WHAT ROLE DOES EXPLICIT CHAIN-OF-THOUGHT PLAY IN SAFETY MODERATION? A CIRCUIT-LEVEL STUDY

Anonymous authors
Paper under double-blind review

ABSTRACT

In safety moderation, explicit chain-of-thought (CoT) is often taken as evidence that a model is making careful and trustworthy judgments. Yet it remains unclear whether generating CoT actually helps form the final moderation decision, or whether it mainly verbalizes a decision that is already taking shape internally. To answer this question, we compare direct-answer and full-CoT prompting in a reasoning-based safety moderation model and trace their internal computation at the circuit level. Across multiple benchmarks, we find that the two prompting modes often share a substantial attributed backbone, while disagreement is concentrated in a smaller set of boundary-related features. We also find that much of the model's latent moderation prediction is already represented at intermediate layers, with later computation increasingly tied to expressing that preference at the output. These results suggest that the final label under full-CoT prompting does not typically emerge entirely from scratch during reasoning generation. Instead, even when direct-answer prompting produces a different final label, its internal state often already reflects support for the label produced under full-CoT prompting. Disagreement between the two modes arises mainly when the internal state under direct-answer prompting remains uncertain, which is also when explicit CoT appears to matter most. Based on this insight, we introduce an uncertainty-guided two-stage moderation pipeline that uses intermediate-layer sparse features for fast prediction and reserves full-CoT reasoning for uncertain cases only. The resulting system reduces reliance on expensive full-CoT inference while maintaining strong moderation performance.

1 INTRODUCTION

Chain-of-thought (CoT) reasoning has become a widely adopted technique for improving the performance of large language models (LLMs) on complex tasks (Wei et al., 2022; Zhou et al.; Chen et al., b; Guo et al., 2025; Ye et al., 2025). Beyond its empirical utility, CoT is often regarded as an interpretable window into model reasoning, as models that explain their judgments step by step are often perceived as reasoning more carefully and transparently than those that provide only direct answers. However, this interpretation has recently come under increasing scrutiny. A growing line of work suggests that explicit CoT may not faithfully reflect the actual internal computation underlying a model's prediction, and may instead function as a post-hoc justification or a surface-level generation pattern (Barez et al.; Chen et al., 2025). These findings raise a fundamental question: when LLMs produce chain-of-thought, are they genuinely reasoning through the problem, or merely verbalizing a decision that has already been formed internally?

In safety moderation tasks, this question is particularly important because CoT is often taken as a signal of reliable judgment (Liu et al.; Zheng et al.). When a moderation LLM produces a detailed rationale for why a piece of content should be allowed, flagged, or blocked, the explanation can appear to provide professional and policy-grounded support for the final decision. Such reasoning traces are therefore often treated not only as part of the model's output, but also as evidence that the model has made its decision in a careful, informed, and trustworthy manner (Yu et al., 2025; Hoover et al., 2025). If, however, these explanations do not play a central role in the underlying decision

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

process, then the apparent transparency offered by CoT may overstate how much insight it actually provides into the model's internal reasoning.

To better understand the role of explicit reasoning in safety moderation, we conduct a simple but revealing comparison between two inference settings for RSafe (Zheng et al.), a reasoning-based safety moderation model: one in which the model generates its full reasoning trace before giving a final label, and another in which it is prompted to output the label directly. As shown in Figure 1, we find that the predictions of the two settings are highly consistent across a wide range of cases. Although RSafe is trained via reinforcement learning to generate step-by-step reasoning before producing a final answer, it can often arrive at nearly the same verdict without explicitly generating the reasoning trace at inference time. This raises a central mechanistic question about CoT in safety moderation LLMs: **if explicit reasoning can be removed with little effect on the final decision, what role does the generated CoT actually play in the model's internal decision process?**

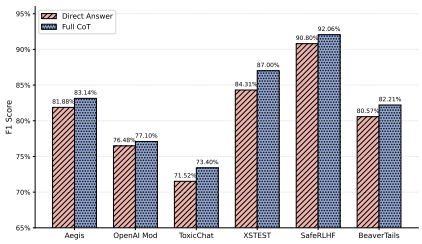


Figure 1: Comparison of F1 scores achieved by RSafe under Direct Answer and Full CoT prompting across six datasets.

While prior studies have questioned whether CoT faithfully reflects model reasoning, most have focused on behavioral outcomes rather than the underlying mechanisms (Tutek et al., 2025; Xiong et al.; Shen et al., 2025), and relatively little is known about how this question plays out in moderation models. To address this gap, we move beyond output-level comparisons and investigate the internal circuits that underlie decisions under direct-answer and full-CoT. Specifically, for each of four safety moderation datasets, we construct two types of cases: those in which the two prompting modes produce different labels, and those in which they agree. We then apply circuit tracing methods based on transcoders and edge attribution patching (EAP) to compare the internal features and pathways attributed in these cases. To further characterize where decision-relevant information comes from, we introduce a segment-level attribution analysis that separates contributions from the query, instruction, and generated CoT segments.

Our analysis reveals several consistent patterns. Direct-answer and full-CoT prompting share a substantial attributed internal backbone across both agreement and disagreement cases. We further find that much of the model's latent moderation preference is already represented at intermediate layers, while later computation increasingly shifts toward expressing the favored label at the output. Disagreement between prompting modes is concentrated in a small set of boundary-related features. Overall, these results suggest that the final moderation label under full-CoT prompting does not typically appear to emerge entirely from scratch during reasoning generation. Instead, direct-answer prompting often already reflects a latent decision state aligned with the eventual full-CoT label, although the two modes may sometimes differ in how they resolve difficult boundary cases. Explicit CoT appears to matter most in difficult boundary cases, where the model's internal state under direct-answer prompting remains uncertain. Building on this insight, we introduce an uncertainty-guided two-stage moderation pipeline. It first reads out the model's latent moderation preference from intermediate-layer sparse features, and invokes full-CoT reasoning only for uncertain cases. This substantially reduces reliance on expensive full-CoT inference while preserving strong performance.

Our contributions are threefold. First, we study an important but underexplored issue in safety moderation, with direct implications for how explicit CoT should be treated as evidence of careful and trustworthy moderation. Second, we provide a mechanistic analysis of reasoning-based safety moderation models to understand how moderation-relevant information is represented and expressed under direct-answer and full-CoT prompting. Third, based on the resulting insights, we design an uncertainty-guided two-stage moderation pipeline that improves the efficiency of reasoning-based moderation by reserving expensive full-CoT inference for uncertain cases only.

2 Mechanistic Analysis

As shown in Figure 1, direct-answer and full-CoT modes often produce highly similar moderation judgments, even though only the latter explicitly generates a reasoning trace. This raises a central

mechanistic question: when the two prompting modes arrive at the same label, do they rely on the same internal computation, or does explicit CoT induce a different decision process that happens to reach the same outcome? To investigate this, we move from output-level comparison to circuit-level analysis, with the goal of tracing how moderation-relevant information is represented and propagated inside the model.

2.1 CIRCUIT TRACING

Our circuit tracing procedure follows the Circuit Tracer (Hanna et al., 2025) framework, which represents model computation as a sparse feature-level graph and attributes the influence of internal components on a chosen output target. In our setting, this requires two key ingredients. First, we replace each MLP sublayer with a transcoder (Dunefsky et al., 2024) so that moderation-relevant computation can be expressed in terms of sparse, interpretable features rather than raw neurons. Second, we use Edge Attribution Patching (EAP) (Nanda, 2023; Hanna et al.) to score directed connections between these features with respect to a contrastive moderation objective. We describe these two components below.

2.1.1 TRANSCODERS

Transformer MLP sublayers write additive updates into the residual stream, making them a natural locus for feature-level circuit analysis (Geva et al., 2021). However, native MLP neurons are a poor basis for this purpose (Cunningham et al., 2023). They are often polysemantic, and coherent semantic patterns are typically distributed across many neurons. As a result, tracing circuits directly over raw MLP activations yields nodes that are difficult to interpret. To obtain a more interpretable basis, we replace each MLP sublayer with a transcoder. A transcoder is a sparse feature model that approximates the input–output behavior of the original MLP using a larger and sparser feature dictionary. Concretely, given an MLP input $x \in \mathbb{R}^h$, the transcoder produces sparse feature activations $z(x)$ and reconstructs the MLP output as a weighted sum of learned output directions:

$$\mathrm{MLP}(x) \approx \sum_i z_i(x)\, v_i, \tag{1}$$

where $z_i(x)$ is the activation of feature i and $v_i \in \mathbb{R}^h$ is its corresponding decoder vector. This representation is especially useful for circuit tracing because each active feature can be interpreted as a sparse, computation-relevant unit.

Transcoder architecture. In our setting, each MLP sublayer is replaced by a transcoder defined over a wider sparse feature space. Let $x \in \mathbb{R}^{1\times h}$ denote the input to an MLP. The transcoder first maps x into a sparse feature activation vector and then reconstructs the output:

$$z_{\mathrm{TC}}(x) = \mathrm{TopK}_k(\mathrm{ReLU}(xW_{\mathrm{enc}} + b_{\mathrm{enc}}))\,, \qquad k = 32, \tag{2}$$

$$\mathrm{TC}(x) = z_{\mathrm{TC}}(x)W_{\mathrm{dec}} + b_{\mathrm{dec}}, \tag{3}$$

where $W_{\mathrm{enc}} \in \mathbb{R}^{h\times N}$ and $W_{\mathrm{dec}} \in \mathbb{R}^{N\times h}$ are the encoder and decoder matrices, respectively, and N denotes the number of transcoder features, which is typically much larger than the model dimension h. At each layer, the operator TopK_k retains the k largest entries of the ReLU activation vector and sets all remaining entries to zero, yielding at most k nonzero feature activations per token.

Each transcoder feature $i \in \{1, \ldots, N\}$ is characterized by an encoder vector $e_i \in \mathbb{R}^{h\times 1}$ and a decoder vector $d_i \in \mathbb{R}^{1\times h}$. The encoder vectors form the columns of W_{enc}, while the decoder vectors form the rows of W_{dec}. For a given input x, the activation of feature i before TopK sparsification is computed as

$$a_i = \mathrm{ReLU}(xe_i + b_i), \tag{4}$$

where b_i is the i-th entry of b_{enc}. This activation measures how strongly the input matches that feature. After TopK sparsification, $[z_{\mathrm{TC}}(x)]_i$ equals a_i if feature i is retained and is zero otherwise. The full transcoder output can therefore be understood as a sum of the retained feature contributions plus the decoder bias, where each retained feature writes its decoder direction back into the residual stream, weighted by its activation strength.

