LOCAL CAUSAL ATTRIBUTION OF CHAIN-OF-THOUGHT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding the structure of a language model's thought process is an important problem for both transparency and safety. In this work, we take a local and causal approach toward this goal by analyzing the causal relationships among components, termed units, of a given, specific chain-of-thought trace. We construct a structural causal model on these units and relate each unit to the log probability of generating subsequent output units. Our AttriCoT algorithm performs attribution by estimating importance parameters in this structural causal model, using $O(U)$ forward passes to compute log probability effects, where U is the number of units. Evaluation of perturbation curves across 5 datasets and 4 reasoning models shows that AttriCoT produces attributions that are a more faithful local explanation of the model's behavior than alternative methods. The attribution results also point to differences in thought structure between models and domains.

1 INTRODUCTION

The success of LLMs and large reasoning models (LRMs) in performing reasoning is due in large part to their generation of sequences of reasoning steps known as chains-of-thought (CoTs). Moreover, because of the readability of their text, CoTs appear to promise the additional benefit of greater insight into the model's reasoning. It is perhaps paradoxical then that the introduction of CoTs has raised more questions than it has answered. Prominent among these questions is that of *CoT faithfulness*, which refers to how well the verbalized CoT text represents the model's latent thought process. Research has shown that CoT text may disregard manipulation of the model's input or injection of helpful hints, even constructing elaborate but false justifications when given incorrect hints (Turpin et al., 2023; Chen et al., 2025; Arcuschin et al., 2025), with evidence that larger models may rely *less* on their stated reasoning than smaller models (Lanham et al., 2023). On the other hand, Von Arx & Deng (2025) illustrate that CoTs may become more faithful for more difficult tasks.

Beyond the debate about faithfulness of CoT *text*, there exist questions about the *mechanisms* that underlie CoTs. Recent work on CoT faithfulness (Barez et al., 2025; Zaman & Srivastava, 2025) has also moved toward this direction, arguing that progress in faithfulness can be achieved via *causal* CoTs that demonstrably affect the model's final answer. These questions, about mechanisms and improving faithfulness, motivate investigation of the causal structure within a CoT trace. The focus of study thus becomes analyzing the impact of earlier thoughts/steps on later ones. Understanding this structure can provide insights into why certain steps are produced, whether intermediate reasoning is consequential, and where/when unfaithfulness may arise.

Causal structure within a CoT can be investigated in terms of effects at different levels. First, there is the *model-internal* level, studying for example how the internal representation of a thought is affected by those of previous thoughts, which we do not tackle in this work. Then there is the level of causal effects on *distributions* over reasoning steps or trajectories. This level involves resampling of CoTs and was explored recently by Macar et al. (2026); Bogdan et al. (2025). Our paper focuses on a third level that we call *local causality*, where "local" refers to restriction to a single, given CoT. This notion of local causality is useful for explaining effects on specific, given CoT steps (that a user might see), without reference to alternative steps or distributions over steps. Moreover, it offers a more computationally efficient approach to understanding CoT structure, requiring only forward passes through the model as opposed to the resampling of Macar et al. (2026); Bogdan et al. (2025), which is significantly more expensive.

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.

We develop a method called *AttriCoT* for estimating causal relationships at the local level, within a given CoT. We generalize the concept of a CoT step to that of a *unit*, i.e., user-configurable segments of the model output as well as the model input. For each unit in the model output, AttriCoT estimates the effects that prior units have on the log probability of generating that unit, thereby attributing it to prior units. AttriCoT works by 1) performing interventions on units, 2) computing forward passes to determine the log probability effects of interventions, and 3) fitting the observed effects to a linear structural equation model. Computational tractability is maintained by restricting the number of forward passes to scale linearly with the number of units. The result can be represented as a matrix of pairwise importance scores, where element (i, j) represents influence of unit i on unit j. Such characterization of causal effects between all pairs of CoT steps has been less studied to date.

Our focus on locality aligns with and extends an established line of work on *local explanations*, which aim to explain specific outputs of ML models on specific inputs, ranging from predictions of classifiers (Ribeiro et al., 2016; Lundberg & Lee, 2017) to responses of LLMs (Sarti et al., 2023; Miglani et al., 2023; Enouen et al., 2024; Cohen-Wang et al., 2024; Monteiro Paes et al., 2025). We extend local explanations to a setting where there is no clear distinction between inputs and outputs, since LLMs consume their generated CoTs as input.

We evaluate AttriCoT's attributions in terms of their behavioral faithfulness as a local explanation of the LRM that generated the CoT. Specifically, we compute perturbation curves that measure the alignment between estimated importance scores and log probability effects on the given output units produced by the LRM. The evaluation spans 4 LRMs and 5 datasets, and compares against the sentence-to-sentence masking method of Thought Anchors (Bogdan et al., 2025) and prompting baselines. AttriCoT consistently outperforms these baselines as measured by the area under the perturbation curve (AUPC). For instance, on MATH500, AttriCoT achieves 8-16% and 45-87% AUPC improvement relative to Thought Anchors and prompting baselines across models, respectively. Furthermore, we propose metrics for analyzing AttriCoT's importance score matrices by summarizing the matrix along different directions. These metrics point to differences in CoT structure between models and datasets, and we believe that this manner of analysis holds significant potential.

Our **contributions** can be summarized as follows:

(**1**) We propose AttriCoT, a local explanation method for attributing CoT steps and other output units (e.g., of the answer) to prior units, based on a structural causal model and using a linear number of forward passes.
(**2**) We show that AttriCoT's attributions provide a consistently more faithful local explanation of the model's behavior than alternative methods.
(**3**) We show that metrics computed from AttriCoT's attribution matrices can open the door toward greater insights into CoTs.

2 Related Work

CoT attribution methods. Various methods have recently been proposed for the attribution analysis of CoTs. To frame our comparison, we position related work along three axes: causality, granularity, and access level (see Table 3 in Appendix A for a detailed comparison). Broadly, perturbation-based methods (Turpin et al., 2023; Atanasova et al., 2023; Parcalabescu & Frank, 2024) measure correlations with respect to the final output, but do not establish causality. White-box causal methods (Zhao et al., 2026; Ye et al., 2026) aim to reveal causal structures but require access to the model's internals. Black-box step-level causal methods (Paul et al., 2024; Macar et al., 2026) are the most relevant category to our work. They perturb individual steps (without access to the model's internals) and measure the attribution as the effect on the final answer. This does have limitations, however, in that the impacts of intermediate steps are not accounted for.

Beyond answer-level attribution. To the best of our knowledge, Thought Anchors (Bogdan et al., 2025), and specifically their sentence-to-sentence masking method (Sec. 5, App. M therein), is the only work that moves beyond answer-level attribution. Thought Anchors attributes every unit in the CoT to prior ones by masking each sentence and performing a forward pass to determine the effects. It measures changes in token distributions using Kullback-Leibler (KL) divergence at every token position in a target CoT unit. In contrast, AttriCoT measures the change in log probability of the

given generated unit, which is fully local in the sense discussed in the introduction. Additionally, AttriCoT fits a structural causal model and can thus exploit information from any intervention type.

3 A Structural Causal Model for Chain-of-Thought

In this section, we present a structural causal model (SCM) for the steps in an LRM's CoT, allowing us to define the causal effect of one CoT step on another and thereby attribute steps to earlier steps.

Let $\boldsymbol{x}$ be a given input, e.g., prompt, to the LRM and $\boldsymbol{y}$ be the corresponding output, e.g., solution, where $\boldsymbol{y}$ consists of both the LRM's CoT and the final answer. We assume that $\boldsymbol{y}$ has been split into what we generally call *units*, $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$. These include steps or "thoughts" in the CoT, followed by the final answer, which may also be split into multiple units (our notation does not distinguish between CoT units and answer units). We consider the input $\boldsymbol{x}$ to be split into units $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S)$ as well. For example, $\boldsymbol{x}$ may be a question and $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S)$ may be sentences in the question that give different pieces of information. We will only consider attributing output units $\boldsymbol{y}_t$ to other units, i.e., we will not model the causes of input units $\boldsymbol{x}_s$. We use the symbols $\boldsymbol{x}_s$, $\boldsymbol{y}_t$ to refer to units at a more abstract level as well as the sequences of tokens that they comprise.

In the classical setting of an SCM with scalar variables $(x_1, \ldots, x_S, y_1, \ldots, y_T)$, the model is specified by structural equations, one for each variable (Pearl, 2009). For example, the structural equation for y_t takes the general form

$$y_t = F_t(\boldsymbol{x}, \boldsymbol{y}_{<t}, \boldsymbol{u}), \tag{1}$$

where we assume an auto-regressive property that mirrors how LLMs/LRMs operate, $\boldsymbol{y}_{<t} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ is the set of output units that causally precede y_t, and $\boldsymbol{u}$ represents exogenous variables. In the LRM setting however, all variables are natural language units, which have representations as sequences of tokens. In particular, we now have such a sequence $\boldsymbol{y}_t$ on the left side of (1). We would like F_t to remain a scalar, real-valued function to quantify the influence that earlier units have on $\boldsymbol{y}_t$. We propose therefore to "scalarize" the unit $\boldsymbol{y}_t$ with a *scalarizer* function S_t, yielding

$$S_t(\boldsymbol{y}_t) = F_t(\boldsymbol{x}, \boldsymbol{y}_{<t}, \boldsymbol{u}). \tag{2}$$

We choose S_t to be the log probability of the token sequence of $\boldsymbol{y}_t$ conditioned on previous units:

$$S_t(\boldsymbol{y}_t) = \log P(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) = \sum_{j=1}^{|\boldsymbol{y}_t|} \log P(y_{t,j} \mid \boldsymbol{x}, \boldsymbol{y}_{<t}, \boldsymbol{y}_{t,<j}), \qquad t = 1, \ldots, T, \tag{3}$$

where we have used the chain rule of probability to express the log probability as a sum of log probabilities of tokens $y_{t,j}$ in $\mathbf{y}_t$. The use of log probability (3) allows causal effects to be measured more efficiently, using non-generative forward passes only, and without reference to alternatives to the given unit $\boldsymbol{y}_t$, reflecting our desire for locality.

For the function F_t, we propose that it be an interpretable model, in line with our goal of providing local explanations. In this work, we use a linear model with "interpretable features" that indicate whether units are present or absent (replaced by the empty sequence). Let $\mathbb{I}(\boldsymbol{x})$ denote such an indicator function (equals 1 if unit $\boldsymbol{x}$ is present and 0 if absent). The linear model is given by

$$F_t(\boldsymbol{x}, \boldsymbol{y}_{<t}, \boldsymbol{u}) = \sum_{s=1}^{S} \alpha_{s,t} \mathbb{I}(\boldsymbol{x}_s) + \sum_{\tau=1}^{t-1} \beta_{\tau,t} \mathbb{I}(\boldsymbol{y}_\tau) + \gamma_t + u_t, \qquad t = 1, \ldots, T, \tag{4}$$

with coefficients $\alpha_{s,t}$ and $\beta_{\tau,t}$ and intercept γ_t. Other interpretable models could also be used, for example those involving Boolean conjunctions of $\mathbb{I}(\boldsymbol{x}_s), \mathbb{I}(\boldsymbol{y}_\tau)$ (Wei et al., 2019), which could account for interactions.

Equations (2)–(4) together attribute the conditional log probability of output unit $\boldsymbol{y}_t$ to the presence or absence of prior units. We therefore interpret coefficients $\alpha_{s,t}$ and $\beta_{\tau,t}$ as importance scores for generating $\boldsymbol{y}_t$. (The intercept γ_t also has an interpretation as the log probability of $\boldsymbol{y}_t$ with an empty prefix.) For future use, we collect all $\alpha_{s,t}$ into a matrix $\boldsymbol{A} \in \mathbb{R}^{S \times T}$, and $\beta_{\tau,t}$ into a *strictly upper-triangular* matrix $\boldsymbol{B} \in \mathbb{R}^{T \times T}$, i.e., $\beta_{\tau,t} = 0$ for $\tau \geq t$. The upper-triangular property of $\boldsymbol{B}$ is due to the auto-regressive property of LRMs for the case of the linear structural equations in (4).

4 ATTRICOT

We now describe our proposed AttriCoT method for attributing CoT steps and other output units to prior units. AttriCoT performs this attribution by fitting the structural equations in (4) via linear regression, thereby estimating the local causal effects of prior units. The process involves three steps: 1) Selecting and performing interventions on prior units, which corresponds to defining the feature matrix for the linear regressions; 2) Measuring the log probabilities of output units that result from interventions, which corresponds to obtaining the target values; and 3) Actually performing the linear regressions. These steps are depicted in Figure 1 and described in more detail below.

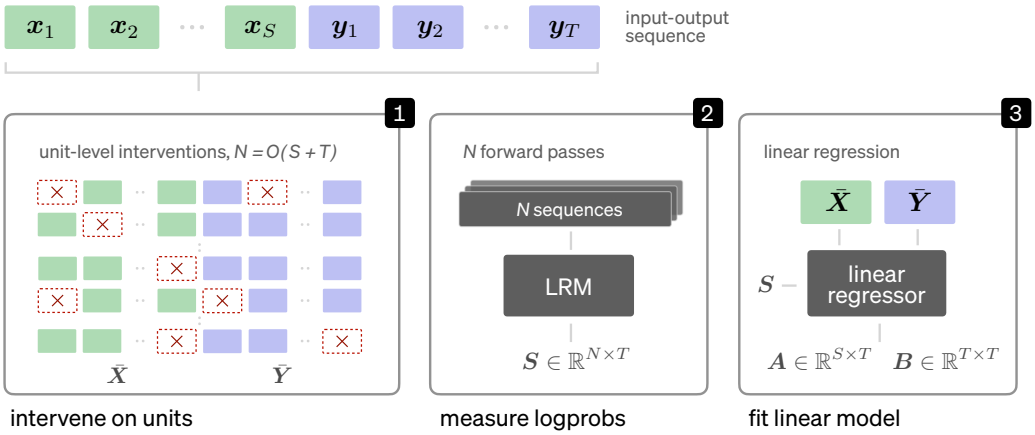


Figure 1: The AttriCoT method consists of three main steps: performing unit-level interventions (1), measuring the impact of the interventions on log probabilities (2), and estimating the attribution scores by fitting a linear model (3).

1 Sample and perform interventions. Given an input-output sequence of units $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$, we perform interventions on units to determine their causal effects. We denote by $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}), i = 1, \ldots, N$ the perturbed sequences resulting from intervention, and also define $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}) = (\boldsymbol{x}, \boldsymbol{y})$ to be the original sequence for $i = 0$. This intervention step involves making three choices: 1) the number of interventions N, 2) which units to intervene on, and 3) how to intervene. For 3), AttriCoT simply removes the selected units from the sequence, consistent with setting the indicator functions in (4) to zero. We investigate an alternative masking perturbation in Appendix D.2.1. For 1), we restrict N to be a multiple of the total number of units $S + T$. As described further in Step 2, N is also the number of forward passes through the LRM, so choosing $N = O(S + T)$ helps to control this computational cost.

