Beyond Analysis: Training Language Models with Internal Mechanistic Feedback

Anonymous ACL submission

Abstract

Recent advances in mechanistic interpretability have revealed how language models process information, yet these insights rarely improve model performance. We propose Interpretable Feature-Space Regularization (IFSR), a training-time framework that transforms mechanistic insights into optimization signals. IFSR identifies error-prone feature interactions through circuit attribution and penalizes them during training, encouraging the model to discover alternative computational pathways. Unlike inference-time interventions that operate on individual features, IFSR targets feature-to-feature edges representing internal computational patterns, and permanently encodes improvements into model parameters. Experiments across ten classification tasks show consistent improvements. Cross-task evaluation demonstrates that IFSR training can transfer positively to unrelated tasks, suggesting benefits to general model capabilities. Our analysis reveals that most identified error patterns resist human interpretation, yet penalizing them still improves performance, suggesting that automatic error identification at the level of feature interactions is feasible and effective. This work demonstrates that mechanistic interpretability can directly enhance task performance through training-time optimization.

1 Introduction

Mechanistic interpretability has made remarkable progress in understanding how language models work internally. Techniques like sparse autoencoders (Cunningham et al., 2023; Gao et al., 2024; Rajamanoharan et al., 2024) and cross-layer transcoders (Ameisen et al., 2025) decompose dense activations into sparse, interpretable features, while circuit attribution methods (Lindsey et al., 2025) are designed to trace how these features causally influence predictions. These tools have revealed computational pathways underlying in-context learning (Olsson et al., 2022), indirect object identification (Wang et al., 2022), and mathematical reasoning (Nanda et al., 2023).

Yet despite these advances, mechanistic interpretability has rarely improved model performance. The insights often remain diagnostic: we analyze model internals but lack principled methods to translate understanding into performance gains. Activation steering attempts to bridge this gap by directly intervening on feature activations during inference. However, recent large-scale evaluation reveals that steering effects are brittle and generalize poorly across model families and tasks (Silva et al., 2025). Indeed, steering has proven most valuable not for improving task performance, but as a tool for causal validation of interpretability hypotheses (Templeton et al., 2024). What is missing is a method that translates mechanistic insights into *training-time* optimization, permanently encoding improvements into model parameters.

We propose a different approach: instead of steering individual features at inference time, we identify error-prone **feature interactions** through circuit attribution and penalize them during **training**. This encourages the model to discover alternative computational pathways that achieve the same objectives while avoiding error-inducing patterns. The improvements are permanently encoded into model parameters, requiring no intervention.

Achieving this requires rethinking what interpretability means for optimization. Due to the superposition phenomenon (Elhage et al., 2022), models encode vastly more features than their neuron count, making exhaustive human analysis infeasible. We observe that many error-inducing feature interactions resist human interpretation, yet penalizing them still improves performance. This suggests a data-driven paradigm: use circuit attribution to automatically distinguish error-prone activation patterns from beneficial ones, and let the model adjust its computations accordingly.

We propose Interpretable Feature-Space Regularization (IFSR), a framework realizing this vision. First, we employ a cross-layer transcoder mapping activations into a sparse feature space. Second, on a held-out attribution set, we identify error edges: feature pairs frequently co-activated in wrong predictions but rarely in correct ones. Third, we train with a penalty on error edge activation, guiding the model toward alternative pathways.

We evaluate IFSR across ten tasks spanning natural language inference, sentiment analysis, financial prediction, code defect detection, and commonsense reasoning. IFSR consistently improves performance, and cross-task evaluation reveals that training preserves or enhances capabilities on unrelated tasks. Ablation studies confirm that random penalties are ineffective, and operating in feature space rather than on neurons is essential.

This work makes three contributions. First, we introduce IFSR, a training-time framework that targets feature interactions rather than individual features, permanently encoding improvements into model parameters. Second, we demonstrate that error-prone computational patterns can be automatically identified and corrected through a data-driven approach. Third, through experiments spanning ten diverse tasks, we provide evidence that internal supervision at the level of feature interactions unlocks gains beyond output-level optimization.

2 Related Work

Mechanistic Interpretability Language models encode information in distributed representations where individual neurons respond to multiple unrelated concepts, known as polysemanticity (Elhage et al., 2022). Sparse autoencoders decompose activations into sparse, monosemantic features (Cunningham et al., 2023; Gao et al., 2024; Rajamanoharan et al., 2024; Templeton et al., 2024). Early circuit analysis identified specific computational patterns in smaller models, such as induction heads for in-context learning (Olsson et al., 2022) and indirect object identification circuits (Wang et al., 2022). Recent advances in transcoders (Dunefsky et al., 2024) and cross-layer transcoders (Ameisen et al., 2025) have extended these capabilities to frontier models, enabling discovery of diverse and complex circuits at scale (Lindsey et al., 2025). Yet translating these insights into training improvements remains unexplored.

Activation Steering Steering methods intervene on model representations to modify behavior (Templeton et al., 2024; Cho et al., 2025; Kang et al., 2025; Xu et al., 2025). However, comprehensive evaluation reveals significant limitations: steering effects are brittle and generalize poorly across model families (Silva et al., 2025). In practice, steering has proven most valuable for causal validation of interpretability findings rather than systematic performance improvement (Templeton et al., 2024). Furthermore, steering operates on individual features, which may be insufficient when errors arise from complex feature interactions.

3 Interpretable Feature-Space Regularization

We present IFSR, a framework that transforms mechanistic interpretability from post-hoc analysis into training-time optimization. As illustrated in Figure 1, IFSR operates in three stages: constructing an interpretable feature space (Section 3.1), identifying error patterns through circuit attribution (Section 3.2), and training with feature-space regularization (Section 3.3). The process can be iterated for progressive refinement (Section 3.4).

Notation We denote the language model's residual stream activation at layer ℓ as $h_\ell \in \mathbb{R}^d$, where d is the model's hidden dimension. The model has L layers in total. For classification tasks, we focus on activations at a designated target position (typically the final token), so we omit position indices.

Data Partitioning We divide data into three disjoint sets: a training set for model optimization, an attribution set for error pattern identification, and a validation set for evaluation. Separating training and attribution sets prevents overfitting to observed error patterns; keeping the validation set disjoint from both prevents data leakage during evaluation.