Training objective. We train transcoders on a corpus of unlabeled text by minimizing reconstruction error between the original MLP output and the transcoder output, together with a sparsity penalty:

$$\mathcal{L}_{\text{trans}}^{(l)} = \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|m^{(l)}(x) - \hat{m}^{(l)}(x)\right\|_2^2\right] + \lambda_{\text{sparse}}\,\mathbb{E}_{x\sim\mathcal{D}}\left[\|z^{(l)}(x)\|_1\right]. \tag{5}$$

In practice, we train one transcoder per layer using activations collected from the frozen base model. After training, each original MLP block $m^{(l)}$ is replaced by its transcoder approximation $\hat{m}^{(l)}$.

2.1.2 Edge Attribution Patching

We use Edge Attribution Patching (EAP) to estimate the contribution of a directed edge in the traced circuit. In our attribution graph, a node is indexed by a layer–feature–token triple (l, i, t), where l is the layer index, i the transcoder feature index, and t the token position. Because these features form a sparse decomposition of MLP computation, they provide a substantially finer and more interpretable basis for circuit tracing than raw residual-stream activations or native MLP neurons. Standard EAP compares a clean run with a corrupted run, scoring an edge by the change in its source activation times the gradient of the attribution objective with respect to the destination input. Formally, for an edge $e = (u, v)$ with clean activation z_u, corrupted activation z'_u, and scalar objective $\mathcal{L}$,

$$\text{EAP}(e) = (z_u - z'_u)^\top \nabla_v \mathcal{L}. \tag{6}$$

In moderation tasks, however, constructing large numbers of semantically valid corrupted inputs is difficult, since small textual changes can alter the safety label or introduce unrealistic artifacts. We therefore adopt a zero baseline, setting $z'_u = 0$, which yields

$$\text{EAP}(e) = z_u^\top \nabla_v \mathcal{L}. \tag{7}$$

This score gives the first-order effect of the observed source activation relative to removing that contribution entirely. We use its magnitude to rank feature-to-feature and feature-to-output edges.

A second key choice in EAP is the attribution objective $\mathcal{L}$. Existing implementations often backpropagate from automatically selected salient logits, which is useful for open-ended generation but not well suited to moderation, where the task is inherently contrastive. We therefore define $\mathcal{L}$ using the moderation label contrast. Let $y^* \in \{\texttt{safe}, \texttt{unsafe}\}$ denote the target label and let $\tilde{y}$ denote the competing label. Accordingly, the attribution objective is defined as

$$\mathcal{L} = \ell(y^*) - \ell(\tilde{y}), \tag{8}$$

where $\ell(y)$ is the final logit assigned to token y at the target prediction position. Equivalently, if $u(y)$ denotes the demeaned unembedding direction for token y, then the target can be represented as the contrastive output direction

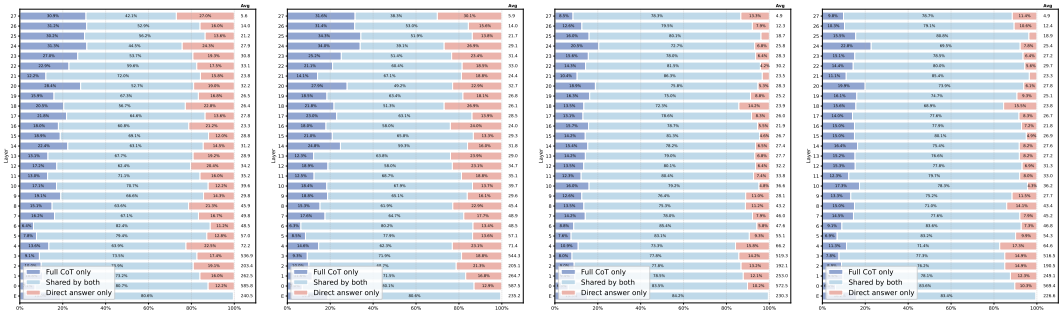
$$v_{\text{target}} = u(y^*) - u(\tilde{y}). \tag{9}$$

This focuses attribution on the internal computation that drives the model toward one moderation label rather than the other. Additional implementation details are provided in Appendix E.

2.2 Findings and Insights

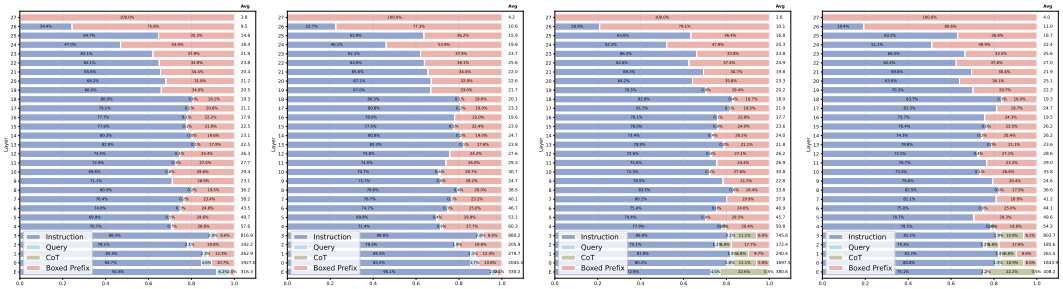
To better understand how RSafe arrives at moderation decisions, we conduct a mechanistic circuit analysis in the prompt-moderation setting. Because circuit tracing is computationally expensive, we restrict this analysis to prompt moderation rather than also extending it to response moderation. Across these datasets, we compare direct-answer and full-CoT modes, localize moderation-relevant computation across prompt segments and layers, and identify attributed features associated with mode disagreement. From this analysis, we obtain the following findings and insights. The analyses in this section focus on RSafe; additional circuit-level analyses of the SFT-based GuardReasoner-8B are provided in Appendix D.8.

Finding ❶ *Direct-answer and full-CoT prompting share a substantial attributed-feature backbone, while disagreement cases show stronger mode-specific divergence.* A natural first question after obtaining the circuit traces is whether direct-answer and full-CoT rely on the same internal



(a) Aegis (b) Xstest (c) Aegis (d) Xstest

Figure 2: Layer-wise overlap of attributed sparse features between direct-answer and full-CoT prompting, shown separately for disagreement cases (a, b) and agreement cases (c,d).



(a) Aegis (b) Xstest (c) Aegis (d) Xstest

Figure 3: Layer-wise distribution of attributed sparse features across sequence segments, shown separately for direct-answer prompting (a, b) and full-CoT prompting (c, d).

mechanism when they produce the same moderation label, and how this changes when they disagree. To answer this, we compare the layer-wise overlap of attributed sparse features between the two prompting modes, separating cases where they yield the same final label from cases where they yield different labels. Specifically, for each layer, we compute the proportions of attributed sparse features that are shared by both prompting modes, unique to direct answer, or unique to full-CoT.

As shown in Figure 2, in both agreement and disagreement cases, the majority of attributed sparse features are shared between direct-answer and full-CoT prompting at most layers. This indicates that the two prompting modes generally rely on a substantial common internal backbone, rather than invoking entirely separate circuits. At the same time, both modes also retain a smaller set of mode-specific features, showing that the computation is not identical even when the final label is the same. Even in disagreement cases, shared attributed sparse features still make up the largest proportion at most layers, indicating that the common backbone remains dominant despite the final label difference. The main difference is therefore relative rather than absolute: compared with agreement cases, disagreement cases exhibit a smaller shared proportion and a larger mode-specific proportion, especially in the higher layers. A plausible interpretation is that later layers, being closer to the final output token, are more involved in mode-dependent realization of the selected label and thus more likely to reflect direct-answer and full-CoT specific differences. Additional feature-overlap results on OpenAI Moderation and Toxic-Chat and an analysis of inter-layer information-flow overlap are provided in Appendices D.1 and D.2, respectively.

Insight ❶ These results suggest that direct-answer and full-CoT prompting do not typically implement moderation through wholly separate mechanisms. Instead, they usually share a substantial attributed-feature backbone, and disagreement appears to arise when the model diverges more strongly from this shared computation in a smaller set of boundary cases.

Finding ❷ *Safety moderation circuits exhibit a functional separation between label selection and label copy.* We also observe that the total number of attributed sparse features is much higher in the earliest layers than in later ones. This suggests that early computation contains richer signals, whereas later computation becomes more compact and specialized. This naturally raises a further

Table 1: Prediction flip rates under feature-negation interventions in RSafe on XSTEST.

Target	Intervention	N	Flip rate	Safe→Unsafe	Unsafe→Safe
Early (label-selection)	Catalog	450	**69.3%**	211/213 (99.1%)	101/237 (42.6%)
	Random ×5	450	15.6%	14/213 (6.6%)	56/237 (23.6%)
Mid (label-copy)	Catalog	450	**90.0%**	209/213 (98.1%)	196/237 (82.7%)
	Random ×5	450	2.2%	6/213 (2.8%)	4/237 (1.7%)

Table 2: Top attributed sparse features most strongly associated with disagreement between direct-answer and full-CoT prompting. Features are ranked by a disagreement-association score that combines the frequency gap, enrichment ratio, and statistical significance.

Rank	Feature ID	Disagree%	Agree%	Diff (pp)	Ratio	p-value	Disagree count	Agree count
1	24_21601	26.4%	1.7%	+24.7	8.1	1.30×10^{-22}	64	7
2	26_5897	20.7%	1.5%	+19.2	7.3	4.40×10^{-17}	50	6
3	22_90685	14.5%	1.2%	+13.2	6.0	1.93×10^{-11}	35	5

question: where across the sequence are attribution-relevant computations concentrated, and does the dominant role of different prompt segments shift across layers? To answer this, we localize attributed features across sequence segments. We assign each feature node to its token's segment and compute the segment proportions at each layer. Direct-Answer inputs contain instruction, query, and boxed-answer-prefix segments; Full-CoT inputs additionally contain the reasoning trace.

Figure 3 shows that attributed features on the query and CoT segments are more prevalent in early layers and become rare from the middle layers onward. Instruction features account for a large share across layers, while the boxed-answer prefix generally gains prominence toward the output. To better understand what the instruction and boxed-prefix dominated pattern is doing, we further inspect where these attributed sparse features are located. On the instruction side, the most prominent features are concentrated around the explicit label tokens `safe` and `unsafe`, rather than being distributed broadly across the instruction. On the output side, attributed sparse features on the boxed-answer segment are concentrated near the token immediately preceding the final label, suggesting that this region is tightly coupled to actually emitting the answer token.

These localization patterns motivate a functional distinction between two interacting subcircuits. We hypothesize that *label selection* integrates evidence from the query and, when present, the CoT to favor a moderation label, while *label copy* routes label information from the instruction toward the output. The boxed-answer prefix serves as the realization site for the selected label. An illustrative case is provided in Appendix D.3.