A basic intervention strategy with $N = S+T$ is the *leave-one-out* (LOO) procedure, which removes each unit $\boldsymbol{x}_s$ or $\boldsymbol{y}_t$ one at a time. This yields perturbed sequences

$$\left(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\right) = \begin{cases} (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_S, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T), & i = 1, \ldots, S, \\ (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-S-1}, \boldsymbol{y}_{i-S+1}, \ldots, \boldsymbol{y}_T), & i = S+1, \ldots, S+T. \end{cases} \tag{5}$$

We also experiment with "2x" variants that use $N = 2(S + T)$ interventions. These are described in Section 5.1 and Appendix C.3.1, where we also further discuss the effects of interventions.

2 Measure log probabilities. The second step in AttriCoT is to measure the scalarized effects on downstream units (i.e., the changes in their log probabilities) of the interventions of the first step. Each perturbed sequence $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ is passed as input to the LRM and a forward pass is performed to compute the conditional log probabilities of all tokens, $\log P(y^{(i)}_{t,j} \mid \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}_{<t}, \boldsymbol{y}^{(i)}_{t,<j})$. For the original sequence $i = 0$, these log probabilities are either pre-computed and given as input to AttriCoT, or we perform one additional forward pass on $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)})$. The log probabilities are then aggregated over the tokens in each output unit $\boldsymbol{y}_t$, where instead of the sum in (3), we take the

mean to normalize by the length of the unit. The result is

$$S_t\left(\boldsymbol{y}_t^{(i)}\right) = \frac{1}{\left|\boldsymbol{y}_t^{(i)}\right|} \sum_{j=1}^{|\boldsymbol{y}_t^{(i)}|} \log P\left(y_{t,j}^{(i)} \mid \boldsymbol{x}^{(i)}, \boldsymbol{y}_{<t}^{(i)}, \boldsymbol{y}_{t,<j}^{(i)}\right), \qquad i = 0, \ldots, N,\ t = 1, \ldots, T. \quad (6)$$

The above computation requires $O(N)$ forward passes for N interventions, as well as access to output log probabilities or logits. To increase computational efficiency, these forward passes should be batched. It is worth noting that AttriCoT attributes all T output units using $O(N) = O(S+T)$ forward passes, i.e., the inference cost is linear in the number of units (not quadratic).

3 Fit linear model. The third step is to perform linear regression to estimate the coefficients $\alpha_{s,t}$ and $\beta_{\tau,t}$ in (4), using the indicator function feature values from the first step and the log probability target values from the second step. To specify the objective function for these linear regressions, we first collect the indicator values for input units into a feature matrix $\bar{\boldsymbol{X}} \in \mathbb{R}^{(N+1)\times S}$ with entries $\bar{x}_{i,s} = \mathbb{I}(\boldsymbol{x}_s^{(i)})$ (including $i = 0$ which yields an all-ones row), and those for output units into another matrix $\bar{\boldsymbol{Y}} \in \mathbb{R}^{(N+1)\times T}$ with entries $\bar{y}_{i,t} = \mathbb{I}(\boldsymbol{y}_t^{(i)})$. We similarly define a target matrix $\boldsymbol{S} \in \mathbb{R}^{(N+1)\times T}$ via $S_{i,t} = S_t(\boldsymbol{y}_t^{(i)})$ from (6). Then for output unit $\boldsymbol{y}_t$, we regress the tth target column $\boldsymbol{S}_{:,t}$ on $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ to obtain the tth columns of $\boldsymbol{A}$ and $\boldsymbol{B}$. However, we also need to exclude rows (interventions) i where $\boldsymbol{y}_t$ itself was removed, since it is not possible to compute the log probability of $\boldsymbol{y}_t$ in those cases. Let $\mathcal{I} = \{i : \bar{y}_{i,t} = 1\}$ be the set of rows where $\boldsymbol{y}_t$ is preserved. The final form of the regression problem (with least-squares loss) is therefore

$$\min_{\boldsymbol{A}_{:,t}, \boldsymbol{B}_{<t,t}, \gamma_t} \left\| \bar{\boldsymbol{X}}_{\mathcal{I},:} \boldsymbol{A}_{:,t} + \bar{\boldsymbol{Y}}_{\mathcal{I},<t} \boldsymbol{B}_{<t,t} + \gamma_t \mathbf{1} - \boldsymbol{S}_{\mathcal{I},t} \right\|_2^2. \quad (7)$$

Regularization could be added if desired, for example ℓ_1 regularization $\propto (\|\boldsymbol{A}_{:,t}\|_1 + \|\boldsymbol{B}_{<t,t}\|_1)$.

Appendix B describes a white-box method that we explored to further reduce computational cost.

5 Evaluation of Attribution Quality

We present experiments that evaluate the quality of AttriCoT's attributions compared to baselines.

5.1 Experimental Setup

Datasets. We selected five datasets for the evaluation. We chose two complementary mathematical reasoning datasets, GSM8K (Cobbe et al., 2021) and MATH500 (Hendrycks et al., 2021). In the domain of general reasoning and knowledge, we selected two more datasets, namely GPQA-Diamond (Rein et al., 2024), including PhD-level questions in physics and biology, and MMLU-Pro (Wang et al., 2024), including 14 different domains. We included a third task of reasoning on logic problems, represented by the ZebraLogic (Lin et al., 2025) benchmark. Table 4 in Appendix C further describes the datasets.

Models. Reasoning traces were generated using four open-source reasoning models. We chose `DS-Qwen3-8B` (DeepSeek-AI, 2025; Team, 2025) for its good performance given its size, `DS-Llama-8B` (DeepSeek-AI, 2025; et al., 2024) for being of a similar size but with different architecture, `DS-Qwen-14B` (DeepSeek-AI, 2025) as a larger model, and `Qwen3-8B` (Team, 2025), as the non-DeepSeek-distilled counterpart to `DS-Qwen3-8B`. Table 5 gives further details.

Attribution methods. We consider three categories of attribution methods: our AttriCoT method, Thought Anchors, and prompting baselines.

AttriCoT. We evaluate several variants of AttriCoT, the LOO variant described in Section 4 that uses $N = S + T$ interventions, and "2x" variants that use $N = 2(S + T)$ interventions. For the latter AttriCoT-2x variants, $S + T$ of their interventions are the same LOO ones as AttriCoT-LOO, while the other $S + T$ leave at least two units out: two units are selected first, then additional units may be added, each independently with Bernoulli probability π (full details in Appendix C.3.1). We regard AttriCoT-LOO as our basic variant and AttriCoT-2x as enhanced variants that use twice the computational cost. The motivation for AttriCoT-2x is that since AttriCoT works by fitting a linear

model, it could potentially take advantage of additional interventions (i.e., observations) if available. Evaluating AttriCoT-2x quantifies this advantage.

Thought Anchors. We compare AttriCoT with the sentence-to-sentence masking method of Thought Anchors (Bogdan et al., 2025) mentioned in Section 2 since it is the closest existing method to ours. Since this method uses KL divergence to quantify causal effects, we refer to it as TA-KL. We evaluate two versions, one that applies a log transform to the KL divergence (as specified in their paper) and one that does not (as implemented in their code). Additional discussion of how we adapted and improved TA-KL can be found in Appendix C.3.2. The computational complexity of TA-KL is similar to that of AttriCoT-LOO, requiring $S + T$ forward passes. However, unlike AttriCoT, TA-KL is not able to exploit additional forward passes if available.

Prompting. We also prompted LLMs to perform CoT attribution. Specifically, since our method of evaluating attributions (described below) depends only on how an attribution method ranks units, we prompted the LLM to rank units in order of importance for generating each output unit, as opposed to the harder task of assigning numerical scores. To make the task easier still, we followed Monteiro Paes et al. (2025) in numbering units with tags and asking only for a ranked list of tags. Please see Appendix C.3.3 for the prompt that we used and other details. Since we prompted the LLM separately for each output unit, the complexity is $O(T)$, but these are generative calls which are more expensive than the non-generative forward passes of AttriCoT and TA-KL.

We experimented with prompting different LLMs: 1) The same model that generated the CoT (i.e., self-attribution); 2) If available, a "counterpart" model with the same architecture that is instruct-tuned and not reasoning-distilled, specifically Qwen2.5-14B-Instruct for DS-Qwen-14B and Qwen3-8B for DS-Qwen3-8B; 3) GPT-OSS-120B, a much larger and more powerful open-weights LLM. We also experimented with providing an in-context learning (ICL) example or not.

Evaluation method. We evaluate CoT attributions in terms of how behaviorally faithful they are to the model that generated the CoT, specifically by computing perturbation curves. Given an attribution method's ranking of prior units for an output unit, the procedure is to perturb (i.e., remove) the top k units, compute the resulting decrease in log probability of the output unit, and plot the curve of these log probability decreases as a function of k. We adopt this evaluation method because it is standard in the explainable AI/NLP literature (Chen et al., 2020; Ju et al., 2023; Monteiro Paes et al., 2025), and because it aligns with our focus on local causal effects on given output units. Appendix D.2.2 presents an alternative evaluation that measures effects on re-generated units instead of the given units, while Appendix D.2.1 considers a masking perturbation instead of removal. We vary k from 0 up to 20% of the number of prior units. For each sample, we average the perturbation curves corresponding to different output units (please see Appendix C.4 for how). In this section, we summarize perturbation curves by computing the area under them (AUPC), where higher values are better. The full perturbation curves are deferred to Appendix D.3.

5.2 Results

We first discuss results on the GSM8K dataset, which we used as an exploratory dataset to evaluate more variations of the methods. We then carried forward the stronger variations to the other datasets. These variations include AttriCoT-2x variants with different Bernoulli sampling probabilities π, both versions of TA-KL (with and without log transform), and prompting the three types of LLMs (self, non-reasoning-distilled counterpart if available, GPT-OSS-120B), with or without an ICL example.

Table 1 shows the AUPC values achieved by all methods for the four CoT models that we used. Each table entry shows the mean AUPC and standard error of the mean over the dataset samples.

- **AttriCoT-LOO** achieves the highest AUPC values among methods that use no more than $S + T$ forward passes or generations. (This remains the case even under the alternative re-generative evaluation in Appendix D.2.2.)
- **AttriCoT-2x**: Base AttriCoT-2x offers a small improvement over AttriCoT-LOO. AttriCoT-2x + Ber(0.3), which perturbs additional units relative to base AttriCoT-2x with probability $\pi = 0.3$, further improves AUPC, quite significantly for DS-Qwen3-8B and Qwen3-8B (please see Appendix D.1 for the full sweep of π). Thus, we choose AttriCoT-2x + Ber(0.3) as the one AttriCoT-2x variant to evaluate on the other datasets.
- **TA-KL**: Between the two variants, no log transform is consistently better, so we use it in the remaining experiments.

Table 1: AUPC (↑) of all attribution methods on the GSM8K dataset. **Bold**: best among methods using $\leq S + T$ forward passes/generative calls. Underline: best overall. Bold/underline also when not significantly worse than best (paired t-test, $p > 0.05$, $n = 1319$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**123.80** ± 1.15	**110.44** ± 0.92	**138.64** ± 1.05	**166.59** ± 1.53
AttriCoT-2x	124.74 ± 1.15	111.35 ± 0.93	139.73 ± 1.05	168.03 ± 1.54
+ Ber(0.3)	126.65 ± 1.24	111.43 ± 0.96	151.22 ± 1.22	184.55 ± 1.74
TA-KL	112.90 ± 1.10	99.82 ± 0.88	116.05 ± 1.12	98.53 ± 1.03
TA-KL (no log)	120.30 ± 1.15	107.91 ± 0.94	123.42 ± 1.20	161.28 ± 1.60
Prompted (self)	43.85 ± 0.73	46.30 ± 0.76	61.39 ± 0.67	67.45 ± 0.98
Prompted-ICL (self)	52.99 ± 0.82	58.28 ± 0.84	60.38 ± 0.65	50.49 ± 0.71
Prompted (counterpart)	—	72.52 ± 0.81	60.20 ± 0.71	—
Prompted-ICL (counterpart)	—	77.34 ± 0.82	58.39 ± 0.67	—
Prompted (GPT-OSS)	54.91 ± 0.76	47.37 ± 0.71	49.65 ± 0.60	54.78 ± 0.77
Prompted-ICL (GPT-OSS)	77.13 ± 0.90	72.59 ± 0.84	62.06 ± 0.73	72.06 ± 0.92

Table 2: AUPC (↑) of top-performing attribution methods on other datasets. **Bold** and underline have the same meanings as in Table 1.

	Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
MATH500	AttriCoT-LOO	**130.44** ± 2.18	**116.44** ± 1.89	**98.38** ± 1.87	**147.56** ± 3.45
	AttriCoT-2x + Ber(0.3)	141.61 ± 2.42	128.69 ± 2.15	102.09 ± 2.17	161.60 ± 3.91
	TA-KL (no log)	120.66 ± 2.23	106.72 ± 1.99	84.16 ± 2.26	134.42 ± 3.91
	Prompted (best same-size)	85.40 ± 1.84	59.03 ± 1.14	44.86 ± 1.29	74.96 ± 2.46
	Prompted-ICL (GPT-OSS)	89.45 ± 1.62	75.16 ± 1.31	52.47 ± 1.38	83.09 ± 2.38
MMLU-Pro	AttriCoT-LOO	**139.12** ± 1.34	**133.47** ± 1.19	**131.87** ± 1.75	**139.61** ± 1.82
	AttriCoT-2x + Ber(0.3)	146.89 ± 1.44	140.70 ± 1.27	141.55 ± 2.03	149.38 ± 1.98
	TA-KL (no log)	129.38 ± 1.36	121.62 ± 1.28	118.84 ± 1.86	127.59 ± 2.02
	Prompted (best same-size)	76.37 ± 0.97	70.36 ± 0.70	61.52 ± 0.82	58.02 ± 0.97
	Prompted-ICL (GPT-OSS)	78.73 ± 0.89	79.08 ± 0.81	56.00 ± 0.72	67.03 ± 1.06
GPQA	AttriCoT-LOO	**98.29** ± 2.73	**107.36** ± 2.39	**65.21** ± 2.41	**84.86** ± 3.22
	AttriCoT-2x + Ber(0.3)	105.75 ± 2.94	117.56 ± 2.77	65.90 ± 2.83	90.69 ± 3.60
	TA-KL (no log)	87.26 ± 2.78	93.84 ± 2.47	54.26 ± 2.59	70.34 ± 3.67
	Prompted (best same-size)	64.03 ± 1.98	57.65 ± 1.63	34.52 ± 1.40	38.89 ± 1.99
	Prompted-ICL (GPT-OSS)	57.38 ± 1.68	61.15 ± 1.56	43.35 ± 1.36	38.38 ± 1.84
ZebraLogic	AttriCoT-LOO	**155.29** ± 2.44	**186.47** ± 2.26	**174.17** ± 2.04	**291.73** ± 3.06
	AttriCoT-2x + Ber(0.3)	166.74 ± 2.71	202.99 ± 2.59	188.81 ± 2.31	329.96 ± 3.86
	TA-KL (no log)	**155.59** ± 2.45	**186.20** ± 2.29	**173.69** ± 1.98	**292.42** ± 3.17
	Prompted (best same-size)	107.13 ± 1.84	83.88 ± 0.94	102.90 ± 1.11	156.24 ± 2.65
	Prompted-ICL (GPT-OSS)	69.77 ± 1.02	83.09 ± 0.96	85.55 ± 0.73	135.95 ± 1.73

- **Prompting**: Among the methods that prompt a same-architecture model (the "self" and "counterpart" rows in Table 1), the best-performing variant depends on the CoT model. For DS-Llama-8B and DS-Qwen-14B, providing an ICL example is better, whereas for DS-Qwen3-8B and Qwen3-8B, ICL does not help. The non-distilled counterpart is significantly better for DS-Qwen-14B, while for DS-Qwen3-8B, it is narrowly beaten by the CoT model itself. (At the time of writing, we were not successful in prompting the counterpart of DS-Llama-8B, namely Llama-3.1-8B-Instruct, for this task.) For GPT-OSS-120B, using ICL is consistently better.