3.1 Feature Space Construction

Individual neurons in language models exhibit polysemanticity, responding to multiple unrelated concepts (Elhage et al., 2022). This makes neuron-level intervention unreliable: penalizing a single neuron may inadvertently suppress multiple unrelated behaviors. We employ cross-layer transcoders (CLTs) (Ameisen et al., 2025) to decompose activations into sparse, monosemantic features.

A CLT maps the residual stream at layer ℓ to a sparse feature vector and reconstructs MLP outputs

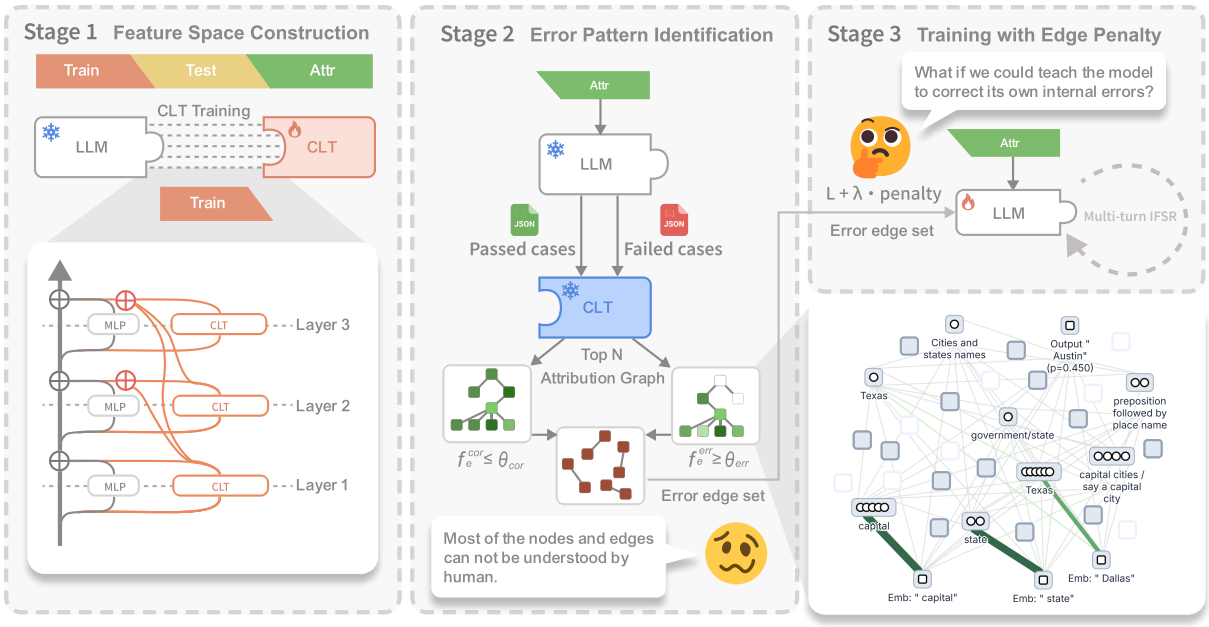

Figure 1: **Overview of the Interpretable Feature-Space Regularization (IFSR).** The framework consists of three key stages: (1) Feature Space Construction: Decomposing polysemantic residual stream activations into sparse, monosemantic features with CLTs. (2) Error Pattern Identification: Using circuit attribution to identify feature-to-feature edges that correlate with incorrect predictions. (3) Training with Edge Penalty: Optimizing the model with a differentiable co-activation penalty on identified error edges.

at all subsequent layers. Given residual stream activation h_ℓ at layer ℓ (at a given token position), the CLT computes:

$$f_\ell = \mathrm{ReLU}(W_{\text{enc}} h_\ell + b_{\text{enc}}) \in \mathbb{R}^D \tag{1}$$

where $D \gg d$ is the number of transcoder features (typically $D = 16 \times d$ or larger). The CLT reconstructs the MLP output at each subsequent layer $\ell' > \ell$ as:

$$\hat{m}_{\ell'} = \sum_{i: f_\ell[i] > 0} f_\ell[i] \cdot W_{\text{dec}}^{(\ell')}[i] + b_{\text{dec}}^{(\ell')} \tag{2}$$

where $W_{\text{dec}}^{(\ell')}[i] \in \mathbb{R}^d$ is the decoder weight for feature i at layer ℓ'.

Why Cross-Layer Transcoders Unlike SAEs operating within single layers, CLTs span multiple layers, enabling direct analysis of feature-to-feature interactions with well-defined linear attribution (Ameisen et al., 2025).

The CLT is pretrained on diverse text to minimize reconstruction error plus a sparsity penalty, independent of any downstream task. Once trained, CLT parameters remain frozen throughout IFSR, providing a stable coordinate system for identifying computational patterns.

3.2 Error Pattern Identification

We identify error-prone feature interactions through circuit attribution on the attribution set. For classification tasks, we focus on feature activations at the **target position**, typically the final token or a designated classification token. This is where the model produces its prediction.

Attribution Graph Construction For each sample, we build an attribution graph $G = (V, E)$ at the target position, capturing how features causally influence the prediction. Nodes in V include feature nodes (ℓ, i) representing the activation of feature i at layer ℓ, as well as logit nodes representing the output logits. We denote the activation value of node v as a_v.

To determine which edges exist in the graph, we compute edge weights using a local linear approximation following Ameisen et al. (2025). The weight of an edge from node s to node t is:

$$A_{s \to t} = \frac{\partial a_t}{\partial a_s} \cdot a_s \tag{3}$$

This captures both the sensitivity of t to s (the gradient) and the magnitude of s's activation. We provide a detailed derivation in Appendix A.

Graph Pruning Since each sample activates thousands of features, directly analyzing the full

graph is intractable. We prune by retaining only the top N features ranked by their total effect on output logits, computed as the sum of $|A_{v\to\text{logit}}|$ for each feature node v. Edges between retained nodes with $|A_{s\to t}|$ above a threshold are included in the pruned graph. After pruning, we treat edges as **binary**: an edge either exists or does not. The subsequent error edge detection operates on edge presence rather than edge weights.

Contrastive Error Edge Detection We partition attribution graphs into those from correct predictions $\mathcal{G}_{\text{cor}}$ and incorrect predictions $\mathcal{G}_{\text{err}}$. Crucially, when comparing across samples, we identify edges by their **layer and feature indices only**, abstracting away position information. This allows us to detect feature interactions that systematically correlate with errors across samples of varying lengths.

