Verifying Chain-of-Thought Reasoning via Its Computational Graph

Zheng Zhao[1,3], Yeskendir Koishekenov[1], Xianjun Yang[2], Naila Murray[1], Nicola Cancedda[1]

[1]FAIR at Meta [2]Meta Superintelligence Labs [3]University of Edinburgh

Overview

- We introduce **Circuit-based Reasoning Verification (CRV)**, a **white-box approach** for checking whether a reasoning step is correct by inspecting its underlying **computational graph**.
- CRV replaces MLPs with **interpretable transcoders**, constructs **attribution graphs** for individual Chain-of-Thought steps, and extracts **structural graph features** that capture the computation behind each step.
- Using this framework, we show that reasoning errors leave **distinct structural fingerprints**: these signals improve verification, vary across domains, and can guide **causal interventions** that fix incorrect reasoning trajectories.

Motivation

- Chain-of-Thought (CoT) reasoning is powerful, but its intermediate steps are not always reliable.
- Models can produce explanations that appear coherent while still making **arithmetic, logical, or order-of-operations errors**.
- In some cases, the CoT itself may be **unfaithful**: for example, the model may say it used a calculator when no such computation actually occurred, making the explanation a post-hoc justification rather than a faithful account of the decision process.
- This makes **step-level verification** essential: we want to know whether a CoT step is correct, and why it failed when it is not.

Why Another Verifier?

- Existing CoT verification methods are mostly **black-box** or **gray-box**: they score the generated text, output probabilities, or hidden states to predict whether a reasoning step is likely to be correct.
- These approaches can often detect that a step is suspicious, but they provide limited insight into **why the underlying computation failed**. In particular, they do not directly expose the model components and information flow responsible for the error.
- This limitation is especially important when CoT is **unfaithful**: a model may produce a plausible explanation or even claim to have used an external tool, while the actual internal computation follows a different path.
- We therefore seek a **white-box verifier** that operates on the model's **computational graph**, allowing us not only to detect incorrect reasoning steps, but also to analyze their **structural signatures** and identify where the computation breaks down.

Building Step-Level Supervision

- We generate CoT traces from our target model and assign a **correct / incorrect label to each reasoning step**.
- We use two controlled synthetic domains, **Boolean** and **Arithmetic**, where intermediate steps can be verified automatically.
- For synthetic tasks, labels are kept only when a **programmatic verifier** and an **LLM judge** agree; for **GSM8K**, we use an **LLM judge** with full problem context.
- To avoid ambiguous supervision, we **truncate after the first incorrect step**.

Boolean example

Expression: `((True or False) and True)`

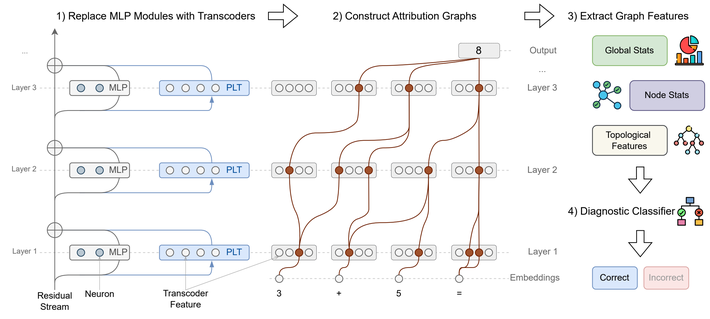
Step 1: `True or False = True`	correct
Step 2: `True and True = True`	correct

Arithmetic example

Expression: `(7*((5+9)+7))`

Step 1: `5 + 9 = 14`	correct
Step 2: `7 * 14 = 98`	incorrect

Circuit-based Reasoning Verification



1. Replace each MLP with a **transcoder** so reasoning flows through interpretable features.
2. Build a **step-level attribution graph**.
3. Extract **global**, **node-level**, and **topological** features from the pruned graph.
4. Train a diagnostic classifier on these graph fingerprints to predict whether the step is correct or incorrect.

Experimental Setup

- **Task:** predict whether an individual Chain-of-Thought step is correct or incorrect.
- **Model:** Llama 3.1 8B Instruct, modified with trained **per-layer transcoders** to enable attribution-graph analysis.
- **Datasets:** **Boolean**, **Arithmetic**, and **GSM8K** with step-level correctness labels.
- **Baselines:** black-box logit-based scores and gray-box hidden-state verifiers, including **CoE**, **CoT-Kinetics**, and probing-based methods.
- **Metrics:** we report **AUROC**, **AUPR**, and **FPR@95**, treating the **incorrect step** as the positive class.

Verification Performance

CRV consistently outperforms black-box and gray-box baselines across Boolean, Arithmetic, and GSM8K.