Causal validation. Guided by the segment-level attribution patterns, we target query-token features in early layers (L0–7) as candidates for label selection and instruction-token features in middle layers (L8–16) as candidates for label copy. This choice reflects the early concentration of input-related features and the subsequent localization around instruction label tokens. We perform separate interventions in RSafe's Direct-Answer mode on XSTEST ($N = 450$). At each layer, we construct a catalog of the attributed transcoder features ranked by attribution frequency across the four Direct-Answer circuit datasets. For each sample, we reverse the sign of nonzero catalog activations ($a \mapsto -a$) only at query positions in L0–7 or only at instruction positions in L8–16. The random control applies the same sign reversal to five times as many non-catalog features per layer as the corresponding catalog intervention. We measure prediction flips relative to the unmodified model. Table 1 shows that catalog interventions produce substantially higher flip rates than the random controls for both early query features (69.3% versus 15.6%) and middle-layer instruction features (90.0% versus 2.2%). These results establish a causal influence of the targeted feature sets on the final prediction, complementing the localization evidence for the proposed label-selection and label-copy components.

Insight ❷ These results suggest that safety moderation does not rely on a single undifferentiated circuit. Instead, the model appears to separate the computation of which label should be chosen from the mechanism that copies and realizes that label at the output position.

Finding ❸ *Disagreement between direct-answer and full-CoT is associated with a small set of attributed sparse features.* Following Findings ❶ and ❷, if most computation is shared across the two modes, and later layers are increasingly involved in expressing rather than newly forming the final label, then disagreement is unlikely to mean that direct-answer and full-CoT follow two completely different circuits from start to finish. A more plausible interpretation is that the two modes usually rely on the same backbone, but diverge on a smaller set of difficult or weakly resolved cases, where the model's internal decision state is not yet fully stable. This naturally motivates asking whether there are specific attributed sparse features that are especially associated with disagreement cases. To identify disagreement-related features, we aggregate agreement and disagreement cases across all four prompt-moderation datasets and compare attributed sparse feature frequencies between the two groups. We rank features using a disagreement-association score that combines the relative frequency gap, the ratio of counts, and statistical significance. As shown in Table 2, three features stand out strongly under this criterion, each showing a large enrichment in disagreement cases. Although only three features satisfy this criterion, together they cover 39.6% of all disagreement samples. This suggests that disagreement is concentrated in a small number of prompting-sensitive attributed features rather than spread across many unrelated states. We further examine the semantic interpretations of disagreement-associated features and features enriched in agreement cases in Appendices D.4 and D.5, respectively.

Causal validation. To test whether CoT-only features contribute to mode-dependent decisions, we evaluate examples from Aegis and OpenAI Moderation on which RSafe's Direct-Answer and Full-CoT predictions disagree. For each sample, CoT-only features are attributed in the Full-CoT circuit but absent from the Direct-Answer circuit at the same layer. We negate only these feature activations during Full-CoT inference and measure label changes relative to the unmodified Full-CoT prediction. In Table 3, N is the number of evaluated disagreement examples, Flips counts changed predictions, and Flip Rate is Flips divided by N. Negating CoT-only features flips 89.1% of predictions on Aegis and 92.9% on OpenAI Moderation. These results support a causal contribution of mode-specific features to the differing predictions under Direct-Answer and Full-CoT prompting.

Table 3: Prediction flips after negating CoT-only features on disagreement examples.

Dataset	N	Flips	Flip Rate
Aegis	92	82	89.1%
OpenAI Mod.	42	39	92.9%

Insight ❸ Prompting disagreement is best understood as a boundary phenomenon rather than evidence of two fundamentally different moderation circuits. Direct-answer and full-CoT prompting usually share the same decision-relevant backbone, but can resolve a small set of weakly specified internal states differently.

3 Application: Uncertainty-Guided Two-Stage Moderation

Motivation. Our circuit analysis suggests that direct-answer and full-CoT prompting often rely on a shared internal backbone, while late-layer activity becomes increasingly concentrated on the boxed-answer prefix region. Moreover, classification-relevant features identified by our attribution analysis appear prominently on the boxed-answer prefix, suggesting that late-stage computation is strongly tied to expressing the selected label at the output position. Together, these findings suggest that the model's eventual moderation preference is often reflected in its internal state before the final layers, with later computation serving primarily to realize that preference in the required output format. This observation leads to a practical hypothesis: *can intermediate sparse features already support accurate readout of the model's eventual moderation decision?* To test this hypothesis, we train simple feature-based classifiers on Toxic-Chat (Lin et al., 2023) and BeaverTails (Ji et al., 2023) using sparse activations from different layers in direct-answer mode and evaluate them on multiple moderation benchmarks (Ghosh et al., 2024; Markov et al., 2023; Röttger et al., 2024). Figure 4 shows that performance consistently peaks

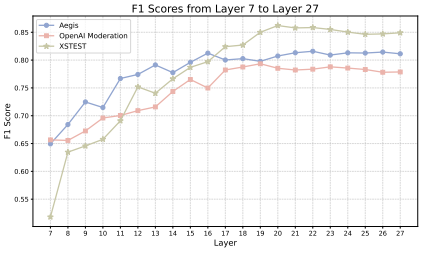


Figure 4: Layer-wise F1 of sparse-feature readouts under direct-answer prompting.

around layers 20–22, with limited gains from later layers. This result suggests that an intermediate-layer readout can already capture much of the model's latent moderation state.

Building on this result, we introduce a simple mechanism-informed pipeline: an uncertainty-guided two-stage moderation pipeline, which we call UTM. We use layer 22 as the readout layer, since it consistently gives near-peak performance. This choice is also computationally advantageous. Under direct-answer prompting, once the sparse features at layer 22 are obtained, the computation beyond layer 22 can be replaced by two lightweight two-layer MLPs: one for computing downstream feature activations, and the other for final classification. As a result, the model can handle many examples without executing the full late-stage transformer computation, while the predicted unsafe probability naturally serves as a confidence signal for deciding whether to fall back to full-CoT reasoning.

Stage 1: Intermediate-Layer Readout. In the first stage, we aim to read out the model's latent moderation state from intermediate sparse features under direct-answer prompting, rather than to learn a new moderation policy from dataset gold labels. Let l denote the selected readout layer. For each input, we run the transcoder-replaced model in direct-answer mode and extract sparse feature activations $z^{(t)} \in \mathbb{R}^{N_l}$ at layer l, where t indexes token positions and N_l is the number of sparse features at that layer. We aggregate activations from the query region onward. Let t_0 denote the starting position of the user query. For each feature $i \in \{1, \ldots, N_l\}$, we define an example-level pooled activation by

$$x_i = \max_{t \geq t_0} z_i^{(t)}. \tag{10}$$

Using these pooled feature values, we construct example-level sparse representations for both the training and test sets, and remove features that are never activated in the training data.

We train a lightweight classifier on the pooled sparse features to predict the model's direct-answer label, rather than the dataset gold label, since the goal of Stage 1 is to recover the model's own emerging decision state. Concretely, given the retained feature vector $\tilde{x} \in \mathbb{R}^{d_c}$, the classifier outputs an unsafe probability

$$p(\texttt{unsafe} \mid \tilde{x}) = \sigma(w^\top \tilde{x} + b), \tag{11}$$

where $w \in \mathbb{R}^{d_c}$ and $b \in \mathbb{R}$ are the classifier parameters. At inference time, we apply the same feature extraction and pooling procedure, compute $p(\texttt{unsafe} \mid \tilde{x})$, and obtain the Stage 1 prediction by thresholding this probability with τ_{cls}. The predicted unsafe probability is also used as the confidence signal for Stage 2.

Stage 2: Uncertainty-Guided Fallback to Full-CoT. We further introduce two confidence thresholds, τ_{safe} and τ_{unsafe}, which define an uncertainty band around the Stage 1 classification threshold τ_{cls}. A sample is treated as uncertain when its predicted unsafe probability is close to the decision boundary, i.e.,

$$\tau_{\text{safe}} < p(\texttt{unsafe} \mid \tilde{x}) < \tau_{\text{unsafe}}. \tag{12}$$

For samples outside this uncertainty band, we directly keep the Stage 1 prediction as the final output. For samples inside the band, we invoke full-CoT moderation and use the full-CoT label as the final prediction. In this way, confident examples are handled by the lightweight Stage 1 readout, while expensive full-CoT reasoning is reserved only for uncertain cases.

Results. We train the Stage 1 readout on Toxic-Chat (Lin et al., 2023) for prompt moderation and on BeaverTails (Ji et al., 2023) for response moderation, and evaluate on OpenAI Moderation (Markov et al., 2023), Aegis (Ghosh et al., 2024), and XSTEST (Röttger et al., 2024) for prompt moderation, as well as SafeRLHF (Ji et al., 2025) for response moderation. We evaluate UTM on two representative reasoning-based safety moderation LLMs with different training paradigms: RSafe (Zheng et al.), which is trained with reinforcement learning to produce proactive reasoning before moderation decisions, and GuardReasoner-8B (Liu et al.), which is trained with supervised fine-tuning and HDPO. This setup allows us to test whether the proposed two-stage design transfers across reasoning-based safeguard models beyond a single training recipe. In this sense, UTM is designed as a plug-and-play inference-time pipeline for reasoning-based moderation models, requiring only access to intermediate-layer sparse features and a lightweight readout classifier.

We compare against both closed-source APIs and open-source moderation models, including OpenAI Moderation (Markov et al., 2023), GPT-4o (OpenAI, 2025), ShieldGemma-9B (Zeng et al.,

2024), LlamaGuard3-8B (Inan et al., 2023), WildGuard-7B (Han et al., 2024), as well as the full-CoT and direct-answer variants of RSafe and GuardReasoner-8B. For the uncertainty-guided routing step, we use fixed confidence thresholds $\tau_{\text{safe}} = 0.05$ and $\tau_{\text{unsafe}} = 0.95$. Samples with predicted unsafe probability between these two thresholds are routed to Stage 2, while the remaining samples are handled directly by the Stage 1 readout. A detailed description of the baseline models and the datasets is provided in Appendix B and Appendix C, respectively.

As shown in Table 4, UTM supports an efficiency–performance trade-off on both RSafe and GuardReasoner-8B. Overall, the results suggest that intermediate-layer sparse features already capture much of a model's latent moderation preference, allowing many examples to be handled without running full-CoT inference. For RSafe, Stage 1 already achieves performance close to direct-answer prompting, with gains on OpenAI Moderation and XSTEST and only minor differences on Aegis and SafeRLHF. This suggests that intermediate sparse features at the selected readout layer encode much of the model's eventual moderation decision. The uncertainty-guided routing mechanism then reserves Stage 2 for only a small fraction of examples, indicating that the routed subset is concentrated on more difficult cases where the full-CoT mode is useful. As a result, most inputs can be resolved by the lightweight Stage 1 readout, while expensive full-CoT inference is invoked only when needed.