For each edge $e = ((\ell_s, i_s), (\ell_t, i_t))$, let n_e^{err} and n_e^{cor} denote the number of graphs in $\mathcal{G}_{\text{err}}$ and $\mathcal{G}_{\text{cor}}$ containing e, respectively. We compute frequency ratios:

$$f_e^{\text{err}} = \frac{n_e^{\text{err}}}{|\mathcal{G}_{\text{err}}|}, \quad f_e^{\text{cor}} = \frac{n_e^{\text{cor}}}{|\mathcal{G}_{\text{cor}}|} \tag{4}$$

We first filter edges satisfying: $f_e^{\text{err}} \geq \theta_{\text{err}}$ and $f_e^{\text{cor}} \leq \theta_{\text{cor}}$, where θ_{err} and θ_{cor} are frequency thresholds. From these candidates, we select the top K edges ranked by log-contrast score:

$$\text{score}(e) = \log\left(1 + \frac{f_e^{\text{err}}}{f_e^{\text{cor}} + \epsilon}\right) \tag{5}$$

where ϵ is a small constant preventing division by zero. The logarithmic transformation prevents extreme weights when f_e^{cor} is near zero. We normalize scores to have mean 1 and use them as edge weights w_e in the penalty term. We focus on feature-to-feature edges, as these represent internal pathways the model can adjust during training.

3.3 Training with Edge Penalty

With error edges identified, we train the model with a combined objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}}(y, \hat{y}) + \lambda \cdot \mathcal{L}_{\text{penalty}} \tag{6}$$

where $\mathcal{L}_{\text{CE}}$ is cross-entropy loss and λ controls penalty strength.

Edge Penalty Computation For each training sample, we extract feature activations at the target position by passing the model's residual stream through frozen CLT encoders at each layer. For each error edge $e = ((\ell_s, i_s), (\ell_t, i_t))$, we penalize the co-activation:

$$\mathcal{L}_{\text{penalty}} = \frac{1}{|E|} \sum_{e \in E} w_e \cdot |f_{\ell_s}[i_s] \cdot f_{\ell_t}[i_t]| \tag{7}$$

where $f_\ell[i]$ denotes the activation of feature i at layer ℓ (at the target position), E is the error edge set, and w_e is the normalized edge weight.

Why Co-Activation Penalty Note that the penalty targets feature *co-activations* rather than directly using attribution weights. This design choice reflects our goal: we want to discourage the model from *simultaneously activating* feature pairs that frequently co-occur in errors. The attribution graph identifies *which* edges are problematic; the co-activation penalty provides a differentiable signal to *avoid* those patterns during training.

Penalizing error edges does not simply suppress model outputs; rather, it encourages the model to discover alternative computational pathways that achieve the same task objectives. Since language models are highly over-parameterized, alternative pathways typically exist but remain underutilized under standard cross-entropy training. The penalty creates a gradient signal that redistributes computations through less error-prone circuits.

3.4 Iterative Refinement

During IFSR training, the CLT remains frozen while model parameters update. This separation is intentional: the CLT provides a stable observation framework for defining problematic activation patterns, while the model learns to adjust its computations within this framework. Within a single training round, parameter updates are sufficiently local that the CLT's feature definitions remain approximately valid.

After substantial training, however, the accumulated parameter changes may shift the model's activation distribution beyond the CLT's effective range. We address this through iterative refinement: after training produces model M_1, we fine-tune the CLT on M_1's activations to yield C_1, then re-run attribution with (M_1, C_1) to identify new error patterns. This updated error set drives another round of IFSR training.

CLT Adaptation Following recent work on efficient SAE training (Karvonen, 2025), we fine-tune the CLT using a mixture of general pretraining data and task-specific data, ensuring the CLT accurately captures the updated model's representations.

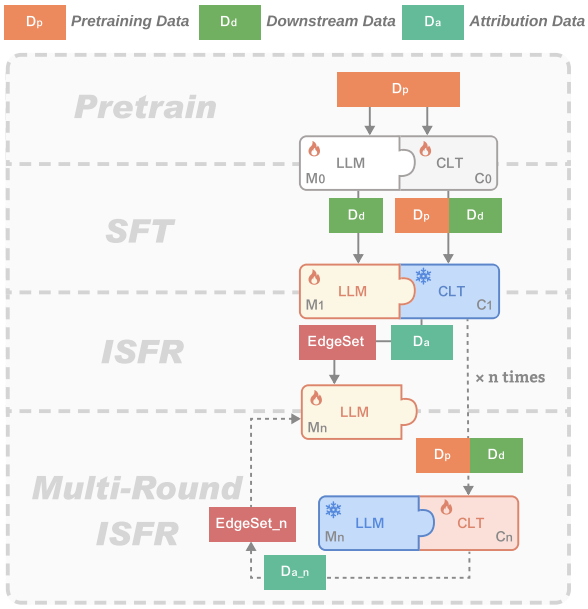

Figure 2: **Multi-round iterative refinement.** IFSR can start from either a base model (M_0, C_0) or an SFT checkpoint (M_1, C_1). Each round re-identifies error edges on fresh attribution data (D_{a_n}) using the updated CLT, enabling progressive refinement. Fire: trainable; snowflake: frozen.

When to Iterate We recommend starting a new round in two scenarios: (1) when validation performance plateaus for several consecutive epochs, indicating the current error patterns have been sufficiently addressed, or (2) when CLT reconstruction quality degrades significantly, indicating the feature space no longer accurately represents the model's computations. Both signals suggest that the current CLT-model pairing has reached its optimization limit and re-alignment is needed.

4 Experiments

4.1 Experimental Setup

Tasks and Datasets We evaluate IFSR on ten classification tasks spanning diverse domains: natural language inference (MNLI), sentiment analysis (SST-2), and linguistic acceptability (CoLA) from the GLUE benchmark (Wang et al., 2019); reading comprehension (BoolQ) (Clark et al., 2019); news classification (AG News) (Zhang et al., 2016); emotion classification (Emotion) (Saravia et al., 2018); financial sentiment (Finance) (Araci, 2019); safety classification (BeaverTails) (Ji et al., 2023); code defect detection (CodeXGLUE) (Zhou et al., 2019); and logical fallacy detection (Logic) (Jin et al., 2022). For each task, we partition the data into training, attribution, and test sets. Detailed dataset statistics and partitioning are provided in Appendix B.

