted]: ['behalf']
[Tokens Suppressed]: ['on', 'in']

Output:
[Concept: The token "on" in the context of "on behalf of"]
}
...
```

C.2 Generating Circuit-Based Descriptions

At the first step, we treat each cluster of a feature separately. We pass the obtained patterns (input-centric or full, i.e. with patterns detected in the model's output) in there in order to generate a description.

```
You are an explainable AI researcher analyzing feature activations in language models.
You will receive short patterns: fragments of text where tokens activated a feature.
ONLY the snippets shown are the evidence, do not assume any extra surrounding context.
```

```
Pattern formatting:
- _ is 1-3 skipped non-important tokens
- [...] is 4 or more skipped not relevant tokens
- The <<<highlighted>>> token of each snippet is usually the most important signal (it is the
    activating token), it can be a part of a word.

Analysis procedure:

1. Do NOT start by interpreting semantics. First treat the data as raw strings.
2. Count and note repeated literal elements (words, single letters, punctuation, suffixes/
    prefixes, LaTeX tokens, different symbols, brackets, arrows, parentheses).
3. Pay special attention to:
  - exact repeated tokens,
  - repeated punctuation or formatting (commas, superscripts, backslashes, braces),
  - positional patterns,
  - capitalization patterns and single-letter variable tokens,
  - functional words, like articles, pronouns, modal verbs, that create a consistent pattern.
4. Only after the literal/structural check, generalize into a short concept (if appropriate).

Decision rules:

- If a single literal token or structural pattern dominates, output that token or structural
    label exactly.
- If it is some grammatical pattern, output exactly that.
- Avoid speculative semantic labels unless literal patterns support them.

Output rules:

- Output exactly ONE concise sentence (<20 words) describing the shared concept or structure.
- If a single token/pattern dominates, output it exactly.
- You may include up to one short example group in parentheses to clarify.
- Do NOT include extra labels or the word "Description:".
- If no clear recurring concept or structure is found, output exactly: No concept found.
- Avoid vague phrases like "in various contexts" or 'a variety of words'.
- Do NOT output your internal reasoning, only the final single sentence.

Example 1

Input:
important to
helps to
permits to
importance to
is possible [...] to
able to
allows us to
purpose [...] is to

Output:
Preposition "to" in phrases that express purpose, intention, or enable an action.
```

At the next iteration, we combine the obtained cluster descriptions into a single one.

```
You are an explainable AI researcher analyzing multiple related concepts.
You will receive a list of **concept descriptions**, each representing a semantic, grammatical
    , or functional element.

[ONLY FOR WeightLens+CircuitLens COMBINED EXPERIMENTS:
Sometimes, you may also receive a phrase at the beginning like "Important tokens: ...", for
    example:
   Important tokens: amazing, largely, upon.
   Important tokens: danger, preparation, prepare, preparing.
   Important tokens: new.
Always integrate these tokens into your description, even if they do not fit naturally with
    the other concept descriptions. ]

Step-by-step reasoning:
1. Examine all provided concepts carefully. Identify recurring themes, functions, or semantic
    roles.
2. Look for commonalities across the concepts, including:
  - grammatical elements (articles, parts of sentences, syntactic patterns)
  - symbols and punctuation (commas, brackets, etc.)
  - semantic categories
  - mathematical or symbolic markers
3. Pay special attention to specific patterns, which often are described through function
    words (articles, modal verbs, etc.).
4. Merge similar or overlapping elements into a single, concise idea.
5. Think step by step:
  a) Identify the core function or role each concept serves.
  b) Group related concepts together.
```

```
  c) Combine them into one coherent description.