Table 4: F1 on four moderation benchmarks. For UTM Stage 2, the second line shows the number and percentage routed to Stage 2. Baselines on OpenAI Moderation, Aegis, and SafeRLHF are taken from RSafe; XSTEST results are from our evaluation.

	OAI	**Aegis**	**XST**	**SRLHF**
Closed-source APIs				
OAI Moder.	0.790	0.362	–	0.140
GPT-4o	0.493	0.462	–	0.464
Open-source models				
ShieldG.	**0.796**	0.686	0.825	0.486
LGuard3	<u>0.792</u>	0.761	0.882	0.890
WildGuard	0.649	0.809	<u>0.943</u>	**0.926**
GuardReasoner-8B				
Full CoT	0.725	**0.835**	**0.944**	0.917
Direct	0.723	0.811	0.934	0.891
UTM S1	0.690	0.811	0.929	0.867
UTM S1+2	0.724	0.821	0.937	0.905
	(305,18.2%)	(361,18.4%)	(78,17.3%)	(2227,13.6%)
RSafe				
Full CoT	0.771	<u>0.831</u>	0.870	<u>0.921</u>
Direct	0.765	0.819	0.843	0.908
UTM S1	0.784	0.816	0.856	0.893
UTM S1+2	<u>0.792</u>	0.820	0.864	0.915
	(93,5.6%)	(79,4.1%)	(19,4.2%)	(2445,14.9%)

We further evaluate the same mechanism-informed design on GuardReasoner-8B. In this setting, we do not perform a separate layer sweep; instead, we directly use layer 22, which was selected based on the layer-wise analysis of RSafe. Even under this untuned setting, Stage 1 still captures a substantial portion of GuardReasoner's moderation behavior, and the uncertainty-guided routing mechanism again selects a minority of examples for Stage 2. We therefore view the GuardReasoner results as evidence that the proposed two-stage design can transfer across reasoning-based moderation models, although its exact effectiveness likely depends on model-specific factors such as the optimal readout layer and uncertainty thresholds. To further assess inference efficiency and the effectiveness of intermediate-layer readouts, we report latency and throughput measurements in Appendix D.6 and comparisons with Logit Lens (Din et al., 2024) and Tuned Lens (Belrose et al., 2023) in Appendix D.7.

4 Conclusion

We investigate explicit CoT in safety moderation through circuit-level analysis of RSafe. Direct-Answer and Full-CoT prompting share a substantial attributed-feature backbone, with greater divergence in disagreement cases. GuardReasoner-8B shows a similar divergence pattern but different segment-level attribution. In RSafe, feature localization suggests a distinction between label selection and label copy, while targeted interventions establish the causal influence of early query and middle-layer instruction features. Disagreement-associated features and CoT-only interventions further connect mode-specific computation to prediction changes. These findings motivate intermediate-layer readout and selective CoT inference, which we test through layer-wise readout experiments and an uncertainty-guided two-stage pipeline. The readout experiments demonstrate that moderation predictions can be recovered before the final layers. The pipeline maintains strong performance across evaluated benchmarks while routing only a minority of inputs to Full-CoT, supporting the practical value of the mechanistic observations. Limitations are discussed in Appendix F.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing, LaTeX formatting, manuscript organization, and discussion of how to present and interpret experimental results. All AI-assisted text and suggestions were reviewed by the authors for accuracy and consistency with the experimental evidence. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work has both potential benefits and potential risks. On the positive side, it improves our understanding of how reasoning-based safety moderation models internally form and express moderation decisions, which may help develop more transparent, efficient, and trustworthy safeguards. Our uncertainty-guided two-stage pipeline also shows how mechanistic insights can reduce reliance on expensive full-CoT inference while maintaining strong moderation performance.

On the negative side, analyzing the internal mechanisms of safety moderation models may also help malicious users understand weaknesses in current safeguard systems, especially around uncertain or boundary-case behavior. Such insights could potentially be used to craft inputs that evade moderation. We therefore view this work as defensive safety research aimed at improving safeguards, and we do not release new attack methods or high-risk model capabilities as part of this paper.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe our circuit-tracing methodology and two-stage moderation pipeline in Sections 2 and 3. Appendix E provides computational resources, software dependencies, prompting and decoding settings, transcoder training configurations, and attribution graph selection criteria.

REFERENCES

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, et al. Chain-of-thought is not explainability.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Jianhui Chen, Xiaozhi Wang, Zijun Yao, Yushi Bai, Lei Hou, and Juanzi Li. Towards understanding safety alignment: A mechanistic perspective from safety neurons. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, a.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, b.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

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.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. Jump to conclusions: Short-cutting transformers with linear transformations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 9615–9625, 2024.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable llm feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

Junfeng Fang, Nachuan Chen, Houcheng Jiang, Dan Zhang, Fei Shen, Xiang Wang, Xiangnan He, and Tat-Seng Chua. Next-guard: Training-free streaming safeguard without token-level labels. *arXiv preprint arXiv:2603.02219*, 2026.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5484–5495, 2021.

Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. Aegis: Online adaptive ai content safety moderation with ensemble of llm experts. *arXiv preprint arXiv:2404.05993*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.

Michael Hanna, Mateusz Piotrowski, Jack Lindsey, and Emmanuel Ameisen. Circuit-tracer: A new library for finding feature circuits. In *Proceedings of the 8th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 239–249, 2025.

Monte Hoover, Vatsal Baherwani, Neel Jain, Khalid Saifullah, Joseph Vincent, Chirag Jain, Melissa Kazemi Rad, C Bayan Bruss, Ashwinee Panda, and Tom Goldstein. Dynaguard: A dynamic guardian model with user-defined policies. *arXiv preprint arXiv:2509.02563*, 2025.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36:24678–24704, 2023.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, et al. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31983–32016, 2025.

Difan Jiao, Yilun Liu, Ye Yuan, Zhenwei Tang, Linfeng Du, Haolun Wu, and Ashton Anderson. Llm safety from within: Detecting harmful content with internal representations. *arXiv preprint arXiv:2604.18519*, 2026.

Seongmin Lee, Aeree Cho, Grace C Kim, ShengYun Peng, Mansi Phute, and Duen Horng Chau. Interpretation meets safety: A survey on interpretation methods and tools for improving llm safety. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21514–21545, 2025.

Zhongzhi Li, Xuansheng Wu, Yijiang Li, Lijie Hu, and Ninghao Liu. Less is enough: Synthesizing diverse data in feature space of llms. *arXiv preprint arXiv:2602.10388*, 2026.

Junyu Lin, Meizhen Liu, Xiufeng Huang, Jinfeng Li, Haiwen Hong, Xiaohan Yuan, Yuefeng Chen, Longtao Huang, Hui Xue, Ranjie Duan, et al. Yufeng-xguard: A reasoning-centric, interpretable, and flexible guardrail model for large language models. *arXiv preprint arXiv:2601.15588*, 2026.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4694–4702, 2023.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Jun Xia, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. Guardreasoner: Towards reasoning-based llm safeguards. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 15009–15018, 2023.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*.

Neel Nanda. Attribution patching: Activation patching at industrial scale. *URL: https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*, 15:19, 2023.

OpenAI. Hello gpt-4o. *URL: https://openai.com/zh-Hans-CN/index/hello-gpt-4o/*, 2025.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5377–5400, 2024.

Xu Shen, Song Wang, Zhen Tan, Laura Yao, Xinyu Zhao, Kaidi Xu, Xin Wang, and Tianlong Chen. Faithcot-bench: Benchmarking instance-level faithfulness of chain-of-thought reasoning. *arXiv preprint arXiv:2510.04040*, 2025.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasović, and Yonatan Belinkov. Measuring chain of thought faithfulness by unlearning reasoning steps. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9946–9971, 2025.

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.

Xu Wang, Yan Hu, Wenyu Du, Reynold Cheng, Benyou Wang, and Difan Zou. Towards understanding fine-tuning mechanisms of llms via circuit analysis. In *International Conference on Machine Learning*, pp. 63088–63112. PMLR, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jiaqi Weng, Han Zheng, Hanyu Zhang, Qinqin He, Jialing Tao, Hui Xue, Zhixuan Chu, and Xiting Wang. Safe-sail: Towards a fine-grained safety landscape of large language models via sparse autoencoder interpretation framework. *arXiv preprint arXiv:2509.18127*, 2025.

Zidi Xiong, Shan Chen, Zhenting Qi, and Himabindu Lakkaraju. Measuring the faithfulness of thinking drafts in large reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Jingcheng Yang, Tianhu Xiong, Shengyi Qian, Klara Nahrstedt, and Mingyuan Wu. Circuit tracing in vision-language models: Understanding the internal mechanisms of multimodal thinking. *arXiv preprint arXiv:2602.20330*, 2026.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. In *Conference on Language Modeling*, 2025.

Shaohan Yu, Lijun Li, Chenyang Si, Lu Sheng, and Jing Shao. Proguard: Towards proactive multimodal safeguard. *arXiv preprint arXiv:2512.23573*, 2025.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, et al. Shieldgemma: Generative ai content moderation based on gemma. *arXiv preprint arXiv:2407.21772*, 2024.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025a.

Zheng Zhao, Yeskendir Koishekenov, Xianjun Yang, Naila Murray, and Nicola Cancedda. Verifying chain-of-thought reasoning via its computational graph. *arXiv preprint arXiv:2510.09312*, 2025b.

Jingnan Zheng, Xiangtian Ji, Yijun Lu, Chenhang Cui, Weixiang Zhao, Gelei Deng, Zhenkai Liang, An Zhang, and Tat-Seng Chua. Rsafe: Incentivizing proactive reasoning to build robust and adaptive llm safeguards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Related Work

A.1 Mechanistic Interpretability of LLMs

Mechanistic interpretability seeks to understand LLM behavior through internal representations and computational circuits. One important direction uses sparse and interpretable feature spaces. Sparse autoencoders (SAEs) have been shown to recover interpretable latent features from model activations (Cunningham et al., 2023), and related work has applied feature-space methods to data synthesis, safety analysis, and harmful-content detection (Li et al., 2026; Weng et al., 2025; Jiao et al., 2026). A complementary direction studies model behavior as circuits. Prior work has identified interpretable circuits for specific behaviors in language models (Wang et al., 2022; Hanna et al., 2023), while later work has developed more automated and feature-level approaches, including attribution-based circuit discovery, sparse feature circuits, transcoders, and tracing frameworks such as Circuit Tracer (Conmy et al., 2023; Marks et al.; Dunefsky et al., 2024; Hanna et al., 2025). Attribution methods such as Edge Attribution Patching (EAP) (Nanda, 2023) further enable efficient analysis of directed interactions, and have been extended to study faithfulness, fine-tuning, chain-of-thought verification, and multimodal reasoning (Hanna et al.; Wang et al., 2025; Zhao et al., 2025b; Yang et al., 2026). Interpretability has also been increasingly applied to safety. Prior work has studied refusal directions, safety neurons, and broader interpretability tools for safer LLM behavior (Lee et al., 2025; Arditi et al., 2024; Chen et al., a).