Model We use Gemma-2-2B (Team et al., 2024) as the base model with its corresponding pretrained cross-layer transcoder from Gemma Scope (Lieberum et al., 2024). We select Gemma-2-2B because it currently has the most mature mechanistic interpretability ecosystem, with publicly available high-quality sparse autoencoders and transcoders. The CLT remains frozen throughout single-round IFSR experiments; only in multi-round iterative experiments do we fine-tune the CLT.

Attribution Set We sample 500 examples as the attribution set for error edge identification across all tasks. Although this sample size appears modest, each pruned attribution graph contains hundreds of feature nodes and thousands of edges, providing sufficient coverage for reliable pattern detection while keeping computational cost tractable.

Baselines We compare IFSR against: (1) the base model without fine-tuning, (2) standard supervised fine-tuning (SFT), (3) SFT with L2 regularization (weight decay = 0.01), (4) SFT with random edge penalty, and (5) SFT with neuron-level penalty. For fair comparison, random edge penalty selects the same number of edges as IFSR but randomly, using identical penalty weight λ. Neuron-level penalty operates directly on MLP activations: we identify error neurons on the attribution set by computing the ratio of mean activation magnitude on incorrect predictions to that on correct predictions, then penalize high-contrast neurons during training with matched λ and count.

Implementation All models are trained on 8× NVIDIA H800 GPUs for 2 epochs with learning rate 2×10^{-5} and batch size 8. The penalty weight λ controls the trade-off between standard language modeling and error suppression. Empirically, we find that $\lambda \in [0.1, 1.0]$ yields optimal performance on most tasks (see Appendix C), with the method showing low sensitivity to the exact value within this range. IFSR introduces modest computational overhead (approximately 5–11% training time); see Appendix D for details.

4.2 Main Results

Table 1 presents test accuracy across all tasks. All reported results are the average of three independent runs with different random seeds to ensure statistical stability. IFSR consistently outperforms

Task	Base	SFT	IFSR
MNLI	30.85	85.90	**90.63**$\uparrow_{4.73\%}$
SST-2	76.60	95.00	**96.50**$\uparrow_{1.50\%}$
Emotion	28.60	93.65	**93.95**$\uparrow_{0.30\%}$
AG News	28.40	91.20	**92.85**$\uparrow_{1.65\%}$
Finance	28.99	78.22	**83.51**$\uparrow_{5.29\%}$
CoLA	65.10	82.55	**84.28**$\uparrow_{1.73\%}$
BoolQ	58.10	80.75	**86.35**$\uparrow_{5.60\%}$
BeaverTails	43.80	84.95	**86.05**$\uparrow_{1.10\%}$
CodeXGLUE	40.86	63.81	**66.04**$\uparrow_{2.23\%}$
Logic	8.59	14.74	**15.09**$\uparrow_{0.35\%}$
Average	41.34	77.08	**79.53**$\uparrow_{2.45\%}$

Table 1: Test accuracy (%) on ten classification tasks. IFSR consistently improves over SFT across all tasks.

standard SFT, with average improvements ranging from 0.3 to 5.6 percentage points.

The gains are particularly notable on medium-difficulty tasks: BoolQ (+5.6%), Finance (+5.3%), and MNLI (+4.7%). For tasks where SFT already achieves near-ceiling performance (SST-2: 95%), IFSR still provides modest gains (+1.5%), but the room for improvement is inherently limited. Similarly, for particularly challenging tasks (Logic: 14.74% on 13-way classification), where the base model struggles fundamentally, IFSR offers limited additional benefit (+0.35%), suggesting that error edge correction cannot compensate for insufficient base capability. These results demonstrate that IFSR provides consistent improvements across diverse task types and difficulty levels, with the largest gains on tasks where SFT leaves meaningful room for improvement.

4.3 Ablation Study

Table 2 compares IFSR against alternative regularization strategies across all ten tasks.

L2 Regularization L2 regularization penalizes all parameters uniformly and cannot distinguish useful patterns from problematic ones. It slightly underperforms SFT on average (76.00% vs 77.08%), confirming that uniform parameter shrinkage does not provide targeted error correction.

Random Edge Penalty Penalizing randomly selected feature edges with identical hyperparameters degrades performance below SFT (75.51% vs 77.08%). This confirms that our error edge identification is essential: arbitrary feature suppression is actively harmful, while principled selection improves performance.

Neuron-Level Penalty Operating directly on neurons rather than sparse features yields substantial degradation (64.47% average), often below the base model. We identify error neurons by computing the ratio of mean activation on incorrect samples to correct samples:

$$\mathcal{L}_{\text{neuron}} = \frac{1}{|\mathcal{N}_{\text{err}}|} \sum_{(l,i)\in\mathcal{N}_{\text{err}}} w_{l,i} \cdot |h_l^{(i)}| \qquad (8)$$

where $\mathcal{N}_{\text{err}}$ denotes error neurons, $h_l^{(i)}$ is neuron activation, and $w_{l,i}$ is the contrast ratio. Crucially, even the highest-ranked error neurons exhibit modest contrast ratios, typically between 1.4× and 2.8× (see Appendix E), indicating that error-related activation patterns are diffusely distributed across the neuron space rather than concentrated in specific units. The severe performance degradation demonstrates the polysemanticity problem: penalizing a neuron suppresses multiple unrelated behaviors, causing catastrophic interference. This validates the necessity of sparse feature space for targeted intervention.

4.4 Cross-Task Generalization

A natural concern is whether IFSR, which penalizes task-specific error patterns, might harm performance on other tasks. Table 3 shows cross-task evaluation: rows indicate the IFSR training task, columns indicate the evaluation task (base model accuracy shown in the first row for reference).

Results show that IFSR training generally does not harm performance on unrelated tasks. In many cases, we observe positive transfer: for example, IFSR trained on Logic improves CoLA from 65.10% to 69.22%, and IFSR trained on Finance improves CoLA from 65.10% to 68.55%. This suggests that error patterns are not entirely task-specific; correcting certain computational pathways may improve general linguistic representations that benefit multiple tasks. A few exceptions exist: AG News training notably degrades SST-2 and CoLA, likely because news classification encourages topic-focused representations that conflict with sentiment and grammaticality judgments. Overall, transfer patterns are not always predictable, but IFSR rarely causes substantial degradation on unrelated tasks.