Table 2 compares the AUPC of the strongest variants in each category from Table 1 on the remaining datasets. "Prompted (best same-size)" refers to the best same-architecture model as discussed above. Similar to Table 1, AttriCoT-LOO is always the best or not significantly worse than best among $\leq (S+T)$-complexity methods. TA-KL is AttriCoT-LOO's closest competitor, with ZebraLogic being its strongest dataset where it is statistically tied with AttriCoT-LOO. Notwithstanding, AttriCoT-2x + Ber(0.3) yields significant improvements over AttriCoT-LOO and TA-KL in almost all cases.

Overall, these results show that AttriCoT is consistently the attribution method that is most behaviorally faithful to the CoT model, across 4 CoT models and 5 datasets covering multiple domains.

6 Analysis of Attribution Scores

In this section, we begin an analysis of the attribution score matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ produced by AttriCoT. We start with a single sample, and then propose metrics that summarize the score matrices (Section 6.1) for aggregate analysis. We present results for one metric in Section 6.2.

Figure 2 presents attribution scores on a sample GSM8K trace for two output units, the last reasoning step C3 and the final answer A5. The summation step C3 draws on the preceding computation (C2, 1.31) and on the question that asks for the total (In2, 0.37). The final answer A5 depends on none of the CoT units (≤ 0.02) and only weakly on the answer's own intermediate step (A3, 0.24).

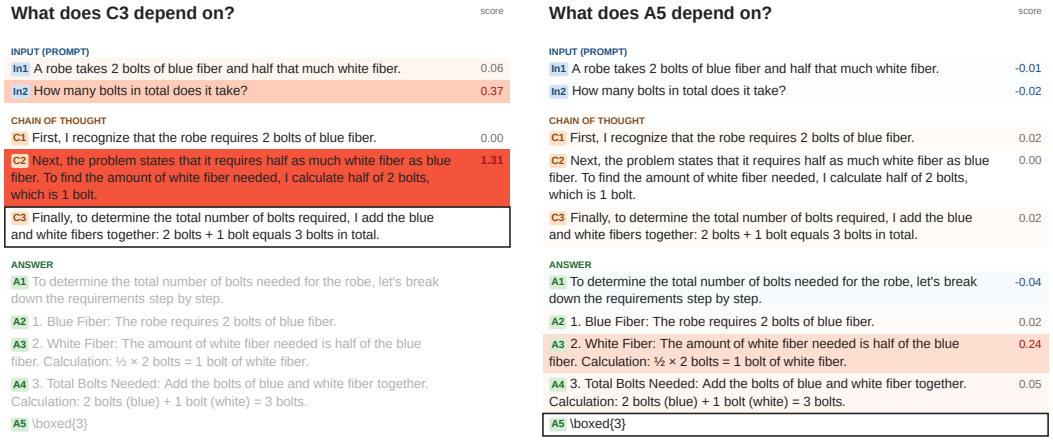


Figure 2: AttriCoT-LOO attributions for one GSM8K trace (`DS-Qwen-14B`), split into prompt (In1–In2), CoT (C1–C3), and answer (A1–A5) units. Both panels fix a target output unit t (black outline) and print the coefficients (scores) $\alpha_{s,t}$ or $\beta_{\tau,t}$ in (4). Red-shaded units lower the target's log prob when removed whereas blue-shaded units raise the log prob (units after target grayed out).

In this instance, attributions concentrate on nearby units. The summation step C3 depends on C2 but not on C1. The boxed answer A5 depends only weakly on the answer's own steps A3–A4, where the model recomputes the result. Note that the input is the exception to this locality. The question In2 matters more to C3 than the nearer C1 because the model consults the prompt at the step where it reaches its answer.

6.1 Metrics

To analyze AttriCoT's attribution score matrices in aggregate, we propose the following metrics (please see Appendix C.5 for the equations). These quantify some of the properties (locality, input dependence, influence on future units) observed in the example in Figure 2.

Score density: The mean of absolute attribution scores over input attribution matrix $\boldsymbol{A}$ and the upper-triangular part of output attribution matrix $\boldsymbol{B}$.

Average influence: For each CoT unit, corresponding to a row in $\boldsymbol{B}$, the mean absolute score over the part of the row to the right of the main diagonal. This represents the CoT unit's average influence on subsequent output units. We compute both the unweighted average over subsequent units as well as exponentially-weighted averages.

Importance by distance: For a given distance d in steps between CoT units, corresponding to the dth diagonal above the main diagonal in $\boldsymbol{B}$, the mean absolute score over the diagonal. This measures the decay in importance of one CoT step to a later step as their separation increases. We compute two versions, one as a function of absolute distance d, the other a function of normalized distance $d/(T_{\text{CoT}} - 1)$, where T_{CoT} is the number of CoT steps (excluding final answer).

Input ratio: For each output unit, corresponding to a column in each of $\boldsymbol{A}$ and $\boldsymbol{B}$, the average of absolute scores in the column of $\boldsymbol{A}$, divided by the average over the columns of both $\boldsymbol{A}$ and $\boldsymbol{B}$. This

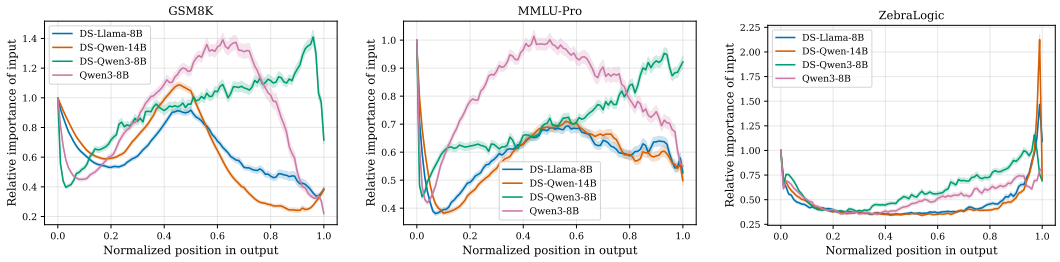
measures the relative importance of input units to the output unit of interest compared to previous output units.

Entropy: For each output unit, we first concatenate the corresponding columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ and normalize so that the absolute sum over the concatenated column is 1. Then we compute the entropy of this normalized column, and normalize the entropy by the entropy of the corresponding uniform distribution. This measures how concentrated are the attributions of the output unit, where significant dependence on only a few prior units would result in low entropy, and diffuse dependence would result in high entropy.

In summary, average influence aggregates over matrix rows, importance by distance aggregates over diagonals, input ratio and entropy over columns, and density over the entire matrix.

6.2 Results

We present results in this section for the input ratio metric applied to AttriCoT-LOO's attributions, on the GSM8K, MMLU-Pro, and ZebraLogic datasets. The full set of results is in Appendix D.5.



(a) GSM8K (b) MMLU-Pro (c) ZebraLogic

Figure 3: Relative importance of input units as a function of normalized position in output.

Figure 3 plots the input ratio metric for output units as a function of their normalized position. The curves represent the mean over samples in the dataset while the shading indicates one standard error above and below the mean. The input ratio is 1 by definition at normalized position $x = 0$ (the input-output boundary). It then drops as expected, as the models digest the problem statement in the first output units before exploring and starting their reasoning. What is notable is the subsequent growth in the input ratio (for example around $x \in [0.2, 0.4]$ for DS-Llama-8B and DS-Qwen-14B on GSM8K). The ratio peaks in the mid to late reasoning stages ($x \in [0.5, 0.9]$) on GSM8K and MMLU-Pro, sometimes above a value of 1. This pattern is suggestive of the model referring back to the input as it reaches one or more answers (as DS-Qwen-14B did in the left panel of Figure 2), for example to verify answers or put them in the context of the input. The peak appears later for more complex problems (i.e., ZebraLogic - Figure 3c, and also MATH500 - Figure 17b, GPQA - Figure 17d in the appendix). This observation aligns with LRMs spending more computation to shape answers on complex problems, which delays verification.

7 Conclusion

Treating a CoT trace as a causal object provides a useful frame for analysis of a model's thought process. Our proposed method, AttriCoT, has pursued this view by decomposing a CoT trace into user-defined units and then formulating and fitting structural equations over the units. The resulting attributions take the form of pairwise importance scores that describe how each unit's log probability depends on prior units. Evaluation on five datasets and four reasoning models shows that AttriCoT produces attributions that provide a consistently more faithful local explanation of model behavior than alternative methods. Metrics derived from AttriCoT's score matrices show suggestive patterns such as models possibly referring back to their inputs in mid-to-late reasoning stages. As next steps, we aim to further develop computationally cheap white-box methods and investigate how the structure of reasoning depends on features such as answer correctness, problem difficulty, and type of reasoning step.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods (mainly AttriCoT, baselines, evaluation, and the analysis metrics in Section 6.1) in software code, and for proposing initial candidates for the above metrics that were then refined by the authors. We have not used generative AI tools for other tasks with required disclosure, such as developing the conceptual framework in Sections 3 and 4, interpreting results, or processing data (code generated by AI was used to process data). The rest of the required disclosure tasks (for example generating synthetic data, formulating and proving mathematical claims, qualitative data analysis) are not applicable to this work. Additionally, we used generative AI tools to run experiments, create and modify scientific figures and tables, make narrow edits to the paper (e.g. updating claims, renaming), summarize and contextualize existing literature, and find references. We have reviewed all AI-assisted work. For example, AI-generated code that implements methods was reviewed by the lead author, paper summaries were cross-checked by our own reading, and all additions to the paper were scrutinized. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix C supplements Section 5.1 in describing details of our experimental setup to aid reproducibility. The appendix covers the selection of datasets and models, generation and segmentation of CoTs, implementation details for the attribution methods and perturbation curve evaluation, and equations for the attribution score metrics. We plan to open-source our code upon acceptance of the paper. However, it is difficult to make the code available for review at this point because it is built upon and intertwined with an existing open-source repository, to the extent that even if the code itself is anonymized, providing it as-is may identify the existing repository and thus compromise our anonymity.

LIMITATIONS

AttriCoT requires the log probabilities of given token sequences, which inference engines for open-weight models expose (e.g., prompt log probabilities in vLLM) but some model APIs do not. Because it scores the given trace rather than regenerating it, its scores are direct effects with intervening units held fixed (not total effects). Additionally, our experiments use open-weight models of 8B–14B parameters, and larger reasoning models remain to be studied.

REFERENCES

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. In *Workshop on Reasoning and Planning for Large Language Models*, 2025. URL https://openreview.net/forum?id=L8094Whth0.

Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, Thomas Lukasiewicz, Jakob Grue Simonsen, and Isabelle Augenstein. Faithfulness tests for natural language explanations. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 283–294, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.25. URL https://aclanthology.org/2023.acl-short.25/.

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. *Preprint, alphaXiv*, pp. v1, 2025.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which LLM reasoning steps matter? In *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=VnSlfeRCaU.

Lang Cao, Yingtian Zou, Chao Peng, Renhong Chen, Wu Ning, and Yitong Li. Step guided reasoning: Improving mathematical reasoning using guidance generation and step reasoning.

In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21101–21118, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1068. URL https://aclanthology.org/2025.emnlp-main.1068/.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5578–5593, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.494. URL https://aclanthology.org/2020.acl-main.494/.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Mądry. ContextCite: Attributing model generation to context. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95764–95807. Curran Associates, Inc., 2024. doi: 10.52202/079017-3035. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/adbea136219b64db96a9941e4249a857-Paper-Conference.pdf.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

James Enouen, Hootan Nakhost, Sayna Ebrahimi, Sercan Arik, Yan Liu, and Tomas Pfister. TextGenSHAP: Scalable post-hoc explanations in text generation with long documents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13984–14011, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.832. URL https://aclanthology.org/2024.findings-acl.832/.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

Yiming Ju, Yuanzhe Zhang, Kang Liu, and Jun Zhao. A hierarchical explanation generation method based on feature interaction detection. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 12600–12611, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.798. URL https://aclanthology.org/2023.findings-acl.798/.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Jinu Lee and Julia Hockenmaier. Evaluating step-by-step reasoning traces: A survey. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 1789–1814, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.94. URL https://aclanthology.org/2025.findings-emnlp.94/.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. ZebraLogic: On the scaling limits of LLMs for logical reasoning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sTAJ9QyA6l.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Uzay Macar, Paul C. Bogdan, Senthooran Rajamanoharan, and Neel Nanda. Thought branches: Interpreting LLM reasoning requires resampling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=bVsAuIOvJ5.

Vivek Miglani, Aobo Yang, Aram Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. Using Captum to explain generative language models. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pp. 165–173, Singapore, December 2023. doi: 10.18653/v1/2023.nlposs-1.19. URL https://aclanthology.org/2023.nlposs-1.19.

Gouki Minegishi, Hiroki Furuta, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. Topology of reasoning: Understanding large reasoning models through reasoning graph properties. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=o1g8NWkxqf.

Lucas Monteiro Paes, Dennis Wei, Hyo Jin Do, Hendrik Strobelt, Ronny Luss, Amit Dhurandhar, Manish Nagireddy, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Werner Geyer, and Soumya Ghosh. Multi-level explanations for generative language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32291–32317, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1553. URL https://aclanthology.org/2025.acl-long.1553/.

nostalgebraist. Interpreting GPT: the logit lens. LessWrong, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Letitia Parcalabescu and Anette Frank. On measuring faithfulness or self-consistency of natural language explanations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6048–6089, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.329. URL https://aclanthology.org/2024.acl-long.329/.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 15012–15032, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.882. URL https://aclanthology.org/2024.findings-emnlp.882/.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, Oskar van der Wal, Malvina Nissim, and Arianna Bisazza. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 421–435, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-demo.40. URL https://aclanthology.org/2023.acl-demo.40.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=bzs4uPLXvi.

Martin Tutek, Fateme Hashemi Chaleshtori, Ana Marasovic, and Yonatan Belinkov. Measuring chain of thought faithfulness by unlearning reasoning steps. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9935–9960, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.504. URL https://aclanthology.org/2025.emnlp-main.504/.

Sydney Von Arx and Amy Deng. CoT may be highly informative despite "unfaithfulness". https://metr.org/blog/2025-08-08-cot-may-be-highly-informative-despite-unfaithfulness/, 8 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=y10DM6R2r3.

Dennis Wei, Sanjeeb Dash, Tian Gao, and Oktay Gunluk. Generalized linear rule models. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6687–6696. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wei19a.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecxO1h.