Output rules:
- Output exactly **one concise sentence** (<30 words) describing the shared concept or several
      main concepts.
- Include all major elements, but merge overlapping items.
- Include short examples of terms or specific patterns, if they clarify the concept.
- Include any "Important tokens" explicitly in the description.
- Do not add labels, headings, or extra commentary.
- Be precise, avoid speculation, and avoid vague expressions like "in various contexts."
```

D EXTENDED RESULTS

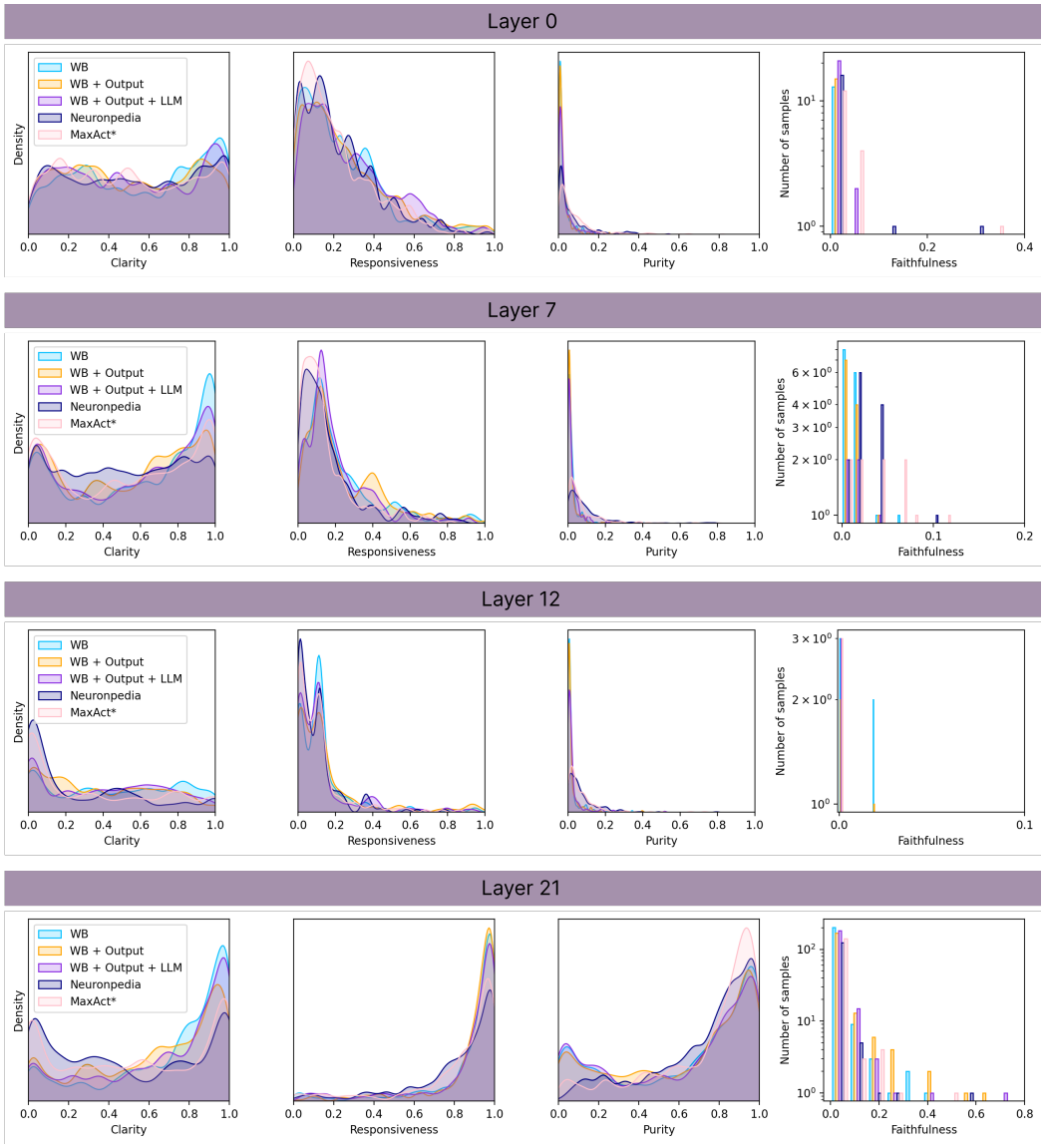


Figure 9: Kernel density estimates illustrating evaluation results of WeightLens methods in comparison to the baselines.

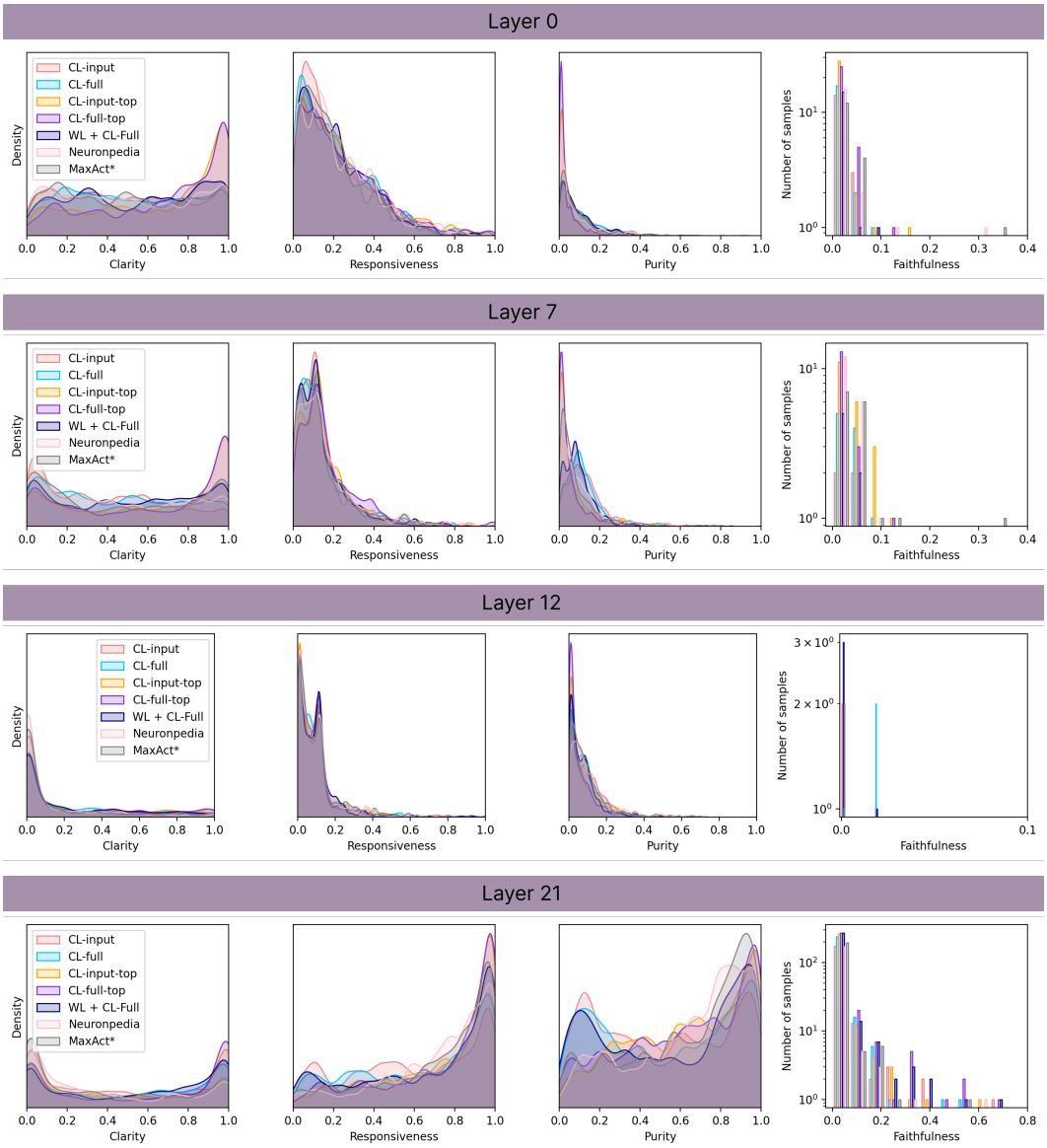


Figure 10: Kernel density estimates illustrating evaluation results of CircuitLens methods in comparison to the baselines.