4.5 Error Edge Analysis

We analyze the structure and semantics of identified error edges to understand what features contribute to model failures.

Method	MNLI	SST-2	Emo.	AG	Fin.	CoLA	BoolQ	Beav.	Code	Logic	Avg.
SFT	85.90	95.00	93.65	91.20	78.22	82.55	80.75	84.95	63.81	14.74	77.08
+ L2 Reg.	84.20	95.80	93.30	91.10	75.70	80.80	77.80	84.30	62.30	14.70	76.00
+ Random Edge	83.50	93.10	91.50	90.70	77.30	81.30	78.10	81.90	63.20	14.50	75.51
+ Neuron Pen.	72.40	67.30	83.40	73.85	53.09	76.13	67.30	71.60	66.27	13.33	64.47
IFSR	**90.63**	**96.50**	**93.95**	**92.85**	**83.51**	**84.28**	**86.35**	**86.05**	**66.04**	**15.09**	**79.53**

Table 2: Ablation study comparing IFSR against alternative regularization strategies. L2 Reg.: weight decay = 0.01. Random Edge: penalizing randomly selected edges with matched count and λ. Neuron Pen.: penalty on error neurons identified in activation space.

Train ↓ / Eval →	MNLI	SST-2	Emo.	AG	Fin.	CoLA	BoolQ	Beav.	Code	Logic
Base Model	30.85	76.60	28.60	28.40	28.99	65.10	58.10	43.80	40.86	8.59
MNLI	–	78.50	37.20	29.85	56.80	67.80	64.20	44.50	42.15	10.30
SST-2	35.20	–	**49.95**	30.85	61.21	48.90	**65.30**	50.05	**59.34**	12.46
Emotion	34.60	68.80	–	**31.80**	**62.76**	61.97	59.10	45.30	44.65	10.70
AG News	32.10	49.15	28.55	–	29.12	31.93	56.45	52.55	47.83	8.02
Finance	33.80	75.00	37.80	28.15	–	68.55	63.00	41.05	39.84	9.47
CoLA	32.40	**78.85**	32.20	29.80	57.09	–	57.75	43.35	39.95	8.60
BoolQ	33.60	74.05	35.55	29.25	57.22	65.29	–	42.30	41.48	8.72
BeaverTails	**35.40**	66.10	36.05	29.80	58.76	66.54	62.65	–	39.72	**13.86**
CodeXGLUE	31.50	73.65	28.50	23.95	22.76	64.53	53.80	**53.05**	–	9.12
Logic	34.80	74.95	35.35	29.85	58.76	**69.22**	62.60	41.75	39.48	–

Table 3: Cross-task generalization. Each row shows performance of an IFSR model (trained on the row task) evaluated on different tasks. Values above base model accuracy indicate positive transfer; most IFSR models maintain or improve base performance on unrelated tasks.

Structural Statistics Table 4 summarizes error edge statistics across tasks. Several patterns emerge: (1) Error edges predominantly connect features (F→F), with only 1–2.5% directly targeting logits (F→L). (2) Source features concentrate heavily in early layers (Layer 0–2), suggesting errors often originate in embedding-level representations. (3) A small number of high-degree nodes, which we term *hub features*, participate in disproportionately many error edges. For instance, in CoLA, a single Layer-0 feature serves as the source for 65.9% of all error edges.

Task	#Edges	F→L%	Src L0%
CoLA	352	2.0%	65.9%
Logic	101	2.0%	37.6%
AG News	622	1.1%	25.1%
Finance	294	1.4%	31.6%
BoolQ	175	2.3%	28.0%
CodeX	1,028	1.3%	18.2%

Table 4: Error edge statistics. F→L%: percentage of edges directly to logits; Src L0%: percentage of edges originating from Layer 0.

Interpretability Assessment We examine hub features on Neuronpedia (Lin, 2023) to assess whether error-contributing features have human-interpretable semantics. We find that most error-contributing features resist clear interpretation, even when with the label provided by automated interpretability tools.

Interpretable Case. In the Logic (fallacy detection) task, the top hub feature `F:13:9197`[1] is the target of 24% of error edges. Its positive logits strongly activate on preference-related vocabulary: *prefers* (0.95), *preference* (0.93), *prefer* (0.91), *inclined* (0.73). A plausible hypothesis: this feature may help detect "Appeal to Personal Preference" fallacies, but over-triggers on legitimate preference expressions, causing false positives.

Uninterpretable Case. In CoLA (grammaticality judgment), the dominant hub feature `F:0:7572`[2] serves as the source for 65.9% of error edges. Neuronpedia labels it "falt," but its activations are incoherent: Android lifecycle methods (`onCreateView`), C# initialization code (`BeginInit`), the Portuguese word "faltando," and the English preposition "of." Why an early-layer feature with such scattered activations dominates grammaticality errors remains opaque.

[1] https://www.neuronpedia.org/gemma-2-2b/13-clt-hp/9197

[2] https://www.neuronpedia.org/gemma-2-2b/0-clt-hp/7572

Error Propagation Chains We observe multi-hop error propagation chains spanning up to 4 layers. In Logic, one chain connects: `F:0:4812` ("searching/proposing") → `F:9:12977` ("posting information") → `F:13:9197` (preference words). While this may suggest an argumentative pattern: proposing claims → disseminating information → expressing preferences, we caution that such post-hoc narratives may be unfounded.

Summary While circuit-based error identification is tractable, most hub features resist human interpretation. This validates our hypothesis: the sparse feature space serves as a computational substrate for automatic error identification, independent of human understanding. The sparse feature space serves as a computational substrate for automatic error identification, not as a medium for human analysis. Our method succeeds precisely because it sidesteps the bottleneck of human interpretation, relying instead on statistical discrimination between correct and incorrect predictions. Additional case studies are provided in Appendix F.

4.6 Iterative Refinement

IFSR supports multi-round iterative refinement. Starting from an SFT checkpoint, we fine-tune the CLT using task data combined with general pretraining data, then re-identify error edges for subsequent rounds. CLT fine-tuning follows the setup of Karvonen (2025). Table 5 shows results over four iterations on four tasks.