Donald Ye, Max Loffgren, Om Kotadia, Linus Wong, and Jonas Rohweder. Mechanistic evidence for faithfulness decay in chain-of-thought reasoning. In *Workshop on Latent & Implicit Thinking – Going Beyond CoT Reasoning*, 2026. URL https://openreview.net/forum?id=wVj7dB7waI.

Kerem Zaman and Shashank Srivastava. Is chain-of-thought really not explainability? Chain-of-thought can be faithful without hint verbalization, 2025. URL https://arxiv.org/abs/2512.23032.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=O6I0Av7683.

Zheng Zhao, Yeskendir Koishekenov, Xianjun Yang, Naila Murray, and Nicola Cancedda. Verifying chain-of-thought reasoning via its computational graph. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CxiNICq0Rr.

A Related Work

This section highlights the existing methods in the literature, what they aim to do, and how they work. Table 3 presents an overview of existing CoT attribution methods Zaman & Srivastava (2025).

Method	Ref.	Type	Intervention	Causal	Granularity	Access
Biasing Features	(Turpin et al., 2023)	Input perturbation	✓		Full CoT	Black-box
Counterfactual Edits	(Atanasova et al., 2023)	CoT perturbation	✓		Step/Token	Black-box
FUR (*)	(Tutek et al., 2025)	Unlearning	✓		Step	White-box
CRV	(Zhao et al., 2026)	Circuit analysis	✓	✓	Step	White-box
NLDD	(Ye et al., 2026)	Counterfactual	✓	✓	Step	Gray-box
-	(Zaman & Srivastava, 2025)	Causal intervention	✓	✓	Step	Black-box
FRODO	(Paul et al., 2024)	Causal Intervention	✓	✓	Step	Black-box
Thought Branches	(Macar et al., 2026)	Causal Intervention	✓	✓	Step	Black-box
Thought Anchors	(Bogdan et al., 2025)	Causal Intervention	✓	✓	Step	White-box / Gray-box
CC-SHAP	(Parcalabescu & Frank, 2024)	Attribution			Token	Gray-box
-	(Minegishi et al., 2025)	Pattern analysis			Step	White-box
-	(Parcalabescu & Frank, 2024)	Benchmark			Full CoT	Black-box
AttriCoT (Ours)	-	Causal intervention	✓	✓	Flexible	Gray-box

Table 3: Overview of CoT attribution and faithfulness methods. **Intervention:** Whether the method perturbs the CoT or the input. **Causal:** Whether the method suggest a causal framework (e.g. activation patching). **Granularity:** The degree at which the CoT is treated (Full CoT considers input/output sets, step/tokens intervene inside the CoT). **Access:** Whether the method requires the model's hidden-states or activations (White-box), only output token probabilities (Gray-box), or just the text the model generates (Black-box).

Faithfulness of LRMs. Faithfulness has two different meanings in the literature in the context of LRMs. First, *"representational"* faithfulness refers to how much the verbalized chain of thought is representative of the model's true hidden computations Barez et al. (2025). Second, faithfulness also refers to how much verbalized steps are contributing to the final answer of the model Zaman & Srivastava (2025). Our work is agnostic of the *"representational"* faithfulness, and operates on a projection of the internal state of the model. It is aligned with the second notion because it exactly addresses the contributions of CoT steps to later steps and the final answer. These dependencies are a meaningful measure of LLM, even when the CoT text does not verbalize internal reasoning Zaman & Srivastava (2025). Rather than measuring how well the generated text aligns with the model internal representation of the reasoning process, AttriCoT identifies which reasoning steps cause subsequent steps in the reasoning trace. In other words, our work aims to reveal if the generated CoT has a coherent causal structure.

Perturbation-based, non-causal methods. The earliest work on CoT attribution are a set of methods that intervenes on the input of the CoT to measure the change in the model's output. First, Turpin et al. (2023) suggests a set of techniques to inject hints in the input and observe how the output is affected. Similarly, Atanasova et al. (2023) inserts tokens in the input and manipulates the CoT to then inspect the changes that it implies. Further, CC-SHAP Parcalabescu & Frank (2024) applies the Shapley-values to the attribution of the CoT at a token level. However, they measure a correlation with the final output, rather than evaluating causality.

White-box and gray-box causal methods. Contrastingly, a set of existing methods evaluates the causality of reasoning traces using the model's internals. CRV Zhao et al. (2026) represent the CoT using hidden-state representation of the model obtained with transcoders. They detect reasoning errors, and uses activation patching to uncover causality of the CoT. NLDD Ye et al. (2026) corrupts individual reasoning steps and measure how the confidence of the model though its output token probabilities is affected.

Black-box step-level causal methods. Recent work suggested black-box methods, evaluating the causality of a CoT at a step-level. These methods perturb individual reasoning steps and evaluate the impact of such edits on the output or on the final answer. FRODO Paul et al. (2024) is one of the first paper introducing the Causal Mediation Analysis (CMA). They compare direct and indirect effects of step perturbations on the model's output. Thoughts Branches Macar et al. (2026) also introduce

a causal framework to attribute the CoT. The authors re-generate the CoT from different steps in the CoT, and measures the final answer distribution.

The closest work to ours is Thought Anchors, specifically their sentence-to-sentence masking method (Bogdan et al., 2025, Sec. 5, App. M). This method also attributes every unit in the CoT to prior ones, producing a matrix of attribution scores (their units can only be sentences however). It does so by masking each sentence, either by suppressing all attention to it (white-box) or removing the sentence from the input (gray-box, measuring the effect on subsequent token logits), and then performing a forward pass to determine the effects. The difference however is that Thought Anchors quantifies effects on a downstream sentence using KL divergences between token distributions. This is not local in the same way as AttriCoT because it does not measure how the given generated sentence was affected. Instead, it measures the change in token distributions at the position of the sentence. Another difference is that Thought Anchors is limited to leave-one-sentence-out interventions, whereas AttriCoT fits a structural causal model and can thus exploit information from additional interventions if available. The authors of Bogdan et al. (2025) also propose a resampling method, which is more distant from AttriCoT in that it does not yield unit-to-unit attributions, is even less local due to resampling CoTs, and is computationally expensive because of resampling.

B Exploratory White-Box Method Based on Attention Intervention

We explored the possibility of further reducing the computational cost of AttriCoT (dominated by the $O(N)$ forward passes through the model) in the case of white-box access to the model. When AttriCoT removes a unit, it removes the unit's tokens from the token sequence that is input to the model and performs a full forward pass to compute log probabilities. For the white-box approach, we instead considered intervening on the attention weights of the unit's tokens in a single Transformer block of the model. The outputs of the attention heads in the block are then recomputed and propagated to the model's logits through a "shallow network." This approach is significantly less expensive because it requires only one full forward pass to obtain the original attention weights and other quantities needed to perform the interventions. Subsequently, N shallow passes suffice to compute perturbed attention outputs and propagate them through the shallow network to the logits.

The following subsections provide more details on this attention intervention method. In Section B.1, we discuss how we intervene by setting attention weights to zero and show how the attention output is recomputed. Section B.2 describes how the shallow network consists essentially of an MLP layer and the language modeling head of the LLM, with choices of which Transformer block to intervene on and how to construct the shallow network. We experimented with two variants, an "untrained" one in which the shallow network is constructed from modules of the LLM, and a "trained" one where the shallow network is then fine-tuned.

B.1 Attention intervention and attention output recomputation

The generation of a token (which we will refer to as the target token) depends on previous tokens through the attention mechanism. The output of each attention head is given by

$$\mathbf{a} = \sigma\left(\frac{\mathbf{q}\mathbf{K}^T}{\sqrt{d_K}}\right)\mathbf{V} = \sigma(\mathbf{s})\mathbf{V}, \tag{8}$$

where $\mathbf{q} \in \mathbb{R}^{1\times d_K}$ is the query vector corresponding to the target token, $\mathbf{K} \in \mathbb{R}^{n\times d_K}$ is the matrix of key vectors for the n previous tokens, $\mathbf{V} \in \mathbb{R}^{n\times d_K}$ is the corresponding matrix of value vectors, $\mathbf{s} \in \mathbb{R}^{1\times n}$ is the vector of $\mathbf{q}\mathbf{k}^T$ inner products ("attention scores"), and σ denotes the softmax operation.

We consider for illustration the case where the target token is the first token of the tth output unit $\boldsymbol{y}_t$. In this case, we segment the previous token sequence into units $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ as before. The vector $\mathbf{s}$ and matrix $\mathbf{V}$ may be partitioned accordingly so that (8) can be rewritten as

$$\mathbf{a} = \sigma\left([\mathbf{s}_{\boldsymbol{x}_1} \quad \cdots \quad \mathbf{s}_{\boldsymbol{x}_S} \quad \mathbf{s}_{\boldsymbol{y}_1} \quad \cdots \quad \mathbf{s}_{\boldsymbol{y}_{t-1}}]\right) \begin{bmatrix} \mathbf{V}_{\boldsymbol{x}_1} \\ \vdots \\ \mathbf{V}_{\boldsymbol{y}_{t-1}} \end{bmatrix} = \sum_{s=1}^{S} \mathbf{p}_{\boldsymbol{x}_s}\mathbf{V}_{\boldsymbol{x}_s} + \sum_{\tau=1}^{t-1} \mathbf{p}_{\boldsymbol{y}_\tau}\mathbf{V}_{\boldsymbol{y}_\tau}, \tag{9}$$

where we have defined *attention weights* $\mathbf{p} = \sigma(\mathbf{s})$.

We now approximate the effect of removing a unit, say the first CoT unit $\boldsymbol{y}_1$, by removing the corresponding subvector $\mathbf{s}_{\boldsymbol{y}_1}$ and submatrix $\mathbf{V}_{\boldsymbol{y}_1}$ from (9). This corresponds to setting the corresponding attention weights $\mathbf{p}_{\boldsymbol{y}_1}$ to zero. The effect on the remaining softmax outputs $\mathbf{p}_{\boldsymbol{x}_1}, \ldots, \mathbf{p}_{\boldsymbol{x}_S}, \mathbf{p}_{\boldsymbol{y}_2}, \ldots, \mathbf{p}_{\boldsymbol{y}_{t-1}}$ is just a renormalization, so that the modified attention output becomes

$$\mathbf{a}' = \sum_{s=1}^{S} \frac{\mathbf{p}_{\boldsymbol{x}_s}}{1 - \mathbf{1}^T \mathbf{p}_{\boldsymbol{y}_1}} \mathbf{V}_{\boldsymbol{x}_s} + \sum_{\tau=2}^{t-1} \frac{\mathbf{p}_{\boldsymbol{y}_\tau}}{1 - \mathbf{1}^T \mathbf{p}_{\boldsymbol{y}_1}} \mathbf{V}_{\boldsymbol{y}_\tau}, \tag{10}$$

where $\mathbf{1}$ is a vector of ones. (This is an approximation because the removal of $\boldsymbol{y}_1$ also modifies the key and value vectors of subsequent thoughts $\boldsymbol{y}_2, \ldots, \boldsymbol{y}_{t-1}$, but we neglect these second-order effects.) The above intervention is repeated on all attention heads within a Transformer block of the model, resulting in a perturbed output like (10) from each attention head.

B.2 Shallow network for propagation to logits

Our proposal is to use perturbed attention outputs as in (10) as the starting point to estimate the downstream effect on the log probabilities of the tokens of a target unit $\boldsymbol{y}_t$. These log probabilities are determined by the corresponding logits, so the task is to propagate perturbed attention head outputs (10) to perturbed logits. We consider doing so using a "shallow network."

We first discuss an "untrained" version in which the shallow network consists of modules in the LLM along the path that connects attention outputs to logits, with the same (frozen) parameter values. In the case where the attention intervention is applied to the model's last Transformer block, then the shallow network is the complete set of modules between attention outputs and logits, namely: the output projection ("o-projection") that combines attention head outputs; addition of the o-projected attention output to the residual stream; the subsequent MLP layer, with residual connection and layer normalization; the LLM's language modeling head, also preceded by a layer normalization. If the Transformer block is not the last one, then we still take the o-projection and MLP layer from the same Transformer block, but then skip over all subsequent Transformer blocks and just attach the language modeling head, similar to the LogitLens technique nostalgebraist (2020) in mechanistic interpretability. The choice of Transformer block where the intervention is applied is a hyperparameter of the method and we experimented with LLM-dependent ranges of block indices (please see Table 14 for the results).

We also tested a "trained" version, in which the shallow network is initialized from the untrained version. We then train the shallow network using AttriCoT's computed log probabilities as supervision. Specifically, for a given sequence of units and a perturbation to some of the units, we can compute the log probabilities of all tokens as AttriCoT does by performing a full forward pass. We also have the shallow network's prediction of these log probabilities, obtained by performing the corresponding intervention on attention weights and propagating through the shallow network. We compare the two sets of log probabilities using mean squared error as the loss function.

We trained shallow networks using a deliberately small set of 10 samples (8 for training, 2 for validation) from the GSM8K dataset. This number was chosen because it represents less than 1% of the full test set, with the idea that the cost of training could be amortized over evaluation on the rest of the dataset. Moreover, each sample actually yields $S + T + 1$ unit sequences, one perturbed sequence for each of its $S + T$ units plus an unperturbed sequence, and each of these sequences can be a long sequence of tokens. Hence the amount of training data is larger than it might appear.

C Experimental Design

C.1 Description of selected datasets and models

Selection of datasets. To evaluate AttriCoT and the baselines, we selected 5 reasoning benchmarks across 3 different domains. Table 4 presents the datasets. For all the datasets, we selected the test split that is available. Due to the computational cost of perturbation curve evaluation (see Appendix C.4), we selected 1,400 samples from the full MMLU-Pro dataset by selecting 100 sam-

Domain	Ref.	Name	Size	Type of evaluation	
				Math-Verify[1]	MCQ Prompting
Maths	Cobbe et al. (2021)	GSM8K	1319	✓	
	Hendrycks et al. (2021)	MATH500	500	✓	
Multi-domain	Rein et al. (2024)	GPQA-Diamond	198		✓
	Wang et al. (2024)	MMLU-Pro	1400		✓
Logic	Lin et al. (2025)	ZebraLogic	564		✓

Table 4: Overview of selected reasoning datasets

ples from each of the 14 categories in MMLU-Pro. These categories correspond to domains such as History, Law, Computer Science, Maths, or Physics.

ZebraLogic. The ZebraLogic dataset is complex by nature, and requires significant reasoning efforts compared to other datasets (i.e. models tend to generate more tokens). At the same time, the dataset is significantly large (around 3.2k samples). Due to computational limitations, we first inferred around 20% of the first samples of the dataset (i.e. 620 samples). On these samples, we observed that a significant portion of them were not completed. By not completed, we mean that some reasoning traces did not include `</think>` under their maximum number of tokens. For each sample, we selected it only if all of our selected models completed their reasoning. Pruning the other samples, we obtained 564 samples.

Next, we analyzed the complexity of the 564 samples obtained on the ZebraLogic dataset. Such analysis would allow us to ensure a fair distribution of the complexity of the questions composing our subset. Figures 4a and 4b present the distribution of the puzzle sizes and the performance of the model with respect to the puzzle dimensions. We define the puzzle size by the product of the number of columns M and rows N of each puzzles ($S = M \times N$). Figure 4a shows that the puzzle size is widely spread across our selected samples, namely from 4 to 36.