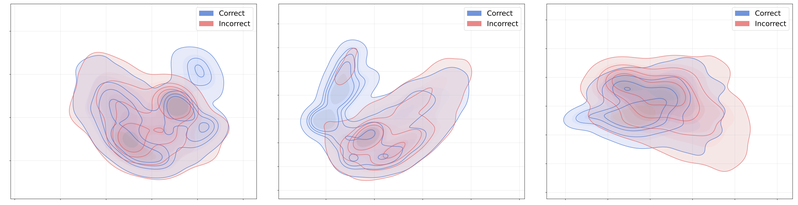
Paradigm	Method	Synthetic (Boolean)			Synthetic (Arithmetic)			GSM8K		
		AUROC ↑	AUPR ↑	FPR@95 ↓	AUROC ↑	AUPR ↑	FPR@95 ↓	AUROC ↑	AUPR ↑	FPR@95 ↓
Black-Box	MaxProb	<u>58.81</u>	0.34	95.20	61.87	1.81	84.98	54.91	7.99	91.86
	PPL	57.37	0.29	91.02	60.19	1.68	85.52	55.46	8.12	90.69
	Entropy	53.56	0.24	97.55	60.03	1.52	85.40	56.67	7.29	87.08
	Temp. Scaling	58.77	0.36	91.41	59.67	1.66	86.96	54.42	8.24	92.28
	Energy	51.08	0.28	95.11	<u>76.45</u>	<u>5.59</u>	73.86	<u>62.55</u>	9.11	<u>86.34</u>
Gray-Box	CoE-R	53.17	0.33	92.85	58.47	1.93	76.68	52.38	8.34	96.20
	CoE-C	51.03	<u>0.38</u>	92.07	69.39	3.03	<u>63.33</u>	53.57	<u>10.80</u>	96.33
	CoT-Kinetics	53.62	0.24	97.13	60.83	1.58	85.09	56.54	7.35	86.83
	LR Probe	52.91	0.25	<u>88.42</u>	54.22	1.50	91.90	55.86	7.99	90.32
	MLP Probe	53.63	0.26	88.56	54.41	1.30	90.98	56.02	8.63	93.94
White-Box	CRV (Ours)	**75.87**	**0.97**	**79.17**	**92.47**	**28.92**	**37.09**	**70.17**	**14.3**	**79.61**

Table 1. Arrows indicate preferred direction (↑ higher is better, ↓ lower is better). **Best** and <u>second-best</u> results are highlighted for each metric. The low AUPR on the Boolean dataset reflects extreme label imbalance, with the incorrect label only 0.2%.

Structural Signatures of Failure

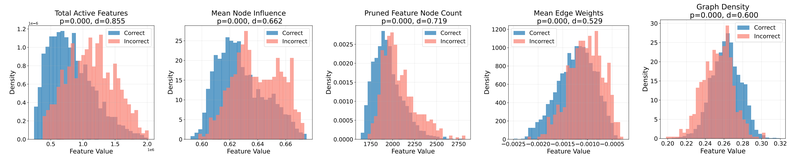
To move beyond aggregate accuracy, we analyze *how* correct and incorrect reasoning steps differ in graph-feature space. The figures below show that reasoning failures are not random: they leave **consistent structural signatures** in the attribution graph.

PCA of full graph fingerprints across domains



Boolean Arithmetic GSM8K

Feature-level fingerprints of error on GSM8K



From Diagnosis to Causal Intervention

A key advantage of CRV is that it localizes failure signals to specific transcoder features in the attribution graph. This enables a direct **intervention**: we can modify the activation of a suspected feature at generation time and test whether the reasoning trajectory changes.

Before Intervention (Incorrect)	After Intervention (Correct)
Evaluate the arithmetic expression below.	Evaluate the arithmetic expression below.
(7 * ((5 + 9) + 7))	(7 * ((5 + 9) + 7))
To evaluate this expression, we need to follow the order of operations (PEMDAS):	To evaluate this expression, we need to follow the order of operations (PEMDAS):
1. Evaluate the expression inside the innermost parentheses: 5 + 9 = 14	1. Evaluate the expression inside the innermost parentheses: 5 + 9 = 14
2. Multiply 7 by the result: 7 * 14 = 98	2. Add 7 to the result: 14 + 7 = 21
3. Add 7 to the result: 98 + 7 = 105	3. Multiply 7 by the result: 7 * 21 = 147
Therefore, the expression evaluates to 105.	Therefore, the value of the expression is 147.

Table 2. Reasoning trace before and after causal intervention. The divergence shows that suppressing a single *multiplication* transcoder feature steers the model onto the correct computational path.

Takeaway. This experiment provides **causal evidence** that CRV is detecting more than a correlational pattern. The structural signatures identified in the attribution graph can guide targeted interventions on specific transcoder features, turning verification into a first step toward **interpretable debugging and repair**.

Scan Me Please!

(a) Paper (b) Data/Code (c) Connect with Me

UK Research and Innovation

ICLR 2026, Rio de Janeiro, Brazil

UKRI CENTRE FOR DOCTORAL TRAINING