Task	SFT	R1	R2	R3	R4
BoolQ	80.75	85.94	87.31	**88.17**	87.91
CodeXGLUE	63.81	65.79	66.40	**66.89**	66.75
MNLI	85.90	89.21	91.47	**92.74**	92.35
CoLA	82.55	83.77	**85.36**	85.23	85.34

Table 5: Iterative refinement over four rounds (R1–R4). CLT is fine-tuned between rounds to adapt to shifted activation distributions.

As shown in Table 5, all four tasks exhibit substantial and consistent performance gains through this iterative process. The most significant uplift is observed in the first two rounds, and performance generally peaks at R3. The diminishing but sustained gains suggest that each iteration reveals new error patterns masked by dominant errors.

Observations on Error Edge Evolution Under fixed frequency thresholds and attribution set size (500 samples), we observe that identifiable error edges decrease across rounds. This aligns with the intuition that IFSR progressively eliminates the most prominent error patterns. However, due to the substantial computational cost of CLT fine-tuning, we have not conducted extensive analysis across more tasks. These observations from four tasks may not generalize universally. Furthermore, since CLT fine-tuning alters feature semantics, we consider cross-round feature tracking to be of limited value: the same feature index may represent different computational roles after CLT adaptation.

5 Conclusion

We have presented Interpretable Feature-Space Regularization (IFSR), a framework that transforms mechanistic interpretability from post-hoc analysis into training-time optimization. Experiments across ten classification tasks demonstrate consistent gains. Ablations confirm that both sparse feature space and principled error edge identification are essential: random penalties and neuron-level interventions fail or harm performance. Cross-task evaluation shows that correcting error patterns can transfer positively to unrelated tasks.

A key finding of this work is that human interpretation of error patterns is unnecessary for effective correction. Most patterns identified by our method resist semantic interpretation, yet penalizing them still improves task performance. This validates our core hypothesis: the sparse feature space serves as a computational substrate for automatic error identification, independent of human understanding. Interpretability, in this framing, serves the model rather than human understanding.

Several directions remain for future work, including more sophisticated algorithms for error pattern discovery, more efficient transcoder adaptation methods, and application to broader tasks and architectures. This work points toward a potential new axis for scaling: as interpretability tools improve and feature spaces become richer, the capacity for automatic internal supervision may scale accordingly. Rather than training models solely on what outputs to produce, we can guide how they compute. We hope this encourages the community to view mechanistic interpretability not merely as a tool for understanding models, but as a foundation for improving them.

Limitations

Model Scope We evaluate IFSR exclusively on Gemma-2-2B, selected for its mature interpretability ecosystem with publicly available cross-layer transcoders. Whether our findings generalize to other architectures (e.g., Llama, Mistral) and scales remains to be validated as interpretability tools become available for more models.

Task Scope Our experiments focus on classification tasks where correctness is unambiguous. Extending IFSR to generation tasks requires defining "error patterns" for sequential outputs where multiple valid completions exist.

Computational Overhead IFSR introduces overhead for CLT inference and attribution graph construction during training. Single-round IFSR adds approximately 5–11% training time compared to standard SFT (see Appendix D for details). Iterative refinement requires additional CLT fine-tuning between rounds, which is computationally more expensive.

Attribution Set Requirements Our method requires held-out samples with both correct and incorrect predictions. For tasks where the initial model achieves very high or very low accuracy, obtaining sufficient error samples may be challenging.

Theoretical Understanding We demonstrate empirical effectiveness but lack theoretical guarantees on when IFSR improves over standard fine-tuning. The relationship between error edge structure and expected gains deserves deeper investigation.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*.

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *Preprint*, arXiv:1908.10063.

Seonglae Cho, Zekun Wu, and Adriano Koshiyama. 2025. Corrsteer: Generation-time llm steering via correlated sparse autoencoder features. *Preprint*, arXiv:2508.12535.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *Preprint*, arXiv:1905.10044.

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

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. 2024. Transcoders find interpretable llm feature circuits. *Preprint*, arXiv:2406.11944.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. Transformer Circuits Thread.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *Preprint*, arXiv:2406.04093.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Preprint*, arXiv:2307.04657.

Zhijing Jin, Abhinav Lalwani, Tejas Vaidhya, Xiaoyu Shen, Yiwen Ding, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, and Bernhard Schölkopf. 2022. Logical fallacy detection. *Preprint*, arXiv:2202.13758.

Xinyue Kang, Diwei Shi, and Li Chen. 2025. Model whisper: Steering vectors unlock large language models' potential in test-time. *Preprint*, arXiv:2512.04748.

Adam Karvonen. 2025. Revisiting end-to-end sparse autoencoder training: A short finetune is all you need. *Preprint*, arXiv:2503.17272.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. 2024. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *Preprint*, arXiv:2408.05147.

Johnny Lin. 2023. Neuronpedia: Interactive reference and tooling for analyzing neural networks. Software available from neuronpedia.org.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, and 8 others. 2025. On the biology of a large language model. *Transformer Circuits Thread*.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. *Preprint*, arXiv:2301.05217.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Preprint*, arXiv:2209.11895.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *Preprint*, arXiv:2407.14435.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Patrick Queiroz Da Silva, Hari Sethuraman, Dheeraj Rajagopal, Hannaneh Hajishirzi, and Sachin Kumar. 2025. Steering off course: Reliability challenges in steering language models. *Preprint*, arXiv:2504.04635.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

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

Haolei Xu, Xinyu Mei, Yuchen Yan, Rui Zhou, Wenqi Zhang, Weiming Lu, Yueting Zhuang, and Yongliang Shen. 2025. Easysteer: A unified framework for high-performance and extensible llm steering. *Preprint*, arXiv:2509.25175.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level convolutional networks for text classification. *Preprint*, arXiv:1509.01626.

Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Preprint*, arXiv:1909.03496.

A Attribution Weight Derivation

We provide a detailed explanation of how edge weights in attribution graphs are computed and why this formulation captures causal effects between features.

Local Linear Approximation In a transformer with cross-layer transcoders (CLTs), the activation of a downstream feature a_t depends on upstream feature activations through a complex computational graph. However, under certain linearization conditions, we can approximate this dependency locally.

Following Ameisen et al. (2025), we freeze attention patterns and layer normalization denominators during attribution. Combined with the fact that CLT features "bridge over" MLP nonlinearities, this makes the direct effect of one feature on another approximately linear at the current operating point. Specifically, near the current activation values, we have:

$$a_t \approx a_t^{(0)} + \sum_s \frac{\partial a_t}{\partial a_s} \cdot a_s \tag{9}$$

where $a_t^{(0)}$ is the value of a_t when all upstream features are zero, and the partial derivative $\frac{\partial a_t}{\partial a_s}$ represents the linear coefficient relating s to t.