A.2 LLMs for Safety Moderation

LLM-based safety moderation has rapidly evolved from direct classification-style guard models to more flexible and reasoning-oriented safeguard systems. Early and widely used guardrail models such as ShieldGemma (Zeng et al., 2024), Llama Guard (Inan et al., 2023), and WildGuard (Han et al., 2024) frame moderation as an input/output classification problem, typically predicting whether content is safe or unsafe under a predefined policy. These systems provide strong practical baselines, but they usually do not expose explicit reasoning traces as part of the moderation process. More recent work has explored *reasoning-based safeguards*, motivated by the idea that explicit deliberation may improve robustness, adaptivity, and interpretability in safety-critical settings. RSafe (Zheng et al.) and GuardReasoner (Liu et al.) train moderation models to generate proactive reasoning before producing a final decision, while YuFeng-XGuard (Lin et al., 2026), Qwen3Guard (Zhao et al., 2025a) similarly emphasize step-by-step moderation and policy-grounded justification. Other recent safeguard systems explore broader settings, including multimodal moderation (Yu et al., 2025), dynamic policy-conditioned moderation (Hoover et al., 2025), and lightweight or streaming moderation pipelines (Fang et al., 2026).

B Baseline Models

We compare the proposed pipeline against both closed-source APIs and open-source moderation models. These baselines cover direct classification-style moderation systems as well as reasoning-based safeguards.

OpenAI Moderation. OpenAI Moderation (Markov et al., 2023) is a widely used proprietary moderation system for detecting unsafe or policy-violating content. It serves as a strong closed-source baseline for prompt- and response-level safety evaluation.

GPT-4o. GPT-4o (OpenAI, 2025) is not a dedicated moderation model, but it can be used as a general-purpose judge through prompting. We include it as a closed-source baseline to reflect the practical alternative of using a frontier instruction-following model for zero-shot or prompt-based safety classification.

ShieldGemma-9B. ShieldGemma-9B (Zeng et al., 2024) is an open-source moderation model built on Gemma and trained specifically for safety classification. It is designed as a lightweight but strong moderation baseline and is widely used in comparisons of open safety guard models.

LlamaGuard3-8B. LlamaGuard3-8B (Inan et al., 2023) is an open moderation model developed for monitoring both user inputs and model outputs. It is one of the most widely adopted open-source guard models and provides a strong baseline for practical moderation performance.

WildGuard-7B. WildGuard-7B (Han et al., 2024) is an open-source moderation model designed to cover multiple safety-related tasks, including harmful prompt detection, harmful response detection, and refusal-related behaviors. We include it as a competitive open baseline with strong coverage of modern moderation benchmarks.

GuardReasoner-8B. GuardReasoner-8B (Liu et al.) is a reasoning-based safeguard model that explicitly incorporates multi-step reasoning into the moderation process. It is especially relevant to our work because it represents an alternative reasoning-oriented moderation approach.

RSafe variants. Since our study focuses on the role of explicit chain-of-thought in reasoning-based moderation, we also compare against two inference variants of RSafe (Zheng et al.): *RSafe Full CoT*, which generates a full reasoning trace before outputting the final label, and *RSafe Direct Answer*, which is prompted to produce the moderation label directly. These two variants provide the core comparison setting for both our mechanistic analysis and our application study.

C Dataset Details

We use several public benchmarks covering both prompt moderation and response moderation. Below we briefly summarize the role and characteristics of each dataset used in this work.

Toxic-Chat. Toxic-Chat (Lin et al., 2023) is a prompt-level moderation benchmark built from real user interactions with an open-source chatbot system. Compared with synthetic or highly curated safety datasets, it better reflects realistic and often noisy user requests, including subtle or adversarial toxic prompts. We use Toxic-Chat as the training source for the Stage 1 readout in the prompt-moderation setting and as one of the evaluation benchmarks in our direct-answer versus full-CoT comparison.

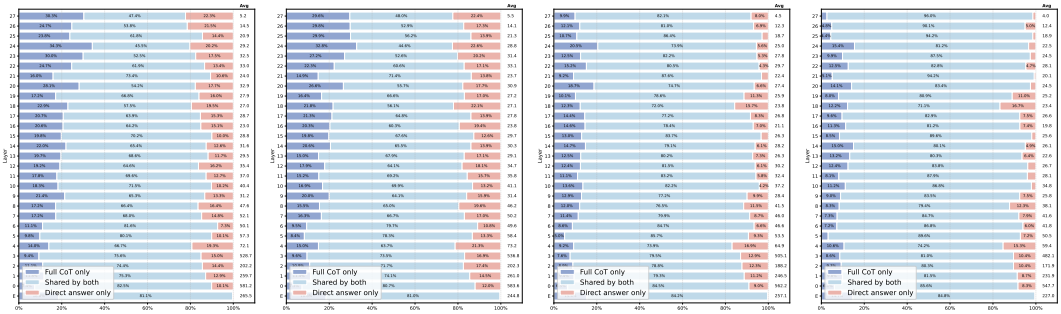
OpenAI Moderation. The OpenAI Moderation dataset (Markov et al., 2023) contains prompts annotated for multiple categories of harmful or policy-violating content. It is widely used as a benchmark for prompt-level content safety classification. In this work, we use it as one of the prompt-moderation evaluation datasets for comparing direct-answer prompting, full-CoT prompting, and the proposed two-stage moderation pipeline.

Aegis. Aegis (Ghosh et al., 2024) is a safety moderation benchmark designed to cover a broader range of policy-sensitive and borderline cases. Its annotations reflect a richer safety taxonomy than simple binary toxicity detection, making it useful for evaluating whether moderation systems can handle nuanced policy boundaries. We include Aegis both in our mechanistic analysis and in our downstream moderation evaluation.

XSTEST. XSTEST (Röttger et al., 2024) is a benchmark for evaluating exaggerated or overcautious safety behavior in language models. It contains many prompts that are superficially similar to unsafe content but are actually benign, making it particularly useful for studying difficult boundary cases. In our setting, XSTEST is important because it helps reveal when direct-answer and full-CoT prompting may diverge on weakly specified or borderline examples.

BeaverTails. BeaverTails (Ji et al., 2023) is a large-scale safety dataset containing question–answer pairs and preference annotations related to harmfulness and harmlessness. Since it contains both prompts and model responses, it is suitable for response-moderation experiments. In our work, we use BeaverTails as the training source for the Stage 1 readout in the response-moderation setting.

SafeRLHF. SafeRLHF (Ji et al., 2025) is a response-level safety benchmark constructed to support multi-level safety alignment of language models. It contains prompts and model responses annotated for safety, and is widely used to evaluate whether a model can identify unsafe generated content.



(a) Openai-moderation (b) Toxic-chat (c) Openai-moderation (d) Toxic-chat

Figure 5: Layer-wise overlap of attributed sparse features between direct-answer and full-CoT prompting, shown separately for disagreement cases (a, b) and agreement cases (c, d).

We use SafeRLHF as the main response-moderation evaluation benchmark for testing whether the proposed UTM pipeline transfers beyond the prompt-moderation setting.

Use in this paper. Our mechanistic circuit analysis is conducted on the prompt-moderation setting, using Aegis, OpenAI Moderation, Toxic-Chat, and XSTEST. For the application study, we train the Stage 1 sparse-feature readout on Toxic-Chat for prompt moderation and on BeaverTails for response moderation, and evaluate on OpenAI Moderation, Aegis, XSTEST, and SafeRLHF.

C.1 Circuit Counts by Dataset and Agreement Status

For the circuit-based analyses in this paper, we group examples according to whether direct-answer prompting and full-CoT prompting produce the same moderation label (*same*) or different labels (*different*). Table 5 reports the number of traced circuits in each group for the four datasets used in our experiments. The *different* group includes all available disagreement cases used for tracing, while the *same* group is obtained by random sampling from the full set of agreement cases. We observe that the number of same-label cases reported here is larger than or comparable to the number of different-label cases across datasets, consistent with the overall tendency of the two prompting modes to agree on most examples.

These counts provide context for the additional circuit analyses in the appendix, including the node-overlap, edge-overlap, and case-study visualizations. In particular, they indicate how many examples contribute to the agreement and disagreement groups on each dataset, and therefore help interpret the relative stability of the estimated circuit patterns across benchmarks.

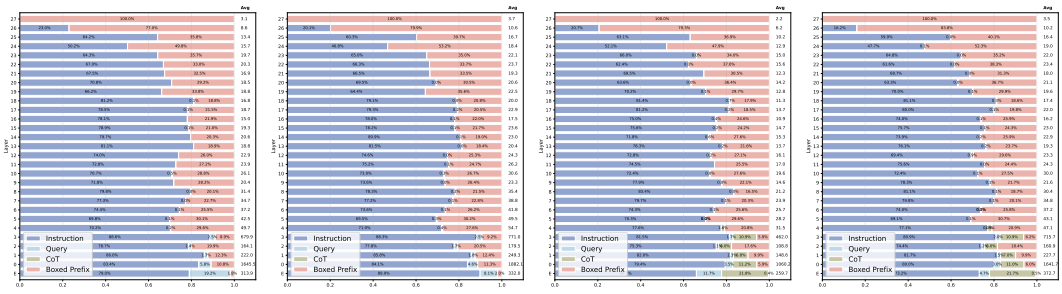
Table 5: Number of traced circuits by dataset and by whether direct-answer and full-CoT prompting produce different or the same moderation label.

Dataset	Different	Same
Aegis	92	105
OpenAI Moderation	42	70
ToxicChat	57	113
XSTEST	51	117

D Additional Results

D.1 Additional Feature-Overlap and Segment-Level Results

Figures 5 and 6 extend the feature-overlap and segment-level analyses in Findings ❶ and ❷ to OpenAI Moderation and Toxic-Chat, respectively showing overlap between prompting modes and the distribution of attributed features across sequence segments.



(a) Openai-moderation (b) Toxic-chat (c) Openai-moderation (d) Toxic-chat

Figure 6: Layer-wise distribution of attributed sparse features across sequence segments, shown separately for direct-answer prompting (a, b) and full-CoT prompting (c, d).