Further, Figure 4b shows the performance of models with regards to the puzzle sizes. We observe that the puzzle dimension correlates well with the complexity of the problem. Indeed, the larger the puzzle is, the more token the models generates, and the lower their performance is. To this extend, we arbitrary defined three levels of complexity, namely easy ($S < 10$), medium ($S \in [10, 20]$), and hard ($S > 20$).

Selection of models. Table 5 presents the reasoning models that we selected.

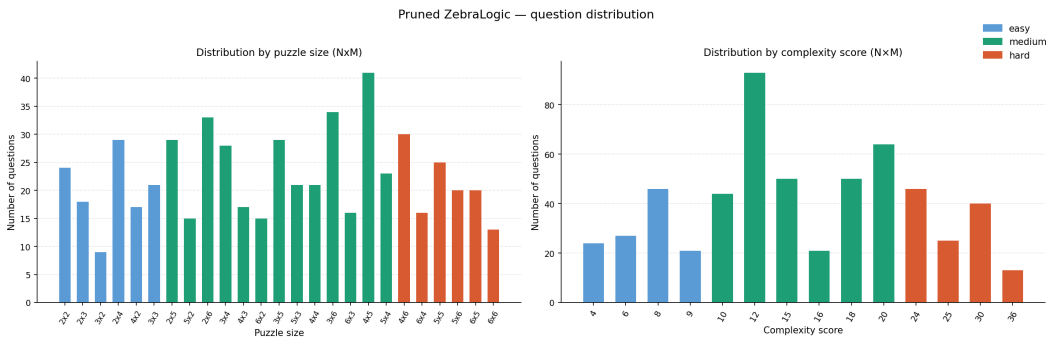
Name	Ref.	Non-reasoning counterpart	Reasoning
`Qwen-3-8B`	Team (2025)	`Qwen/Qwen3-8B`	`Qwen/Qwen3-8B`
`DS-Qwen3-8B`	DeepSeek-AI (2025)	`Qwen/Qwen3-8B`	`deepseek-ai/DeepSeek-R1-0528-Qwen3-8B`
`DS-Llama-8B`	DeepSeek-AI (2025)	`meta-llama/Llama-3.1-8B-Instruct`	`deepseek-ai/DeepSeek-R1-Distill-Llama-8B`
`DS-Qwen-14B`	DeepSeek-AI (2025)	`Qwen/Qwen-2.5-14B`	`deepseek-ai/DeepSeek-R1-Distill-Qwen-14B`

Table 5: Overview of selected reasoning models and their non-reasoning counterparts.

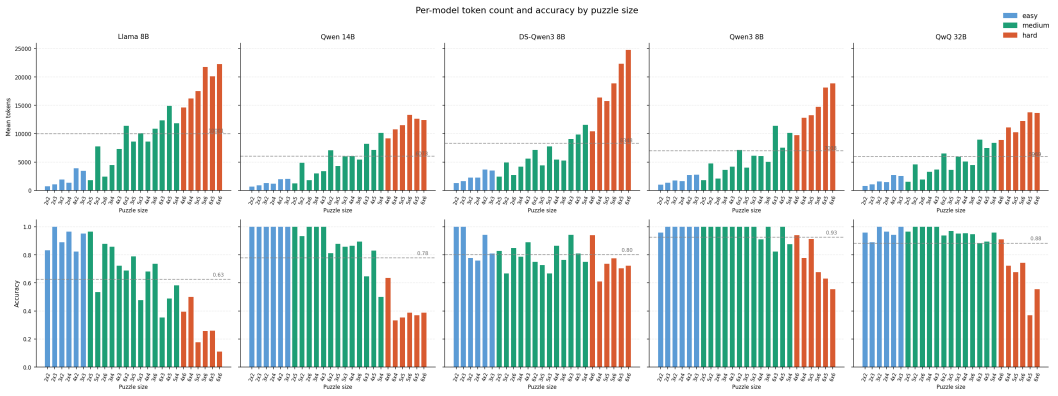
C.2 Generation and segmentation of CoTs

For each sample of each dataset shown in Table 4 (e.g. 1400 samples from MMLU-Pro), we take the prompt as input $\boldsymbol{x}$ and generate an output $\boldsymbol{y}$ from each of the models in Table 5. The prompt may be just a question in the case of the math datasets or also include answer choices in the MCQ datasets. Each output consists of a CoT reasoning trace, terminated by an end-of-thinking `</think>` token, followed by a "final answer," which may also be of significant length.

Segmentation of the prompt. For the math datasets (GSM8K and MATH500), the prompt/question $\boldsymbol{x}$ was split into sentences $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S)$. For the MCQ datasets (MMLU-Pro, GPQA, ZebraLogic), the prompt was split on the newline character `"\n"` into $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S)$. In both cases, each segment $\boldsymbol{x}_s$ may provide a different piece of information or instruct the model on



(a) Distribution of puzzle's size



(b) Model's performance per puzzle's size

Figure 4: Analysis of ZebraLogic subset

how to express its answer. For the MCQ datasets, the segments also separate the answer choices ("A. ...", "B. ...", etc.). Segmenting prompts in this way allows differential attribution to elements within the prompt.

Segmentation of the CoT. We segment the CoT reasoning traces by splitting on a delimiter. The resulting segments form the first part of the output unit sequence $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$. In the literature, multiple delimiters have been proposed (Zhang et al., 2025; Yao et al., 2023; Lee & Hockenmaier, 2025; Cao et al., 2025). In this paper, we segmented the reasoning traces based on the double-newline delimiter `"\n\n"` (Zhang et al., 2025), since reasoning models appear to natively generate these tokens.

After segmentation, we limit the number of CoT steps to 100, again because of the computational constraints of perturbation curve computation (Appendix C.4). We do so by merging consecutive steps together. If segmentation produces $T_{\text{CoT}} > 100$ steps, then each step in the merged sequence consists of either $\lfloor T_{\text{CoT}}/100 \rfloor$ or $\lceil T_{\text{CoT}}/100 \rceil$ old steps, chosen so that the sum over merged steps equals T_{CoT}. For example if $T_{\text{CoT}} = 140$, 60 of the merged steps consist of 1 old step while 40 consist of 2. When there is a choice between $\lfloor T_{\text{CoT}}/100 \rfloor$ or $\lceil T_{\text{CoT}}/100 \rceil$, the one that keeps the cumulative length in characters closer to that of a uniform distribution is chosen.

Segmentation of the final answer. We also segment the final answer by splitting on `"\n\n"`, since models often produce a summary of their reasoning in their final answer and continue to use the `"\n\n"` separator in doing so. In the rare cases where it is needed, we also merge final answer steps into 100 total steps as described above. The sequences of CoT steps and final answer steps together make up the output units $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$.

C.3 Additional details on attribution methods

C.3.1 AttriCoT

Base AttriCoT-2x variant The AttriCoT-2x variants introduced in Section 5.1 use $N = 2(S+T)$ interventions, where the first $S + T$ of these are leave-one-out. Among these variants, we define a base AttriCoT-2x variant for which the remaining $S + T$ interventions are exactly leave-two-out, sampled uniformly from all $\binom{S+T}{2}$ pairs of units. The base AttriCoT-2x variant follows the approach of the C-LIME algorithm in Monteiro Paes et al. (2025), which limits the number of units K perturbed at one time to a small integer $K_{\max}$, and samples an equal number of perturbations of each cardinality from 1 to $K_{\max}$. Base AttriCoT-2x thus corresponds to $K_{\max} = 2$, with its perturbations evenly split between cardinalities 1 and 2.

Bernoulli augmentation In addition to the AttriCoT-LOO and base AttriCoT-2x variants, we consider further increasing the number of units K perturbed at one time through random Bernoulli sampling. In the first case, we start with AttriCoT-LOO and, to each LOO subset of units (a singleton τ), we add each unit not already included ($\neq \tau$), independently with probability π. This increases K by a binomially-distributed amount. For base AttriCoT-2x, we apply this Bernoulli augmentation only to its leave-two-out interventions, so that the number of perturbed units is 2 plus a binomial random variable. Bernoulli augmentation is not applied to the LOO interventions of AttriCoT-2x (see Appendix D.1 for results justifying why).

Effects of interventions on the causal graph One might wonder about the problem of CoT units being dependent on input units when both are included as covariates in a linear regression model. AttriCoT's interventions deal with this dependency by implicitly intervening on all units, not just those that are removed (or otherwise perturbed). This implicit intervention breaks the dependence of CoT units on input units when estimating their effects on a downstream output unit.

We use the following simplified example to explain further, where the input, CoT, and final answer each consist of one unit. The causal graph has (in the general case) the edges (input $\rightarrow$ CoT), (input $\rightarrow$ answer), (CoT $\rightarrow$ answer) (the first edge is the potentially problematic dependency). When attributing the final answer, the latter two edges are the ones of interest. In this case, when the input is perturbed (setting it to a different value), the CoT is also fixed to the value it had in the original sequence. This implicit intervention on the CoT deletes the (input $\rightarrow$ CoT) edge, allowing the linear regression to focus on quantifying the (input $\rightarrow$ answer), (CoT $\rightarrow$ answer) edges that are of interest (i.e., the mutilated graph). More generally when there are more than three units, fixing non-perturbed units to their original values deletes incoming edges to their corresponding nodes in the causal graph and isolates the "star" subgraph centered on the output unit of interest.

Regularization For the experimental results reported in this paper, we did not use regularization. One reason was to avoid ambiguity in ranking units (as required for evaluating perturbation curves) due to having multiple zero-valued coefficients.

C.3.2 Thought Anchors-KL

We re-implemented TA-KL by starting from the authors' code and making several modifications and improvements:

1. **More general units:** We generalized TA-KL to operate on the same more general units used by AttriCoT (prompt, CoT, and final answer units) rather than the original's restriction to CoT sentences only. This enables a fairer comparison using the same unit segmentation for both AttriCoT and TA-KL.
2. **Batch forward passes:** Like AttriCoT-LOO, TA-KL performs leave-one-out perturbations on the unit sequence and then runs forward passes through the LLM. In the case of TA-KL, each forward pass computes token distributions at all token positions, rather than just the log probability of the originally generated token in AttriCoT's case. We replaced TA-KL's per-perturbation calls to the LLM with a single batch of forward passes, including the original sequence and all perturbed sequences. This batching is more efficient and does not change the result.

3. **Token position matching:** TA-KL quantifies the causal effect of a perturbation on a target unit by computing KL divergences between token distributions at every token position in the target unit, and then taking the mean of these KL divergences over the unit. This requires matching token positions in the original sequence with their corresponding token positions in a perturbed sequence. We improved the matching procedure and made it more robust. TA-KL's original approach searched for an "anchor" of consecutive matching tokens following the unit that was removed in the perturbed sequence. This often failed in our experience because of re-tokenization near the removal boundaries (e.g., newline characters merging with adjacent tokens). Our approach first finds the token ranges corresponding to the target unit in both the original and perturbed sequences. It then runs a greedy two-pointer alignment algorithm on the two token ranges.
4. **Log transform:** The Thought Anchors paper (Bogdan et al., 2025) proposed first applying a log transform to KL divergences before taking the mean over a target unit. The authors' code however does not apply a log transform. We evaluated both versions on the GSM8K dataset, as reported in Table 1, to see which is better for our purpose.

C.3.3 PROMPTING

To prompt an LLM to perform CoT attribution, we used the conversation format shown in Figure 5. As mentioned in Section 5.1, we prompt the LLM to attribute output units one at a time, and the example in Figure 5 is for an output unit J. The first user turn is the original prompt (e.g. a math question), divided into input units $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S$ (we use 0-based indexing in the code). The first assistant turn is the LRM's response to the prompt, divided into units $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$. Before each unit $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_S, \boldsymbol{y}_1, \ldots$, we insert a tag of the form `<uN>`, where the unit index increases in a single sequence across input and output units. The response has also been truncated to the target unit J, which we found led to better performance by removing the distraction of later units that cannot affect unit J.

Prompting baseline

[in-context example if used, following the same format below]

< | User | > <u0> *[unit 0]* <u1> *[unit 1]* ... <u($S-1$)> *[unit $S-1$]*
< | Assistant | > <uS> *[unit S]* <u($S+1$)> *[unit $S+1$]* ... <u($S+J$)> *[unit $S+J$]*
< | User | > The above is a conversation that has been divided into units.
Each unit is tagged with an identifier in the format <u0>, <u1>, etc.
For unit <u($S+J$)>, please list the preceding units that were most important for generating it.
List the unit numbers in order from most important to least important, for example [3, 1, 4].
Answer with only the list and do not explain.
< | Assistant | > *[response giving attributions as a ranked list, e.g. [1, 5, 9, 2]]*

Figure 5: Conversation format used by the prompting baseline to attribute output unit J.

The second user turn is the instruction to the LLM, describing the attribution task and answer format. The LLM then responds with attributions as a ranked list of unit indices. Note that the LLM being prompted to perform attribution may be different from the LRM that generated the response in the first assistant turn, as discussed in Section 5.1. In particular, we observed that when a DeepSeek-distilled model is prompted to attribute its own response (self-attribution), it first engages in lengthy reasoning before finally outputting a ranked list. Because of this, we allowed the prompted model to generate up to 1000 new tokens, even though the ranked list is expected to take far fewer tokens. The non-DeepSeek-distilled counterpart models did not have the same issue.

An in-context learning (ICL) example can be prepended to the conversation shown in Figure 5, following the same conversation format. We used AttriCoT's attribution results on the same (dataset, CoT model) pair as the source of ICL examples, specifically a sample with the largest number of output units. This choice of data sample allowed us to match the ICL example to the output unit being attributed, i.e., for attributing output unit J as in Figure 5, we supply an example of AttriCoT also attributing the Jth output unit. Since AttriCoT produces numerical scores while we only prompt LLMs for a ranked list, we converted AttriCoT's scores to a ranked list of units with the top k scores.

The number k was set to 20% of the number of units preceding the output unit (e.g. 20% of $S + J$ in Figure 5), which matches the largest k used in perturbation curve evaluation (see the next section).

C.4 Perturbation curve evaluation

As mentioned in Section 5.1, for each output unit $\boldsymbol{y}_t$, we compute a perturbation curve that measures the decrease in log probability of $\boldsymbol{y}_t$ as a function of the number k of top-ranked prior units that are removed. This decrease in log probability is given by

$$D_t(k) = \frac{1}{|\boldsymbol{y}_t|} \log P\left(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}\right) - \frac{1}{|\boldsymbol{y}_t|} \log P\left(\boldsymbol{y}_t \mid \boldsymbol{x}^{(k)}, \boldsymbol{y}_{<t}^{(k)}\right), \tag{11}$$

where $(\boldsymbol{x}^{(k)}, \boldsymbol{y}_{<t}^{(k)})$ is the sequence of prior units with the top k removed, and we take the mean of token log probabilities as in (6) rather than the sum in (3). Note that the decrease $D_t(k)$ could be negative.