Direct Effect Interpretation The edge weight $A_{s \to t} = \frac{\partial a_t}{\partial a_s} \cdot a_s$ can be interpreted as the *direct causal contribution* of node s to node t. The gradient term $\frac{\partial a_t}{\partial a_s}$ measures how sensitive t is to changes in s, while multiplying by a_s weights this sensitivity by the actual magnitude of s's activation. This ensures that:

- A feature with high sensitivity but near-zero activation contributes little;
- A strongly activated feature with low sensitivity also contributes little;
- Only features that are both sensitive *and* strongly activated have large effects.

Computation via Automatic Differentiation In practice, we compute these partial derivatives using automatic differentiation. For efficiency, rather than computing gradients for each pair of nodes separately, we perform a single backward pass from the target nodes (e.g., output logits) to obtain gradients with respect to all feature activations simultaneously:

$$\frac{\partial a_{\text{logit}}}{\partial a_s} = \texttt{autograd.grad}(a_{\text{logit}}, a_s) \tag{10}$$

For intermediate feature-to-feature edges, we can similarly compute gradients by treating downstream features as targets. The computational cost is linear in the number of layers times the number of target nodes, making it tractable for graphs with thousands of active features.

Relationship to Intervention-Based Attribution This gradient-based attribution can be viewed as a first-order approximation to intervention-based causal analysis. A true causal effect would require intervening on a_s (setting it to different values) and observing changes in a_t. The linear approximation $A_{s \to t}$ estimates this effect without requiring multiple forward passes, enabling efficient graph construction over large feature sets.

B Dataset Details

Table 6 summarizes the dataset statistics and partitioning for all ten tasks.

Data Partitioning Strategy For datasets with sufficient samples, we use up to 20,000 examples for training; for smaller datasets, we use all available training data. We reserve 10% of the data as the validation set for evaluation. The attribution set consists of 500 uniformly sampled examples, held out from both training and validation sets.

Multi-Round Attribution For the four tasks where we conduct multi-round iterative refinement (BoolQ, CodeXGLUE, MNLI, and CoLA), we sample a *fresh* set of 500 attribution examples for each round. This ensures that error edge identification in later rounds is not biased by patterns already penalized in earlier rounds.

C Hyperparameter Sensitivity

We sweep the penalty weight λ from 0.1 to 1.0 to study the sensitivity of IFSR to this hyperparameter. Table 7 presents results on four representative tasks spanning different domains and difficulty levels.

Optimal λ values consistently fall within the range $[0.5, 0.6]$, and performance degrades gracefully outside this range. This suggests that practitioners can safely use $\lambda = 0.5$ as a default without extensive tuning.

Task	Train	Val	Attr	Source
MNLI	20,000	2,000	500	GLUE
SST-2	20,000	2,000	500	GLUE
CoLA	8,551	1,043	500	GLUE
BoolQ	9,427	3,270	500	SuperGLUE
AG News	20,000	2,000	500	Zhang et al.
Emotion	16,000	2,000	500	SemEval
Finance	4,070	776	500	FPB
BeaverTails	20,000	3,021	500	PKU-A
CodeXGLUE	6,664	851	500	Devign
Logic	2,680	570	500	Jin et al.

Table 6: Dataset statistics. Train/Val/Attr denote the number of examples in training, validation, and attribution sets, respectively. FPB: FinancialPhraseBank; PKU-A: PKU-Alignment.

λ	MNLI	Finance	BoolQ	SST-2
0.1	89.85	82.47	86.10	96.15
0.3	90.21	82.99	86.20	96.35
0.5	**90.63**	83.25	**86.35**	**96.50**
0.6	90.45	**83.51**	86.15	96.40
0.8	90.12	83.12	85.80	96.25
1.0	89.78	82.73	85.55	96.10
SFT	85.90	78.22	80.75	95.00

Table 7: Test accuracy (%) across different λ values. Bold indicates best performance per task. All optimal values fall within $[0.3, 0.8]$, demonstrating low sensitivity to exact hyperparameter choice.

D Computational Overhead Analysis

We analyze the computational overhead of IFSR compared to standard SFT training. All experiments are conducted on 8× NVIDIA H800 GPUs with identical batch sizes and training configurations.

Training Time Table 8 summarizes the time overhead. IFSR introduces a modest slowdown of approximately 5–11% compared to standard SFT, primarily due to: (1) loading and processing error edge information, (2) computing the penalty term through CLT forward passes, and (3) additional gradient computation for the penalty loss.

Metric	IFSR	SFT
Training speed (it/s)	6.9–8.6	7.2–9.0
Relative slowdown	4–5%	
Total time overhead	5–11%	

Table 8: Training speed comparison between IFSR and standard SFT on 8× H800 GPUs.

Memory Requirements IFSR requires additional memory for: (1) storing error edge indices and weights, (2) maintaining intermediate activations for penalty gradient computation, and (3) CLT encoder parameters (frozen). We estimate the additional memory overhead at 10–30% depending on the number of error edges. In practice, IFSR training fits within the same GPU memory constraints as SFT for all experiments.

Attribution Graph Construction The one-time cost of constructing attribution graphs on the 500-sample attribution set is approximately 15–30 minutes per task, depending on sequence length. This cost is amortized across training and does not affect per-iteration training speed.

E Neuron-Level Ablation Details

Table 9 presents the top-10 error neurons identified for each task, ranked by contrast ratio (mean activation on incorrect samples divided by mean activation on correct samples). Across all tasks, even the highest-contrast neurons show ratios below 3.5×, with most tasks exhibiting maximum ratios between 1.5× and 2.8×. This confirms that error patterns are diffusely encoded across the dense neuron space, making targeted intervention infeasible without sparse decomposition.

Layer Distribution Error neurons are distributed across all transformer layers (0–25 for Gemma-2-2B), with no clear concentration in early, middle, or late layers. This further supports the hypothesis that error-inducing computations are not localized to specific model components.

Comparison with Sparse Features In contrast to the modest neuron contrast ratios (1.4–3.4×), error edges in the sparse feature space can exhibit substantially higher selectivity, as CLT features are trained to capture monosemantic concepts. This difference explains why feature-level intervention succeeds where neuron-level intervention fails.