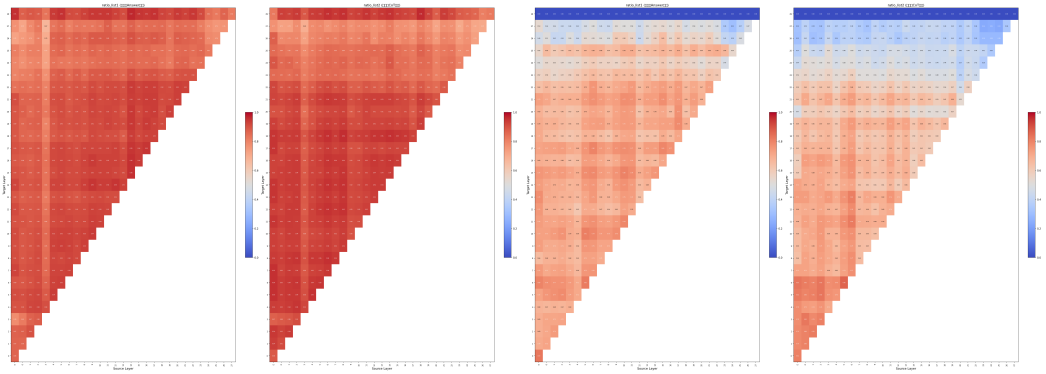


Figure 7: Layer-wise overlap ratios of traced inter-layer edges on Aegis, shown for agreement cases (left two panels) and disagreement cases (right two panels). Warmer colors indicate higher overlap for each source–target layer pair. Agreement cases show greater overlap in information-flow patterns than disagreement cases.

D.2 Information-Flow Overlap

In addition to the node-level overlap analysis reported in the main paper, we also study overlap at the level of inter-layer edges. This provides a complementary view of whether the two prompting modes rely on similar information-flow patterns, rather than merely activating similar sets of features.

Setup. For each example in the Aegis dataset, we extract the traced circuit under the direct-answer prompting setting and under the full-CoT prompting setting. We then compute, for each pair of source and target layers, the overlap ratio between the sets of traced edges in the two circuits. We group examples according to whether the two prompting modes produce the same final moderation label or different labels, and visualize the resulting layer-pair overlap matrices.

Results. Figure 7 shows the edge-overlap patterns on Aegis. When the two prompting modes arrive at the same final answer, the overlap in inter-layer edges is consistently high across most layer pairs. This suggests that, in agreement cases, the two prompting modes not only activate similar internal features, but also rely on broadly similar information-flow pathways through the model.

By contrast, when the two prompting modes produce different final answers, the overlap in inter-layer edges drops substantially, especially in the middle-to-late layers where moderation-relevant computation is most concentrated. This indicates that disagreement between prompting modes is associated not only with different node usage, but also with different patterns of communication between layers.

Interpretation. These results strengthen the main conclusion of the paper. Internal disagreement between direct-answer and full-CoT prompting is reflected not only in which features appear in the

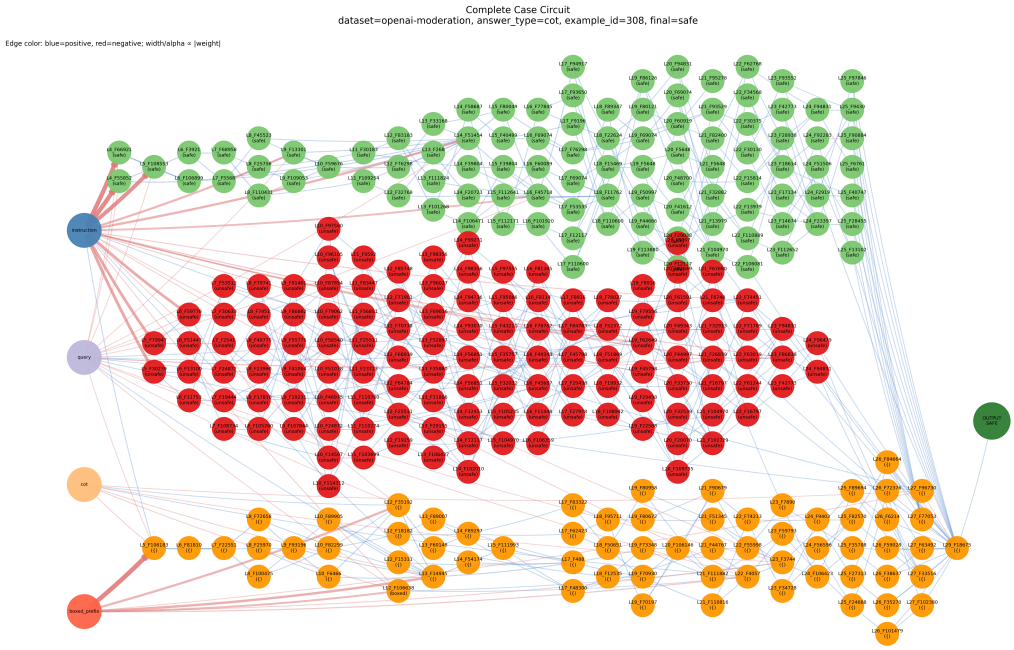


Figure 8: A traced Full-CoT circuit with final prediction `SAFE`, illustrating the proposed roles of early query features in label selection and middle-layer instruction-label features in label copy, with the boxed-answer prefix serving as the output realization site. Node colors indicate interpreted semantic roles; edge color and width indicate attribution sign and magnitude, respectively.

traced circuit, but also in how information is routed across the network. In other words, information-flow structure itself provides a useful signal for identifying cases where the model's internal computation diverges across prompting modes.

D.3 Case Study: A CoT Circuit for Label Selection and Label Copy

To complement the aggregate analyses, we visualize a traced Full-CoT circuit with a final prediction of `SAFE`. We interpret this example using the distinction between label selection and label copy proposed in the main text.

Visualization procedure. For readability, we merge semantically related early-layer features at the token-fragment level where appropriate, while retaining important high-attribution features and annotating their interpreted semantics. The graph highlights pathways connecting the input context, intermediate features, and the final moderation output.

Qualitative observations. Figure 8 shows features associated with both `safe` and `unsafe`, alongside lexical and formatting-related features. We distinguish their proposed roles using their token locations and circuit connections, rather than label-related semantics alone. Early query features are candidates for *label selection*, as they process input-specific evidence relevant to the moderation judgment. Middle-layer features at instruction label tokens are candidates for *label copy*, providing pathways through which the candidate label representations can be routed toward the output. The boxed-answer prefix serves as the site for realizing the selected label, with the circuit ultimately leading to `OUTPUT SAFE`.

Interpretation. This case illustrates how input-dependent evidence and prompt-provided label representations may contribute to the same prediction through different pathways. The CoT context provides an additional source of information within this computation. The visualization offers qualitative context for the proposed functional distinction; causal evidence for the targeted early-query

Table 6: Representative top-activating snippets for the most salient disagreement-associated features.

Feature	Representative top-activating snippets
22_90685	references to violence, weapons, illegal activities, or explicit sexual content. It does not fall into the categories of U1, U2, U3, U5, U6, U7, U8, U9, U10, U11, U14, ... The message does not suggest self-harm or suicide, nor does it mention any illegal or harmful activities that would fall under U1, U2, U3, U5, U6, ... The suggestions are generally peaceful and do not fall into the categories of U1, U2, U3, U4, U5, U6, U7, ...
24_21601	... U10: Privacy & PII – The content involves the theft and misuse of personal information ... ... U9 (Self-Harm & Suicide) ... U10 (Privacy & PII) ... U11 (Intellectual Property) ... U12 (Misinformation & Manipulation) ... ... the message discusses the exfiltration of sensitive patient records, which is a violation of privacy and PII ...
26_5897	You FIRST go through a step-by-step reasoning process and then provide the final answer ... ... The thinking process MUST BE enclosed within ¡/think ... ... best movie ever ..." / "... can I continue driving this car without being discovered ..." / other heterogeneous noisy continuations

and middle-instruction feature sets comes from the Direct-Answer interventions reported in the main text.

D.4 Interpretation of Disagreement-Associated Features

To interpret these disagreement-enriched features, we inspect their top-activating snippets from the transcoder training corpus. As summarized in Table 6, two of the three features admit relatively clear interpretations: one is activated by text about moderation-policy boundaries or exclusion conditions, suggesting a boundary-checking state, while another is activated by examples where privacy or PII is entangled with other harm categories, suggesting a semantically mixed gray-area state. The third feature is less semantically clear, with noisier top activations that do not yield a single stable interpretation. Taken together, these patterns suggest that RSafe is especially uncertain on cases near moderation-policy boundaries and on inputs where privacy or PII overlaps with other harm-related categories, more generally indicating difficulty on semantically mixed or weakly specified cases. In such cases, the model's internal decision state may remain only partially resolved, making the final label more sensitive to prompting mode and helping explain why disagreement is rare overall yet concentrated in a small subset of difficult moderation examples. For completeness, we also analyze features enriched in agreement cases in Appendix D.5.

D.5 Top Agreement-Associated Features

To complement the analysis of disagreement-associated features, we also identify features that are strongly enriched in *agreement* cases, i.e., examples for which direct-answer and full-CoT prompting produce the same moderation label. Table 7 lists the top agreement-associated features ranked by the same score used in the main analysis, which combines enrichment ratio and statistical significance.

A clear pattern is that many of these features are activated frequently in agreement cases but are almost absent in disagreement cases. Several top-ranked features have near-zero activation in the disagreement group while appearing in a substantial fraction of agreement examples. This suggests that these features are highly characteristic of cases in which the model's internal preference is already stable and robust to prompting format.

These agreement-associated features are useful for understanding what kinds of inputs RSafe handles with high confidence. Intuitively, when such features are active, the model appears to form a strong and consistent internal moderation judgment that does not depend heavily on whether explicit chain-of-thought is elicited. In this sense, they provide a mechanistic signature of *easy* or *high-certainty* cases, in contrast to the disagreement-associated features analyzed elsewhere in the paper, which are more indicative of uncertainty, boundary cases, or prompt-sensitive computation.

This analysis also supports the motivation for our two-stage pipeline. If agreement-associated internal signals can be detected reliably at intermediate layers, then many examples may be safely handled without invoking the more expensive full-CoT process, reserving additional reasoning only for cases that lack such stable evidence.

Table 7: Agreement-associated features across 405 agreement and 242 disagreement examples, ranked by the updated score within the listed feature set. Ratio denotes the add-one-smoothed count ratio, and $\mathrm{Score} = -\mathrm{Ratio}\ln(p + 10^{-10})$. The p-values are from two-sided Fisher's exact tests without multiple-testing correction.