Since we sweep k from 0 to 20% of the total number of prior units, $S + t - 1$, each output unit's perturbation curve has a different number of points. To average perturbation curves corresponding to different output units, we first convert k to a fraction $k/(S + t - 1)$, and then linearly interpolate each perturbation curve onto a common grid of perturbation fractions (spanning 0% to 20%). These interpolated perturbation curves can then be averaged to produce one perturbation curve per data sample, and thus one AUPC value per sample as well.

It is worth noting that evaluating perturbation curves is more computationally expensive than running any of the attribution algorithms. Since a perturbation curve is computed for each output unit $t = 1, \ldots, T$, and the number of perturbations/forward passes is proportional to the number of prior units $S + t - 1$, the total number of forward passes is quadratic: $O(T(S + T))$. It is primarily for this reason that we limited the number of output units to $T = 100$. We believe that this cost is justified in order to thoroughly evaluate CoT attributions, and we emphasize that all of the attribution algorithms have linear cost in $S + T$.

C.5 Metrics for attribution score matrices

Here we provide equations for the metrics defined in Section 6.1 for summarizing attribution score matrices $\boldsymbol{A} \in \mathbb{R}^{S \times T}$ (input $\rightarrow$ output) and $\boldsymbol{B} \in \mathbb{R}^{T \times T}$ (output $\rightarrow$ output, strictly upper triangular). Let T_{CoT} denote the number of CoT output units (excluding final answer units). We also define the concatenated matrix $\boldsymbol{C} = [\boldsymbol{A}; \boldsymbol{B}] \in \mathbb{R}^{(S+T) \times T}$, which stacks $\boldsymbol{A}$ above $\boldsymbol{B}$.

Score density. The mean absolute attribution score over all structurally non-zero entries:

$$\text{density} = \frac{\displaystyle\sum_{s=1}^{S}\sum_{t=1}^{T}|\alpha_{s,t}| \; + \; \sum_{\tau=1}^{T}\sum_{t=\tau+1}^{T}|\beta_{\tau,t}|}{S \cdot T + T(T-1)/2}. \tag{12}$$

Average influence. For each CoT unit $\tau = 1, \ldots, T_{\text{CoT}}$, the mean absolute attribution from all downstream output units:

$$I(\tau) = \frac{1}{T - \tau} \sum_{t=\tau+1}^{T} |\beta_{\tau,t}|. \tag{13}$$

We also define an exponentially-weighted variant with decay rate $r \in [0, 1)$:

$$I_r(\tau) = \frac{1 - r}{1 - r^{T-\tau}} \sum_{d=1}^{T-\tau} r^{d-1} \, |\beta_{\tau,\tau+d}|. \tag{14}$$

The motivation for the exponentially-weighted variant and the choice of r are discussed in Appendix D.5. Both (13) and (14) are plotted as a function of normalized CoT position τ / T_{CoT}.

Importance by distance. For a distance d in CoT steps, the mean absolute score along the d-th super-diagonal of $\boldsymbol{B}$:

$$L(d) = \frac{1}{T_{\text{CoT}} - d} \sum_{\tau=1}^{T_{\text{CoT}}-d} |\beta_{\tau,\tau+d}| \,. \tag{15}$$

We plot this both as a function of absolute distance d and of normalized distance $d/(T_{\text{CoT}} - 1)$.

Input ratio. For each output unit t, the ratio of mean absolute input attribution to mean absolute attribution over all prior units. Let $c_{j,t}$ denote entry (j, t) of $\boldsymbol{C}$; the units prior to output unit t correspond to rows $j = 1, \ldots, S + t - 1$ of $\boldsymbol{C}$:

$$R(t) = \frac{\dfrac{1}{S}\displaystyle\sum_{s=1}^{S}|\alpha_{s,t}|}{\dfrac{1}{S+t-1}\displaystyle\sum_{j=1}^{S+t-1}|c_{j,t}|} \,. \tag{16}$$

Normalized entropy. For each output unit t, we form a distribution from the absolute scores of its predecessors,

$$p_j = \frac{|c_{j,t}|}{\displaystyle\sum_{j'=1}^{S+t-1}|c_{j',t}|}, \quad j = 1, \ldots, S + t - 1, \tag{17}$$

and compute the normalized Shannon entropy:

$$E(t) = \frac{-\displaystyle\sum_{j=1}^{S+t-1} p_j \ln p_j}{\ln(S + t - 1)} \,. \tag{18}$$

This metric lies in $[0, 1]$, where low values indicate concentrated attributions and high values indicate diffuse attributions.

Table 6: AUPC (↑) of AttriCoT-LOO on GSM8K with Bernoulli augmentation at probability π. **Bold**: best per model. $\pi = 0.05, 0.1$ were run only on DS-Llama-8B and DS-Qwen-14B.

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**123.80** $\pm$ 1.15	**110.44** $\pm$ 0.92	**138.64** $\pm$ 1.05	**166.59** $\pm$ 1.53
+ Ber(0.01)	122.50 $\pm$ 1.13	109.18 $\pm$ 0.92	136.64 $\pm$ 1.08	165.16 $\pm$ 1.54
+ Ber(0.05)	119.55 $\pm$ 1.14	105.34 $\pm$ 0.91	—	—
+ Ber(0.1)	116.41 $\pm$ 1.15	102.11 $\pm$ 0.92	—	—

Table 7: AUPC (↑) of AttriCoT-2x on GSM8K with Bernoulli augmentation at probability π applied only to its leave-two-out interventions. **Bold**: best per model. †: $\pi = 0.3$, the single best value whose results are reported in Tables 1, 2. $\pi = 0.5$ was not run on DS-Llama-8B and DS-Qwen-14B.

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
Base AttriCoT-2x	124.74 $\pm$ 1.15	111.35 $\pm$ 0.93	139.73 $\pm$ 1.05	168.03 $\pm$ 1.54
+ Ber(0.01)	124.92 $\pm$ 1.15	111.46 $\pm$ 0.94	140.06 $\pm$ 1.08	168.80 $\pm$ 1.56
+ Ber(0.05)	125.55 $\pm$ 1.17	111.34 $\pm$ 0.94	142.36 $\pm$ 1.15	172.96 $\pm$ 1.63
+ Ber(0.1)	126.12 $\pm$ 1.19	**111.56** $\pm$ 0.94	145.23 $\pm$ 1.18	177.20 $\pm$ 1.68
+ Ber(0.2)	**126.86** $\pm$ 1.22	111.53 $\pm$ 0.96	149.32 $\pm$ 1.21	182.42 $\pm$ 1.71
+ Ber(0.3)†	126.65 $\pm$ 1.24	111.43 $\pm$ 0.96	151.22 $\pm$ 1.22	184.55 $\pm$ 1.74
+ Ber(0.4)	125.86 $\pm$ 1.22	110.71 $\pm$ 0.95	**151.29** $\pm$ 1.21	**184.58** $\pm$ 1.73
+ Ber(0.5)	—	—	150.70 $\pm$ 1.19	183.09 $\pm$ 1.71

D Additional Experimental Results

D.1 Effect of Bernoulli augmentation on AttriCoT

Table 6 shows the AUPC values that result from applying the Bernoulli augmentation described in Appendix C.3.1 to AttriCoT-LOO. As the probability π of perturbing additional units increases, AUPC decreases monotonically for every model, and plain AttriCoT-LOO ($\pi = 0$) is significantly best for all four (paired t-tests between AttriCoT-LOO and every augmented variant yield p-values $< 10^{-14}$). For this reason, $\pi = 0.05, 0.1$ were not run for the slower DS-Qwen3-8B and Qwen3-8B models.

Table 7 shows the results of applying Bernoulli augmentation to AttriCoT-2x. Given the above finding that $\pi = 0$ yields the best AUPC for AttriCoT-LOO, we augment only the leave-two-out interventions of the base AttriCoT-2x variant, not its LOO interventions. In contrast to Table 6, augmentation increases AUPC for all models, most significantly for DS-Qwen3-8B and Qwen3-8B. While the best value of π varies with the model, we find that $\pi = 0.3$ is the single best choice because it is not significantly worse than each model's optimal π, as shown in Table 8. We therefore report the results for $\pi = 0.3$ in Table 1 in the main body and carry $\pi = 0.3$ forward as the one AttriCoT-2x variant evaluated on the remaining non-GSM8K datasets. For $\pi > 0.3$ however, AUPC either saturates or declines; $\pi = 0.5$ was not run for DS-Llama-8B and DS-Qwen-14B because $\pi = 0.4$ is already significantly worse than these models' optimal π values.

Table 8: Paired t-tests comparing Bernoulli probability $\pi = 0.3$ to each LRM's optimal π from Table 7 (per-sample AUPC values on GSM8K, $n = 1319$). The difference is not statistically significant for any LRM.

Model	Optimal π	Optimum $-$ ($\pi = 0.3$)	p-value
DS-Llama-8B	0.2	$+0.103 \pm 0.093$	0.27
DS-Qwen-14B	0.1	$+0.062 \pm 0.084$	0.46
DS-Qwen3-8B	0.4	$+0.032 \pm 0.069$	0.64
Qwen3-8B	0.4	$+0.013 \pm 0.145$	0.93

Table 9: AUPC (↑) on GSM8K of a masking variant of AttriCoT (AttriCoT-mask) compared to AttriCoT-LOO and TA-KL, where perturbation curves are computed by removing units. **Bold**: best per model. All pairwise differences are significant (paired t-test, $p < 10^{-10}$, $n = 1319$) except for AttriCoT-mask − TA-KL on DS-Llama-8B ($p = 0.74$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**123.80** ± 1.15	**110.44** ± 0.92	**138.64** ± 1.05	**166.59** ± 1.53
AttriCoT-mask	120.19 ± 1.16	109.53 ± 0.92	136.71 ± 1.03	163.81 ± 1.52
TA-KL (no log)	120.30 ± 1.15	107.91 ± 0.94	123.42 ± 1.20	161.28 ± 1.60

Table 10: AUPC (↑) on GSM8K when perturbation curves are computed by *masking* units rather than removing them, matching AttriCoT-mask's perturbation type instead of that of AttriCoT-LOO and TA-KL. **Bold**: best per model. All pairwise differences are significant (paired t-test, $p < 10^{-3}$, $n = 1319$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	109.41 ± 1.01	96.75 ± 0.76	**126.15** ± 0.92	**146.30** ± 1.22
AttriCoT-mask	**109.95** ± 1.01	**97.58** ± 0.76	125.89 ± 0.92	145.75 ± 1.21
TA-KL (no log)	105.85 ± 1.00	94.33 ± 0.77	120.00 ± 0.82	142.85 ± 1.23

D.2 Sensitivity studies

D.2.1 Changing the perturbation: Removal versus masking

In the evaluation of Section 5, there are two places where units are perturbed: in the attribution methods (AttriCoT and TA-KL specifically), and in the evaluation of perturbation curves. For the main results in this paper, units were perturbed simply by removing them, in all of the above places. However, units can be perturbed in other ways, and it is possible that removing units could create appreciable distribution shifts, for example by decreasing the coherence of the text or shifting the token positions of subsequent units.

To address potential sensitivity to the perturbation type, we performed experiments where instead of removing a unit, we replace it with an equal number of mask tokens, so that the token positions of subsequent units do not change. The underscore ("_") was chosen as the mask token because all 4 LRMs that we use encode it consistently as a single, non-special token.

In Table 9, we compare the AUPC of a masking variant of AttriCoT (AttriCoT-mask) to AttriCoT-LOO (original removal variant) and TA-KL (also removal) on the GSM8K dataset, where perturbation curves are also computed using removal. The table shows that the change in perturbation type has a small effect on the AUPC of AttriCoT, costing between 0.9 and 3.6 AUPC units across the four LRMs. AttriCoT-mask nonetheless remains ahead of TA-KL on three of the four, the exception being DS-Llama-8B, where the two are effectively tied (-0.11 AUPC units, paired t-test $p = 0.74$).

We attribute AttriCoT-mask's slightly worse AUPC to its mismatch with the perturbation curve evaluation, which still removes units. If we instead compute perturbation curves by masking units (with an equal number of mask tokens) rather than removing them, then most of its deficit disappears, as seen in Table 10: AttriCoT-mask overtakes AttriCoT-LOO on DS-Llama-8B and DS-Qwen-14B, while on the two Qwen3-based models AttriCoT-LOO stays ahead by a small margin. Similarly, the tie between AttriCoT-mask and TA-KL on DS-Llama-8B in Table 9 becomes a significant difference of 4.1 AUPC units ($p = 8 \times 10^{-54}$) in favor of AttriCoT-mask in Table 10, where AttriCoT-mask remains ahead on the three other models.

Overall, the effect of the perturbation type is minor either way and can be largely attributed to the match or mismatch with the evaluation.

Table 11: AUPC (↑) on GSM8K under re-generative evaluation, where the change between re-generated and original output units is quantified by **BERTScore**. **Bold**: best among methods using $\leq S + T$ forward passes/generative calls. Underline: best overall. A method is also marked when it is not significantly worse than the corresponding best (paired t-test, $p > 0.05$, $n = 1319$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**32.42** $\pm$ 0.25	**35.53** $\pm$ 0.24	**26.10** $\pm$ 0.23	**39.77** $\pm$ 0.24
AttriCoT-2x	32.21 $\pm$ 0.25	35.52 $\pm$ 0.24	26.23 $\pm$ 0.23	39.73 $\pm$ 0.24
+ Ber(0.3)	31.00 $\pm$ 0.27	34.49 $\pm$ 0.24	25.88 $\pm$ 0.24	39.21 $\pm$ 0.25
TA-KL (no log)	31.75 $\pm$ 0.26	34.73 $\pm$ 0.25	25.77 $\pm$ 0.24	**39.95** $\pm$ 0.25
Prompted (best same-size)	17.93 $\pm$ 0.24	27.30 $\pm$ 0.23	16.13 $\pm$ 0.17	21.22 $\pm$ 0.22
Prompted-ICL (GPT-OSS)	20.83 $\pm$ 0.22	25.18 $\pm$ 0.23	16.01 $\pm$ 0.18	22.16 $\pm$ 0.19

D.2.2 Changing the measured effect: Existing output units versus re-generated units

Recall from Section 1 that AttriCoT targets a *local* notion of causality, referring to restriction to the single, given CoT. Causal effects are quantified "locally," through conditional log probabilities of steps in the given CoT, without reference to alternative CoTs. The evaluation of perturbation curves has been done in the same way (as described in Section 5.1 and Appendix C.4) for the main results in Section 5.2, thus also addressing local causality.

With the above in mind, it is also of interest to evaluate AttriCoT and the other attribution methods under a "re-generative" notion of causality, where we measure effects on output unit re-generation. We thus performed an alternative evaluation of perturbation curves and AUPCs, where for each output unit, we take an attribution method's existing attribution scores and remove prior units in decreasing order of importance, as before. But instead of computing the log probability of the given output unit, we re-generate an output unit conditioned on the perturbed prefix and quantify the change between the re-generated and original output units by computing text similarity. We use three metrics for quantifying text similarity: BERTScore, sentence embedding similarity, and bi-directional NLI entailment scores.