F Error Edge Case Studies

This appendix provides additional analysis of error edges and hub features across tasks.

F.1 Hub Feature Summary

Table 10 summarizes the top hub features across tasks with their Neuronpedia interpretations and our interpretability assessment.

Rank	MNLI	SST-2	Emotion	AG News	Finance	CoLA	BoolQ	Beaver.	CodeX	Logic
1	1.73	2.81	1.92	1.74	2.73	1.91	1.50	1.55	2.84	3.41
2	1.69	2.75	1.91	1.70	2.52	1.90	1.50	1.52	2.45	2.36
3	1.65	2.70	1.88	1.66	2.50	1.83	1.49	1.50	2.44	2.02
4	1.62	2.63	1.80	1.64	2.41	1.80	1.47	1.48	2.33	1.93
5	1.59	2.61	1.77	1.61	2.39	1.72	1.46	1.47	2.27	1.93
6	1.56	2.58	1.73	1.59	2.34	1.64	1.45	1.46	2.27	1.92
7	1.53	2.50	1.71	1.56	2.34	1.64	1.43	1.45	2.16	1.88
8	1.50	2.48	1.71	1.56	2.33	1.63	1.42	1.45	2.14	1.82
9	1.48	2.47	1.71	1.55	2.31	1.63	1.42	1.45	2.13	1.75
10	1.45	2.42	1.71	1.55	2.31	1.62	1.41	1.45	2.08	1.74
Max	1.73	2.81	1.92	1.74	2.73	1.91	1.50	1.55	2.84	3.41

Table 9: Contrast ratios of top-10 error neurons per task. Values represent the ratio of mean absolute activation on incorrectly classified samples to correctly classified samples. Even the highest-ranked neurons show modest contrast (typically $< 3\times$), indicating diffuse error encoding in dense neuron space.

Task	Feature	Label	Interp.	Notes
CoLA	F:0:7572	"falt"	$\times$	Activates on code, Portuguese, prepositions; 65.9% of edges
CoLA	F:3:5222	"mes"	$\times$	Medical terms + code; unclear grammar relevance
Logic	F:13:9197	preference	✓	Clear preference vocabulary; plausible fallacy connection
Logic	F:0:4812	"searching"	$\sim$	Proposing/searching verbs; serves as chain source
AG News	F:2:4725	"appropriate"	$\sim$	Evaluative adjective; unclear news category link
AG News	F:1:10855	"mark"	$\times$	Person/place names; no clear news classification relevance
Finance	F:6:8804	"calendar"	$\sim$	Date/time expressions; possible sentiment interference
Finance	F:3:5691	"choices"	✓	Decision vocabulary (option, risk); common in financial text
BoolQ	F:10:5154	"measurements"	✓	Numbers/percentages; numerical reasoning difficulty
CodeX	F:9:14755	"works"	$\sim$	Functionality semantics; defect detection confusion

Table 10: Hub feature analysis across tasks. ✓ = interpretable with plausible task connection; $\sim$ = has label but task connection unclear; $\times$ = uninterpretable or incoherent activations.

F.2 Finance Sentiment: Decision Vocabulary

The Finance task provides an instructive case where feature semantics partially explain errors. The hub feature `F:3:5691`[3] (labeled "choices") activates on decision-related abstract concepts:

- *option* (17.50), *decision* (17.00), *assumption* (15.44)
- *risk* (14.81), *right* (14.56), *principle* (14.31)
- *opportunity* (13.81), *advantage* (11.69), *liberty* (12.06)

In financial texts, words like "option" (stock options), "risk," and "opportunity" are domain-specific but sentiment-neutral. The model may incorrectly associate these decision-related terms with positive or negative sentiment, leading to misclassification. This represents a case where interpretable feature semantics provide a plausible (though unverified) explanation for task errors.

F.3 AG News: Token-Level Patterns

In contrast, the AG News hub feature `F:1:10855`[4] (labeled "mark") exemplifies features with clear activation patterns but no obvious task relevance. It activates on:

- Personal names: Mark Twain, Mark Wahlberg
- Geographic names: Finnmark, Denmark, Telemark

[3] https://www.neuronpedia.org/gemma-2-2b/3-clt-hp/5691

[4] https://www.neuronpedia.org/gemma-2-2b/1-clt-hp/10855

- Words containing "mark": unremarkable, supermarket

This feature participates in 276 error edges (second-highest for AG News), yet the connection between the token "mark" and news topic classification remains opaque. Such cases illustrate that even features with coherent activation patterns may participate in errors through mechanisms that resist human interpretation.

F.4 Error Chain Structure

Beyond individual features, we observe that errors propagate through multi-hop chains in the computational graph. The longest observed chain (4 hops) occurs in Logic:

$$\texttt{F:0:4812} \xrightarrow{\text{L0}\to\text{L9}} \texttt{F:9:12977} \xrightarrow{\text{L9}\to\text{L13}} \texttt{F:13:9197} \xrightarrow{\text{L13}\to\text{L14}} \texttt{F:14:11674}$$

The semantic labels of these features suggest a narrative:

- **F:0:4812** (Layer 0): "searching and proposing"—activates on verbs like *searched*, *propose*, *examined*
- **F:9:12977** (Layer 9): "posting information"—activates on *posting*, *updates*, *announcing*
- **F:13:9197** (Layer 13): preference vocabulary (discussed in main text)

One interpretation: this chain represents an argumentative structure where claims are proposed, information is disseminated, and preferences are expressed—patterns that might co-occur with certain fallacy types. However, we emphasize this is speculative; establishing true causal mechanisms would require intervention experiments beyond our current scope.

F.5 Interpretability Limitations

Our analysis reveals several systematic challenges in interpreting error-contributing features:

1. **Label-behavior mismatch**: Automated interpretability labels often describe surface token patterns rather than computational functions
2. **Post-hoc rationalization**: Connecting feature semantics to task-specific errors invites unfounded speculation
3. **Early-layer dominance**: Many error sources are Layer 0–2 features with mixed or incoherent activation patterns
4. **Polysemantic features**: Despite CLT training, some features still activate on multiple unrelated patterns

These findings reinforce the central claim of IFSR: effective error correction does not require human understanding of error mechanisms. Statistical identification of error-correlated features suffices for targeted intervention, even when the "why" remains opaque.