Rank	Feature ID	Agree%	Disagree%	Diff (pp)	Ratio	p-value	Score	Agree count	Disagree count
1	18_106753	33.1%	0.0%	+33.1	135.0	2.80×10^{-32}	3108.5	134	0
2	19_23516	20.2%	0.0%	+20.2	83.0	8.14×10^{-19}	1911.1	82	0
3	17_21150	16.5%	0.0%	+16.5	68.0	2.77×10^{-15}	1565.8	67	0
4	17_16423	32.3%	0.4%	+31.9	66.0	1.70×10^{-29}	1519.7	131	1
5	18_5690	15.3%	0.0%	+15.3	63.0	6.92×10^{-14}	1450.6	62	0
6	13_95278	15.3%	0.0%	+15.3	63.0	6.92×10^{-14}	1450.6	62	0
7	21_42194	42.7%	0.8%	+41.9	58.0	3.80×10^{-40}	1335.5	173	2
8	19_20977	13.1%	0.0%	+13.1	54.0	5.67×10^{-12}	1240.4	53	0
9	26_30803	12.8%	0.0%	+12.8	53.0	1.08×10^{-11}	1215.0	52	0
10	19_109295	24.0%	0.4%	+23.5	49.0	7.71×10^{-21}	1128.3	97	1
11	24_32972	44.0%	1.2%	+42.7	44.8	3.55×10^{-40}	1030.4	178	3
12	24_16127	21.7%	0.4%	+21.3	44.5	1.04×10^{-18}	1024.7	88	1
13	23_8380	37.0%	1.2%	+35.8	37.8	5.34×10^{-32}	869.2	150	3
14	19_52375	41.2%	1.7%	+39.6	33.6	2.17×10^{-35}	773.7	167	4
15	14_80049	39.5%	1.7%	+37.9	32.2	1.68×10^{-33}	741.4	160	4
16	20_41115	14.8%	0.4%	+14.4	30.5	3.42×10^{-12}	701.3	60	1
17	4_41395	20.7%	0.8%	+19.9	28.3	1.75×10^{-16}	652.4	84	2
18	24_100156	20.5%	0.8%	+19.7	28.0	3.19×10^{-16}	644.7	83	2
19	10_88664	33.1%	1.7%	+31.4	27.0	2.81×10^{-26}	621.7	134	4
20	17_113522	47.7%	2.9%	+44.8	24.2	1.59×10^{-39}	558.4	193	7

D.6 End-to-end Latency Measurements

To evaluate inference efficiency beyond routing rates, we benchmark end-to-end latency and throughput for all inference modes on a single NVIDIA A100 80GB GPU. We report median (p50) and 95th-percentile (p95) per-sample latency in milliseconds, throughput in requests per second (Req/s), and speedup computed as the ratio of Full-CoT p50 latency to each method's p50 latency.

Table 8: Inference latency and throughput of Full CoT, Direct Answer, and UTM Stage 1 for RSafe and GuardReasoner-8B on a single NVIDIA A100 80GB GPU. We report median (p50) and 95th-percentile (p95) per-sample latency, throughput in requests per second (Req/s), and speedup computed as the ratio of Full-CoT p50 latency to each method's p50 latency. UTM S1 results cover Stage 1 only.

Method	Latency p50 (ms)	Latency p95 (ms)	Req/s	Speedup vs Full CoT
RSafe				
Full CoT	2833.8	5166.6	0.32	1.0×
Direct Answer	76.0	101.5	10.69	37.3×
UTM S1	65.4	85.1	14.60	**43.3**×
GuardReasoner-8B				
Full CoT	6134.9	8722.3	0.16	1.0×
Direct Answer	369.6	717.9	1.83	16.6×
UTM S1	25.1	32.2	37.20	**244.4**×

UTM S1 is faster than Direct Answer, not just Full CoT. UTM S1 replaces all transformer layers beyond layer 22 with two lightweight MLPs and outputs a single probability score with no autoregressive decoding. This makes it 14.7× faster than Direct Answer on GuardReasoner-8B and 244× faster than Full CoT. On RSafe, UTM S1 is also faster than the full forward pass without generation, confirming the efficiency gain from layer truncation alone.

For the full two-stage pipeline (UTM S1+2), efficiency gains come from the lightweight Stage 1 readout and selective use of Full-CoT inference. As shown in Table 4, most examples are resolved at Stage 1, while only a minority are routed to Stage 2. Together with the measured speedup of Stage 1, this supports the pipeline's potential to reduce overall inference cost while maintaining

strong moderation performance. The realized end-to-end speedup depends on the routing rate and the cost of processing routed examples.

D.7 Distinction from Early-exit Methods

Distinction from early-exit literature. Tuned Lens(Belrose et al., 2023) and Logit Lens(Din et al., 2024) study whether final predictions can be decoded from intermediate representations, using per layer affine transformations or learned linear shortcuts to project hidden states into vocabulary space. Our work asks a different question, namely what computational role explicit CoT plays in safety moderation. In our setting, layer wise analysis is a tool for mechanistic understanding rather than the main objective. Our core contributions are mechanistic findings about the model's internal computation, including the shared feature backbone between Direct Answer and CoT, the functional separation into label selection and label copy subcircuits, the concentration of disagreement on boundary features, and the causal role of CoT only features in correcting boundary decisions. These insights cannot be obtained from early exit methods or standard vocabulary space projection methods. Experimental comparison reveals a qualitative gap in intermediate-layer information. To directly validate this distinction, we compare our transcoder-based approach with Logit Lens and Tuned Lens on RSafe under Direct Answer prompting. F1 at representative layers:

Table 9: F1 scores of Logit Lens, Tuned Lens, and our transcoder-based sparse-feature readout at layers 7, 14, and 22 of RSafe under Direct-Answer prompting. Results are reported on Aegis, XSTEST, and OpenAI Moderation. Bold values indicate the highest F1 within each layer and dataset, including ties.

Layer	Method	Aegis	XSTEST	OpenAI Mod.
L7	Logit Lens	0.000	0.000	0.000
	Tuned Lens	0.000	0.000	0.000
	Transcoder (Ours)	**0.649**	**0.518**	**0.657**
L14	Logit Lens	0.000	0.000	0.000
	Tuned Lens	0.000	0.000	0.000
	Transcoder (Ours)	**0.777**	**0.766**	**0.744**
L22	Logit Lens	0.803	**0.856**	0.781
	Tuned Lens	0.814	0.843	0.767
	Transcoder (Ours)	**0.816**	**0.856**	**0.784**

The results show that our transcoder based method recovers the moderation decision substantially earlier than Logit Lens or Tuned Lens. The two lens based baselines become competitive only in the final layers, whereas our method already achieves strong F1 well before that point. This suggests a functional difference between the methods. Logit Lens and Tuned Lens rely on vocabulary space projections and therefore mainly capture the decision once it has been translated into output aligned representations. In contrast, the transcoder captures sparse internal features that encode safety relevant information before it appears in token space. This is consistent with our claim that label selection occurs earlier in the network, while vocabulary projection methods primarily observe the later label copy stage.

D.8 Cross-Model Circuit Analysis on GuardReasoner-8B

Experimental setup. To assess whether our circuit-level findings extend beyond RSafe, we apply the same transcoder and EAP circuit-tracing pipeline to GuardReasoner-8B on XSTEST. Unlike RSafe, which is trained with reinforcement learning, GuardReasoner-8B uses supervised fine-tuning followed by HDPO. We repeat the feature-overlap analysis between Direct-Answer and Full-CoT modes and the segment-level attribution analysis within each mode. This comparison tests the findings in a second model trained under a different paradigm.

Feature Overlap Across Prompting Modes Table 10 reports the shared-feature proportion averaged across layers within each layer range, separately for agreement and disagreement cases. In the lower layers, the two groups have nearly identical shared proportions (36.9% versus 36.6%). The

gap increases to 6.5 percentage points in the middle layers and 14.1 percentage points in the higher layers. Thus, disagreement is associated with greater divergence in attributed feature usage from the middle layers onward, reproducing the depth-dependent agreement–disagreement contrast observed in RSafe.

The absolute overlap is more limited: the reported shared proportions range from 20.9% to 36.9%. These results therefore support the presence of shared attributed features and stronger divergence in disagreement cases, rather than establishing that shared features constitute a majority in GuardReasoner-8B. The replicated contrast suggests that this pattern is not confined to RSafe's RL-trained reasoning behavior.

Table 10: Shared attributed-feature proportions between Direct-Answer and Full-CoT modes on GuardReasoner-8B evaluated on XSTEST. Values are averaged across layers within each range. Gap denotes agreement minus disagreement, in percentage points (pp). E denotes the embedding layer.

Layer range	Agreement (%)	Disagreement (%)	Gap (pp)
Low (E–10)	36.9	36.6	0.3
Mid (11–20)	28.1	21.6	6.5
High (21–31)	35.0	20.9	14.1

Segment-Level Attribution Table 11 reports the distribution of attributed features across sequence segments at each layer. In Direct-Answer mode, the instruction share fluctuates but generally decreases with depth, while the answer-prefix share becomes larger in later layers, reaching 74.1% at layer 21 and 69.6% at layer 22. This differs from the instruction-dominated pattern observed in RSafe, while remaining consistent with increasing concentration of attributed computation near the output position. The average feature count is small in later layers; in particular, layers 23–31 have fewer than one attributed feature per example on average and are omitted for this mode.

Full-CoT mode exhibits a different distribution. The query and instruction shares diminish with depth and reach zero from layer 20 onward. In contrast to RSafe, attributed features on the CoT segment remain prominent throughout the middle and later layers, and their share exceeds that of the answer prefix at many layers from layer 12 onward. Thus, GuardReasoner-8B does not reproduce the disappearance of CoT-segment features observed in RSafe. The counts also become small at greater depths: layer 30 has only 0.9 attributed features per example on average, and layer 31 has none. Accordingly, proportions at these depths should be interpreted cautiously.

Label Information in the CoT Segment To investigate the persistent attribution to the CoT segment, we inspect the semantic content of its most highly attributed features. These features are primarily associated with label information, including `harmful` and `unharmful`, rather than generic reasoning content. This observation suggests that the CoT segment may serve as a source of label information for the final prediction in GuardReasoner-8B.

We further quantify explicit label-word mentions in generated CoT traces across five moderation datasets, including the response-moderation benchmark SafeRLHF. We count the label words used by each model: `safe`/`unsafe` for RSafe and `harmful`/`unharmful` for GuardReasoner-8B. Table 12 reports the mean number of mentions per trace. GuardReasoner-8B averages 4.90–6.50 mentions across datasets, compared with 0.40–0.86 for RSafe. Across the evaluated samples, GuardReasoner-8B explicitly mentions a label in 99.6% of traces, whereas RSafe does so in 37%.