Tables 11–13 show the AUPCs that result from this re-generative evaluation on GSM8K, with one table for each text similarity metric. (Due to re-generation, this evaluation is an order of magnitude more expensive than log probability evaluation and is therefore prohibitive to extend to the other datasets.) The tables include the strongest methods of Table 1, together with the Bernoulli-augmented variant of Appendix D.1. We make the following observations:

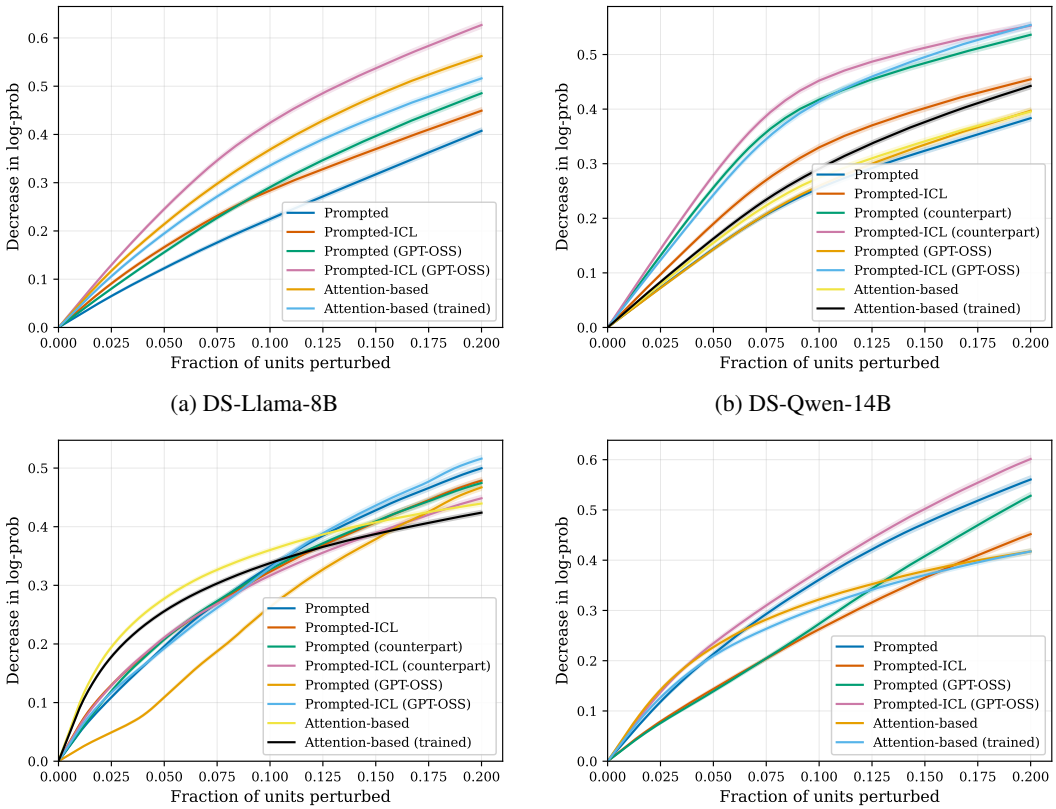
- **AttriCoT-LOO vs. TA-KL**: AttriCoT-LOO remains the best among methods that use $\leq S + T$ forward passes or generative calls; it is best or tied for best across all models and similarity metrics. TA-KL does narrow the gap under re-generative evaluation, compared to Tables 1 and 2. In the case of Qwen3-8B in Tables 11 and 12, TA-KL is marginally better than AttriCoT-LOO but not significantly so; for DS-Llama-8B in Table 13, TA-KL is worse but not significantly so. However, the remaining 9 differences between AttriCoT-LOO and TA-KL are statistically significant.
- **AttriCoT-2x**: Unlike under log probability evaluation, AttriCoT-2x does not improve on AttriCoT-LOO in any of the cases; the two are statistically indistinguishable in every case (paired t-test, $p > 0.05$). Similarly, Bernoulli augmentation, which yielded a significant benefit under log probability evaluation (Table 7), does not transfer to re-generative evaluation: Ber(0.3) is below base AttriCoT-2x in all 12 cases.
- Both **prompting** baselines remain far behind, as they are under log probability evaluation.
- The three similarity metrics are largely consistent with each other on the ordering of methods.

Table 12: AUPC (↑) on GSM8K under re-generative evaluation, where the change between re-generated and original output units is quantified by **sentence embedding similarity**. **Bold**: best among methods using $\leq S + T$ forward passes/generative calls. Underline: best overall. A method is also marked when it is not significantly worse than the corresponding best (paired t-test, $p > 0.05$, $n = 1319$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**39.57** $\pm$ 0.35	**42.46** $\pm$ 0.33	**38.59** $\pm$ 0.37	**48.60** $\pm$ 0.36
AttriCoT-2x	39.19 $\pm$ 0.35	42.57 $\pm$ 0.34	38.86 $\pm$ 0.37	48.60 $\pm$ 0.36
+ Ber(0.3)	37.51 $\pm$ 0.36	41.07 $\pm$ 0.33	37.94 $\pm$ 0.37	47.53 $\pm$ 0.37
TA-KL (no log)	38.55 $\pm$ 0.36	41.45 $\pm$ 0.35	38.20 $\pm$ 0.38	**48.74** $\pm$ 0.37
Prompted (best same-size)	21.10 $\pm$ 0.32	31.64 $\pm$ 0.31	23.73 $\pm$ 0.27	26.09 $\pm$ 0.32
Prompted-ICL (GPT-OSS)	24.69 $\pm$ 0.30	29.07 $\pm$ 0.31	22.87 $\pm$ 0.28	26.55 $\pm$ 0.27

Table 13: AUPC (↑) on GSM8K under re-generative evaluation, where the change between re-generated and original output units is quantified by **bi-directional NLI entailment**. **Bold**: best among methods using $\leq S + T$ forward passes/generative calls. Underline: best overall. A method is also marked when it is not significantly worse than the corresponding best (paired t-test, $p > 0.05$, $n = 1319$).

Algorithm	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
AttriCoT-LOO	**501.16** $\pm$ 5.13	**585.74** $\pm$ 5.01	**347.26** $\pm$ 5.55	**541.21** $\pm$ 5.06
AttriCoT-2x	497.07 $\pm$ 5.18	584.92 $\pm$ 4.95	347.68 $\pm$ 5.55	539.24 $\pm$ 5.13
+ Ber(0.3)	486.64 $\pm$ 5.44	571.35 $\pm$ 4.95	340.25 $\pm$ 5.52	534.12 $\pm$ 5.18
TA-KL (no log)	**493.90** $\pm$ 5.18	568.52 $\pm$ 5.28	335.57 $\pm$ 5.49	532.08 $\pm$ 5.05
Prompted (best same-size)	280.63 $\pm$ 4.56	450.14 $\pm$ 4.66	203.86 $\pm$ 3.59	286.58 $\pm$ 4.05
Prompted-ICL (GPT-OSS)	351.69 $\pm$ 4.71	445.07 $\pm$ 4.82	211.09 $\pm$ 3.86	294.26 $\pm$ 3.66



(a) DS-Llama-8B

(b) DS-Qwen-14B

(c) DS-Qwen3-8B

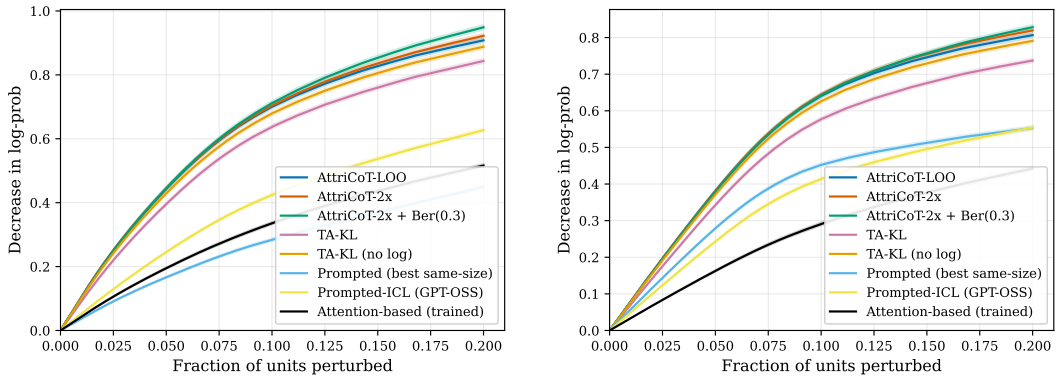
(d) Qwen3-8B

Figure 6: Perturbation curves of all prompting baselines and attention-based methods on GSM8K.

D.3 Perturbation curves

Figures 6–11 plot the full perturbation curves corresponding to the AUPC results in Tables 1 and 2. For the GSM8K dataset, since many attribution algorithms were compared in Table 1, we split the perturbation curves across Figures 6 and 7. Figure 6 compares all of the prompting baselines and the attention intervention methods, which are competitive with each other. Figure 7 then compares the best-performing prompting methods and trained attention intervention with the causal attribution methods (AttriCoT variants and TA-KL). Figures 8–11 compare the top-performing methods included in Table 2 on the MATH500, MMLU-Pro, GPQA, and ZebraLogic datasets.

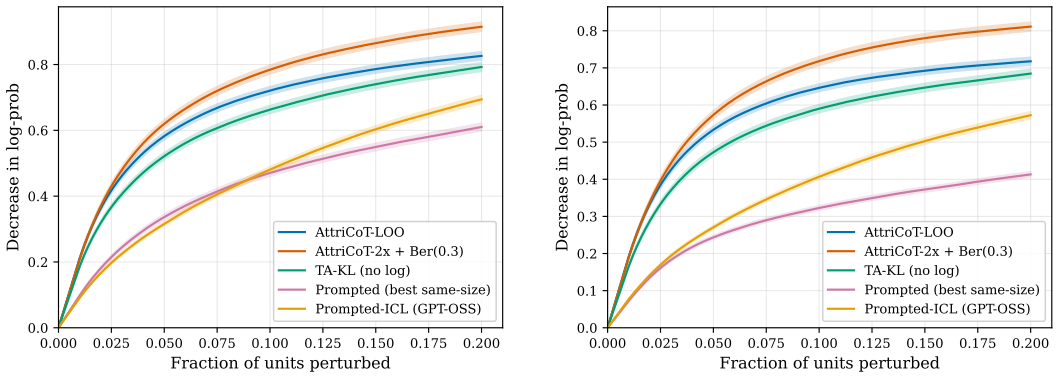
A better perturbation curve is one that shows a more rapid decrease in log probability of output units as more preceding units are perturbed, i.e., a curve that rises more quickly. The curves in Figures 6–11 thus mirror the AUPC values in Tables 1 and 2, confirming that AttriCoT's attributions are consistently the most behaviorally faithful to the CoT model's log probabilities.



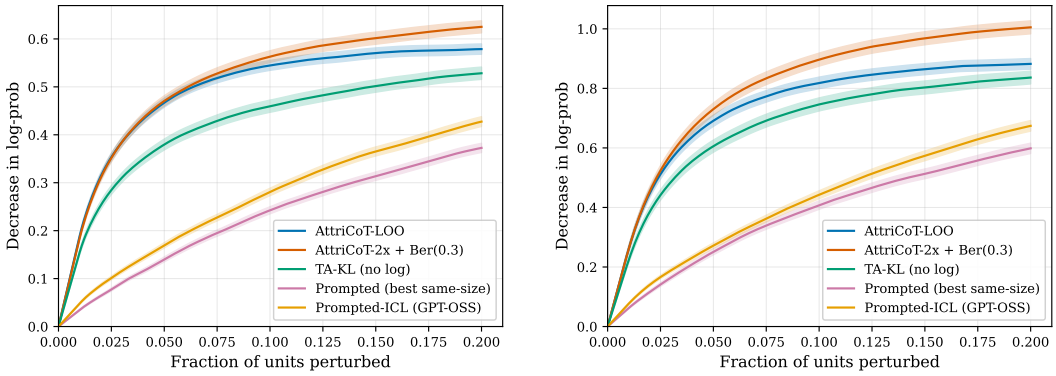
(a) DS-Llama-8B (b) DS-Qwen-14B

(c) DS-Qwen3-8B (d) Qwen3-8B

Figure 7: Perturbation curves of all causal attribution methods and best-performing prompting and attention-based methods on GSM8K.

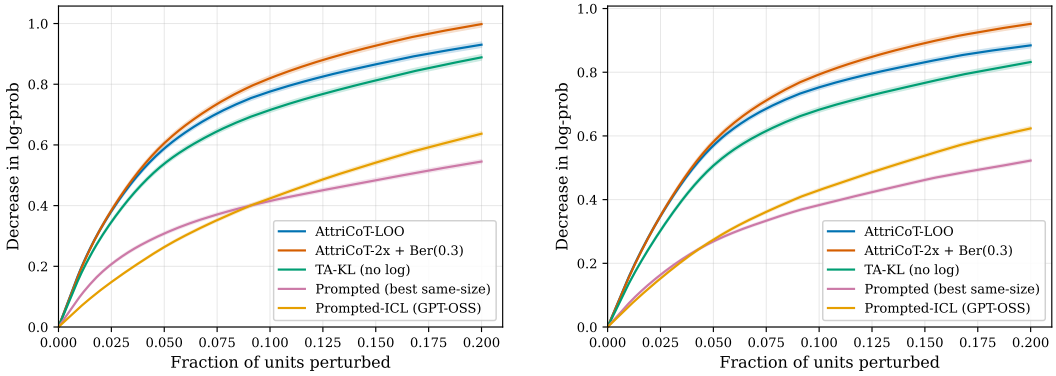


(a) DS-Llama-8B (b) DS-Qwen-14B

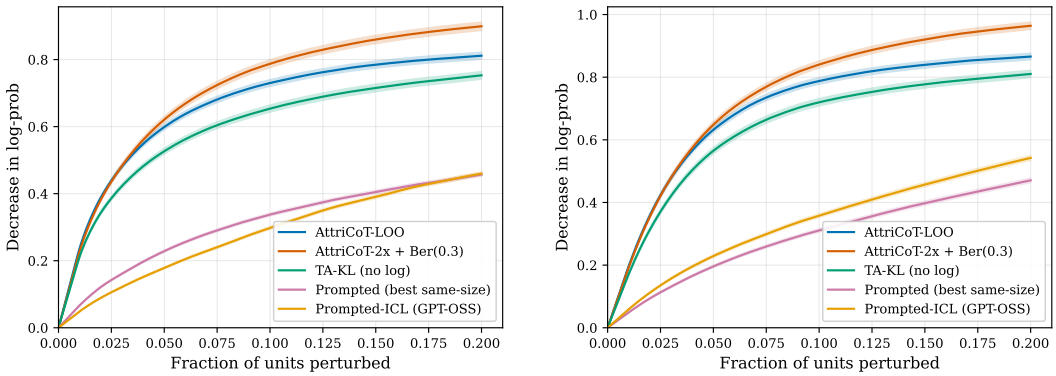


(c) DS-Qwen3-8B (d) Qwen3-8B

Figure 8: Perturbation curves on MATH500.