Interpretation. Together, these observations are consistent with a distinction between label selection and label copy or realization, while suggesting that the source of label information differs across models. In RSafe, later attributed computation is concentrated on the instruction and answer-prefix segments. In GuardReasoner-8B, the generated CoT repeatedly contains explicit label information and remains a prominent attributed segment alongside the answer prefix. Label information may therefore be routed from the CoT itself to the final output. This interpretation explains why persistent CoT attribution need not exclusively reflect continued deliberation, although attribution patterns and label mentions alone do not establish a causal label-copy mechanism or precisely locate the transition between selection and realization.

Table 11: Layer-wise segment distributions of attributed features for GuardReasoner-8B on XSTEST. All segment shares are percentages; Avg denotes the mean number of attributed features per example. Prefix denotes the answer-prefix segment, and E denotes the embedding layer. Direct-Answer layers 23–31 are omitted because their average attributed-feature counts are below 1. Full-CoT layer 31 has no attributed features, so its segment proportions are undefined. Minor deviations from 100% reflect rounding.

	Direct-Answer				Full-CoT				
Layer	Instr.	Query	Prefix	Avg	Instr.	Query	CoT	Prefix	Avg
E	89.7	8.0	2.3	130.5	32.0	3.0	63.3	1.7	347.3
0	57.0	32.2	10.8	1347.5	7.3	15.6	64.2	13.0	2244.2
1	35.9	41.3	22.8	284.6	2.6	21.3	36.1	39.9	318.4
2	38.3	40.0	21.7	239.0	3.0	22.8	31.9	42.3	256.2
3	37.1	30.3	32.6	114.5	3.0	12.3	25.0	59.7	135.4
4	46.2	27.8	26.0	77.4	2.6	10.8	24.0	62.6	95.3
5	41.6	30.9	27.5	64.6	1.4	9.5	25.6	63.5	97.4
6	39.3	27.2	33.5	56.9	2.0	7.6	26.1	64.3	70.8
7	37.2	28.3	34.5	39.0	3.2	7.1	36.3	53.4	53.1
8	36.2	21.0	42.8	24.0	3.1	4.8	27.9	64.2	38.0
9	31.2	18.7	50.1	22.9	1.3	4.7	28.2	65.9	30.4
10	43.8	20.0	36.1	22.8	1.8	4.1	30.3	63.8	29.0
11	41.7	23.4	34.9	14.9	3.6	4.2	36.8	55.4	21.3
12	54.3	15.8	29.9	16.0	3.8	4.6	57.4	34.1	23.6
13	52.5	19.1	28.4	6.7	3.4	4.7	29.8	62.1	15.2
14	33.7	30.4	35.9	4.4	0.5	5.4	48.4	45.7	8.9
15	25.7	31.4	42.9	3.3	4.6	3.4	54.6	37.4	8.3
16	27.6	56.9	15.5	2.8	0.7	4.5	37.3	57.5	6.4
17	19.0	33.3	47.6	2.0	2.0	0.7	27.5	69.7	6.8
18	11.1	44.4	44.4	1.3	2.6	0.0	40.0	57.4	5.5
19	21.2	39.4	39.4	1.6	0.0	2.9	51.0	46.2	5.0
20	10.0	25.0	65.0	1.0	0.0	0.0	52.3	47.7	5.3
21	11.1	14.8	74.1	1.3	0.0	0.0	35.8	64.2	5.2
22	17.4	13.0	69.6	1.1	0.0	0.0	51.1	48.9	2.1
23	–	–	–	–	0.0	0.0	55.6	44.4	4.7
24	–	–	–	–	0.0	0.0	54.5	45.5	3.1
25	–	–	–	–	0.0	0.0	73.5	26.5	1.6
26	–	–	–	–	0.0	0.0	84.2	15.8	1.8
27	–	–	–	–	0.0	0.0	55.6	44.4	2.1
28	–	–	–	–	0.0	0.0	57.4	42.6	3.2
29	–	–	–	–	0.0	0.0	70.2	29.8	2.7
30	–	–	–	–	0.0	0.0	0.0	100.0	0.9
31	–	–	–	–	–	–	–	–	0.0

Table 12: Mean number of explicit label-word mentions per generated CoT trace. Label words are `safe`/`unsafe` for RSafe and `harmful`/`unharmful` for GuardReasoner-8B.

Model	Aegis	OAI Mod.	Toxic-Chat	XSTEST	SafeRLHF
RSafe	0.47	0.40	0.86	0.40	0.53
GuardReasoner-8B	5.56	6.50	5.90	5.60	4.90

The different patterns may reflect differences in training data or supervision format that encourage GuardReasoner-8B to verbalize label information during reasoning. Overall, this additional model reproduces the stronger feature divergence in disagreement cases while revealing a different distribution of label-related computation across sequence segments. These findings extend the analysis beyond RSafe and suggest that the proposed functional distinction may be implemented through different pathways across models.

D.9 Other Evaluation Metrics

We report the full breakdown of false positive rate (FPR), and false negative rate (FNR) for both RSafe and GuardReasoner-8B across all evaluation datasets.

Table 13: F1, false positive rate (FPR), and false negative rate (FNR) of RSafe and GuardReasoner-8B under Direct-Answer and Full-CoT prompting across four moderation benchmarks. FPR measures the fraction of safe examples classified as unsafe, while FNR measures the fraction of unsafe examples classified as safe. Higher F1 and lower FPR/FNR indicate better performance.

Dataset	Mode	**RSafe**			**GuardReasoner-8B**		
		F1	FPR	FNR	F1	FPR	FNR
Aegis	Direct	0.819	0.198	0.173	0.811	0.343	0.109
	CoT	0.831	0.159	0.192	0.835	0.267	0.113
OpenAI Mod.	Direct	0.765	0.241	0.046	0.723	0.309	0.046
	CoT	0.771	0.235	0.045	0.725	0.304	0.048
XSTEST	Direct	0.843	0.209	0.077	0.934	0.112	0.035
	CoT	0.870	0.104	0.130	0.944	0.060	0.040
SafeRLHF	Direct	0.908	0.161	0.047	0.891	0.245	0.007
	CoT	0.921	0.123	0.051	0.917	0.164	0.021

Overall, CoT mainly helps by reducing false positives rather than false negatives. Across both models, FPR decreases consistently, while FNR changes only modestly. This suggests that CoT refines boundary decisions instead of simply pushing predictions toward unsafe, which is also consistent with our mechanistic analysis that CoT specific features play a corrective role on uncertain inputs.

E Computational Resources and Implementation Details

E.1 Computational Resources and Software

We report the main hardware and software resources used in this work to support reproducibility.

Hardware. The mechanistic analysis in this paper, including transcoder-based circuit tracing and attribution analysis, is the most computationally expensive component of the study. These experiments were conducted using 8 NVIDIA H20 GPUs. In contrast, the experiments for the proposed uncertainty-guided two-stage moderation pipeline are substantially lighter. Inference for the two-stage method was run on a single NVIDIA A100 80GB GPU.

Software and libraries. Our implementation builds on several open-source libraries commonly used in mechanistic interpretability research. For transcoder training, we use the `sparsify` codebase from EleutherAI.[1] For circuit tracing, we build on the `circuit-tracer` framework.[2] For model loading, activation access, and intervention-based analysis, we use TransformerLens.[3]

[1] `https://github.com/EleutherAI/sparsify`
[2] `https://github.com/decoderesearch/circuit-tracer/tree/main`
[3] `https://github.com/TransformerLensOrg/TransformerLens`

Implementation modifications. On top of these open-source codebases, we implemented several engineering modifications needed for our experiments, including support for multi-GPU execution in parts of the TransformerLens and circuit-tracer-based pipeline. These changes were necessary to make the circuit-tracing analysis feasible at the scale used in this work.

Practical note. The substantial difference in compute requirements between circuit tracing and the two-stage moderation experiments is also reflected in the scope of our study: the mechanistic analysis is necessarily more limited in scale, whereas the proposed two-stage pipeline is much cheaper to run at inference time.

E.2 Prompts and Decoding Settings

Prompts. For each model, the Full-CoT prompt follows the template released in its official codebase. The Direct-Answer prompt uses the same template, with `\n<think>\n</think>` appended immediately after the query to close the reasoning block and elicit the moderation label directly.

Decoding configuration. All classification-evaluation runs use deterministic greedy decoding (no sampling): `temperature=0.0`, `top_p=1.0`, no top-k restriction, and `max_tokens=2048`. This removes decoding randomness as a confound when comparing Direct-Answer and Full-CoT outputs. Chat formatting follows each model's original release setting: RSafe uses the Qwen-style chat template; GuardReasoner-8B is run without a chat template, consistent with its original paper.

E.3 Transcoder Architecture and Training

We train one transcoder per layer using activations collected from the frozen base model. Each transcoder uses a sparse dictionary of size $N = 32h$, where h is the residual-stream dimension, with learned bias terms in both the encoder and decoder. We apply a TopK activation with $k = 32$, retaining only the 32 largest feature activations for each token and setting the remaining activations to zero.

We train the transcoders on 40K samples drawn from in-domain content-moderation data. The training objective combines reconstruction MSE with an ℓ_1 sparsity penalty weighted by $\lambda_{\text{sparse}} = 10^{-3}$. We use Adam with a learning rate of 2×10^{-4}, a batch size of 16, and one training epoch.

E.4 Feature and Edge Selection Criteria

We use Circuit Tracer's linear attribution graph rather than a fixed absolute attribution-score threshold τ.

Attribution. For each sample, we use the logit difference between the target moderation label and the competing label as the attribution objective. This contrastive objective focuses attribution on computation that favors the target label over its alternative. Feature nodes are then expanded greedily according to partial influence, with `max_feature_nodes=8192`.

Pruning. Rather than thresholding individual scores, we keep the smallest node set whose cumulative node influence reaches 80% (`node_threshold=0.8`), then the smallest edge set whose cumulative edge influence reaches 98% (`edge_threshold=0.98`) within that node set, iteratively removing nodes left without any remaining incoming or outgoing edges.

Threshold selection. These thresholds are fixed constants chosen to match the Circuit Tracer defaults. We did not perform a grid search over them or tune them against downstream classification metrics. Rather, they define what counts as attributed solely through the graph's own influence mass criterion and are therefore independent of task performance.

F LIMITATIONS

Our main limitation is the substantial computational cost of mechanistic analysis, including transcoder training, circuit tracing, and causal interventions. This cost constrains both the breadth and depth of our experiments. While we conduct a comprehensive analysis of RSafe, our additional experiments on GuardReasoner-8B are limited to feature-overlap and segment-level attribution analyses on XSTEST and are therefore less extensive. Moreover, our circuit-level analysis focuses on prompt moderation and does not extend to response moderation. Although we evaluate the UTM pipeline on a response-moderation benchmark, a corresponding mechanistic analysis remains future work. Extending the analysis across more models and moderation settings will require additional computational resources.