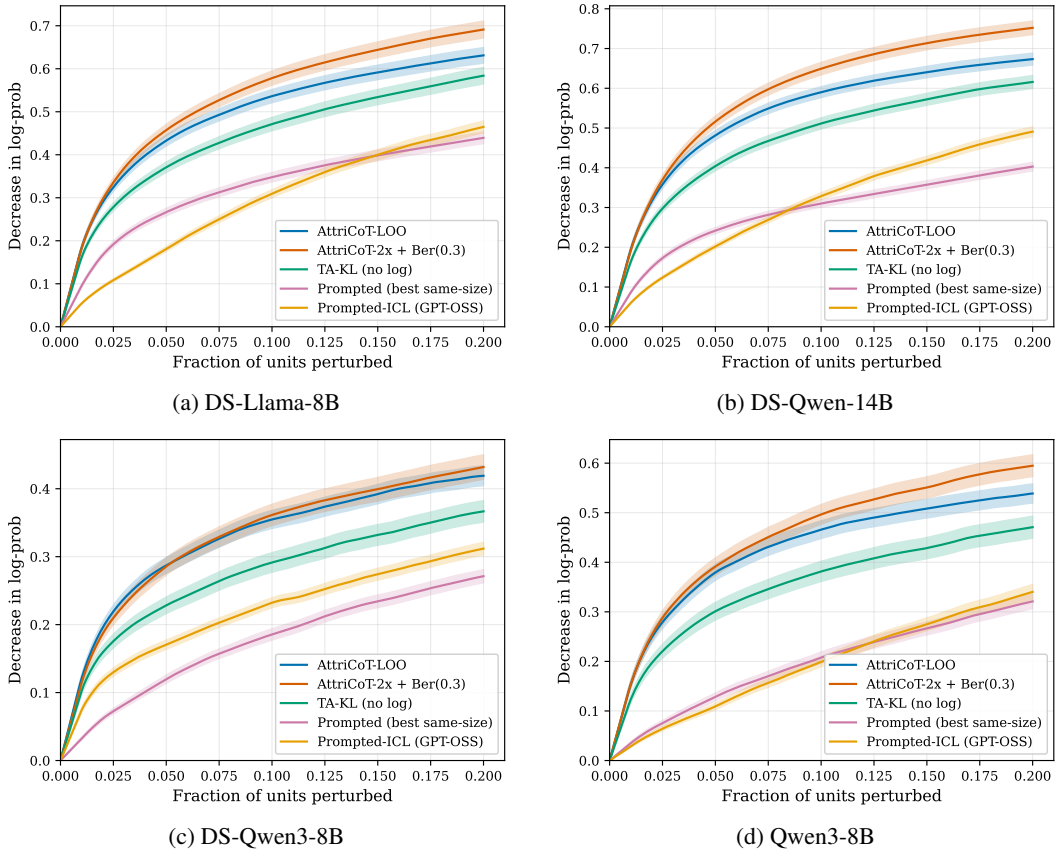


(a) DS-Llama-8B (b) DS-Qwen-14B



(c) DS-Qwen3-8B (d) Qwen3-8B

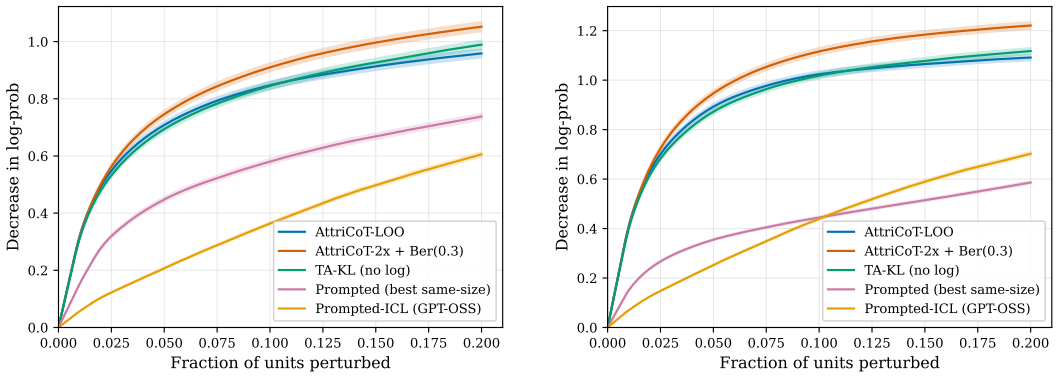
Figure 9: Perturbation curves on MMLU-Pro.



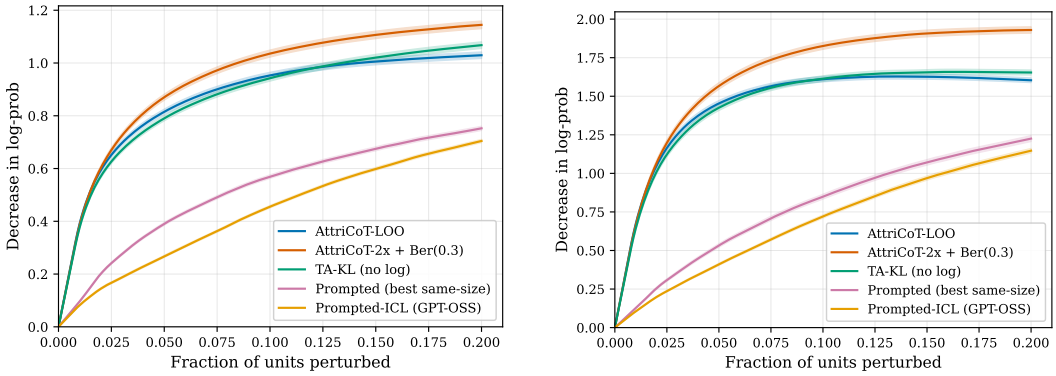
(a) DS-Llama-8B (b) DS-Qwen-14B

(c) DS-Qwen3-8B (d) Qwen3-8B

Figure 10: Perturbation curves on GPQA.



(a) DS-Llama-8B (b) DS-Qwen-14B



(c) DS-Qwen3-8B (d) Qwen3-8B

Figure 11: Perturbation curves on ZebraLogic.

Table 14: AUPC (↑) of the white-box attention intervention method on GSM8K, intervening on different Transformer blocks. **Bold**: best block per model, for each of the untrained and trained versions.

Model	Block	Untrained	Trained
DS-Llama-8B	15	**68.27** $\pm$ 0.83	**62.19** $\pm$ 0.78
	19	58.50 $\pm$ 0.75	57.07 $\pm$ 0.78
	23	53.89 $\pm$ 0.77	59.83 $\pm$ 0.81
	27	50.62 $\pm$ 0.69	51.40 $\pm$ 0.69
	31	52.66 $\pm$ 0.73	45.32 $\pm$ 0.65
DS-Qwen-14B	23	33.13 $\pm$ 0.60	37.17 $\pm$ 0.68
	29	28.79 $\pm$ 0.57	39.02 $\pm$ 0.61
	35	41.46 $\pm$ 0.63	**53.26** $\pm$ 0.72
	41	37.26 $\pm$ 0.59	39.80 $\pm$ 0.64
	47	**49.07** $\pm$ 0.80	48.31 $\pm$ 0.78
DS-Qwen3-8B	17	42.72 $\pm$ 0.48	39.48 $\pm$ 0.48
	21	33.91 $\pm$ 0.42	50.88 $\pm$ 0.55
	26	**65.72** $\pm$ 0.61	**61.88** $\pm$ 0.58
	31	48.09 $\pm$ 0.54	52.04 $\pm$ 0.55
	35	31.51 $\pm$ 0.46	36.99 $\pm$ 0.54
Qwen3-8B	17	54.48 $\pm$ 0.72	40.12 $\pm$ 0.66
	21	42.54 $\pm$ 0.61	54.05 $\pm$ 0.73
	26	**58.31** $\pm$ 0.73	**56.02** $\pm$ 0.71
	31	42.14 $\pm$ 0.59	55.63 $\pm$ 0.65
	35	39.40 $\pm$ 0.59	39.82 $\pm$ 0.60

D.4 Attention intervention method

We evaluated the attention intervention method of Appendix B only on the GSM8K dataset. Table 14 presents the full AUPC performance results, including both untrained and trained versions and interventions on different Transformer blocks within each model. Five Transformer block indices were selected to span depths 50%–100% of each model (index 0 is the first block). The block for which the highest AUPC is obtained varies across models: For DS-Llama-8B, it is the shallowest block considered, at half depth (block 15 of 0–31); for DS-Qwen3-8B and Qwen3-8B, it is the block at three-quarters depth; for DS-Qwen-14B, three-quarters depth is best for the trained version and maximum depth for the untrained one. We also observe that while training increases AUPC in 12 of the 20 cases, it improves the best AUPC only for DS-Qwen-14B.

Comparing the bolded values in Table 14 with Table 1, one can see that attention intervention is mostly competitive with the prompting baselines at much lower inference cost (the attention interventions plus shallow network inference versus full generative calls). However, its performance is not better than the strongest prompting methods and is far below that of the causal intervention methods, AttriCoT and TA-KL. For these reasons, we did not evaluate attention intervention on the other datasets and leave further development of a white-box method to future work.

Table 15: Score density metric (average absolute score) applied to AttriCoT's attributions.

Dataset	DS-Llama-8B	DS-Qwen-14B	DS-Qwen3-8B	Qwen3-8B
GSM8K	0.1417 ± 0.0018	0.1465 ± 0.0013	0.0471 ± 0.0010	0.0744 ± 0.0010
MATH500	0.0506 ± 0.0018	0.0541 ± 0.0018	0.0279 ± 0.0010	0.0522 ± 0.0022
MMLU-Pro	0.0535 ± 0.0009	0.0639 ± 0.0009	0.0345 ± 0.0007	0.0477 ± 0.0009
GPQA	0.0225 ± 0.0017	0.0326 ± 0.0020	0.0147 ± 0.0007	0.0198 ± 0.0011
ZebraLogic	0.0316 ± 0.0013	0.0420 ± 0.0014	0.0288 ± 0.0009	0.0525 ± 0.0015

D.5 Analysis of attribution scores

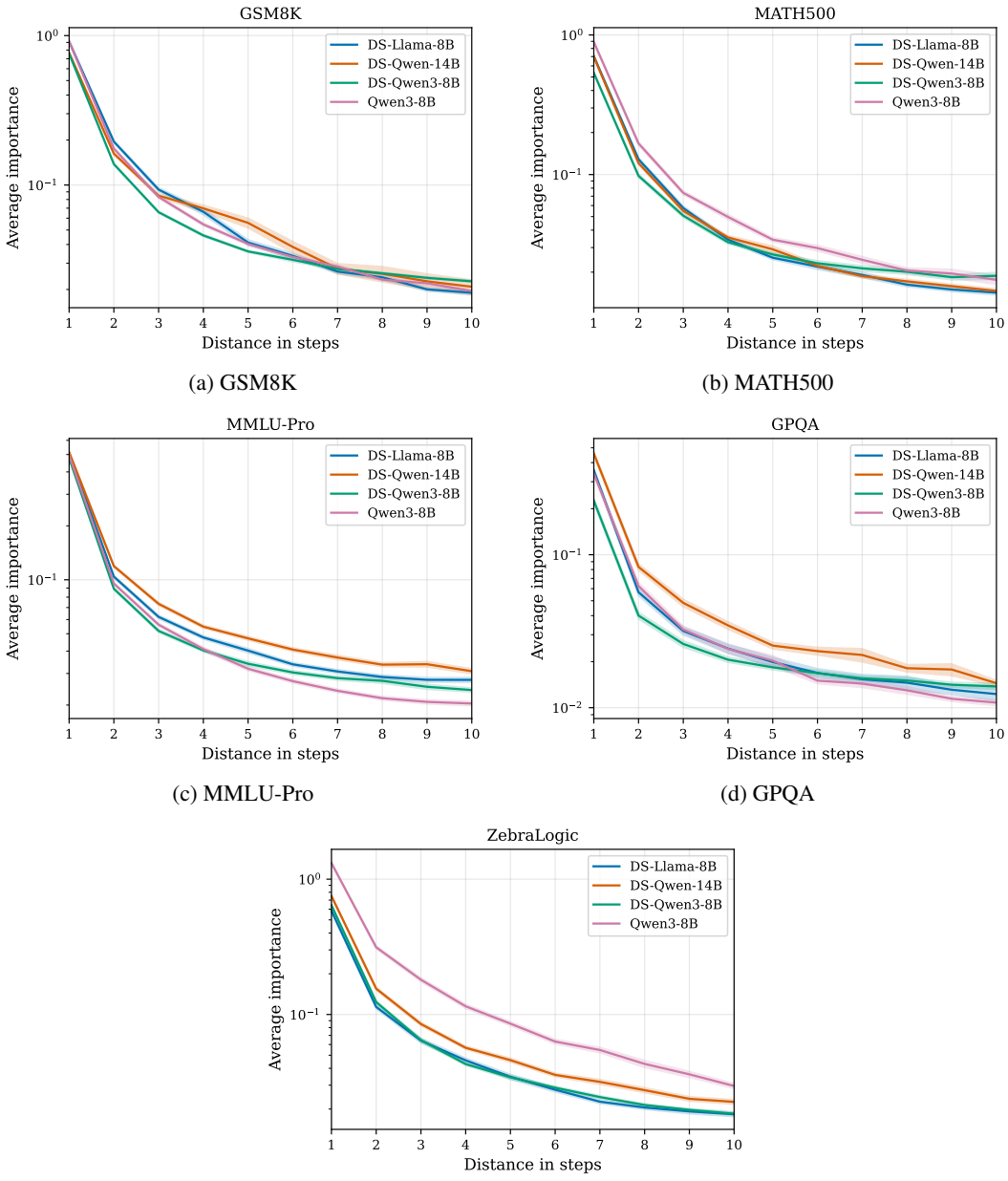
In this appendix, we show the full set of results for the metrics discussed in Section 6 for summarizing AttriCoT's attribution score matrices ($\boldsymbol{A}$ and $\boldsymbol{B}$).

Score density Table 15 shows values of the score density metric (average absolute score, (12)) for different pairs of datasets and CoT models (mean and standard error over the samples in each dataset). In general, it appears that score density varies inversely with the length of the model output in terms of number of units. Densities are highest for the GSM8K dataset, which has the shortest CoTs because of the simplicity of the tasks. Densities are lowest for GPQA and ZebraLogic, which tend to have the longest CoTs, while MATH500 and MMLU-Pro are in the middle. Similarly among models, DS-Qwen-14B has the shortest CoTs and correspondingly the highest densities on four of the five datasets, while DS-Qwen3-8B has the longest CoTs (notwithstanding the merging of steps into no more than 100 steps) and the lowest densities on all five. The ordering of the remaining two models is dataset-dependent: DS-Llama-8B is second-highest on GSM8K, MMLU-Pro and GPQA, whereas Qwen3-8B is second on MATH500 and the highest of all four on ZebraLogic.

Importance by distance Figure 12 plots the importance of CoT units to later CoT units (measured by mean absolute score) as a function of the number of steps between them (i.e., absolute distance, see (15)). Each curve represents the mean over the dataset samples and the shading indicates one standard error above and below the mean.

Generally, the decay in importance is rapid in the first few steps and then flattens to a long tail. The rates of decay are similar for the four models. The clearest exception is Qwen3-8B on the ZebraLogic dataset (Figure 12e), where its decay is noticeably slower than for the other models.

Figure 13 shows an alternative view of importance by distance, where the distance has been normalized to a fraction $d/(T_{\text{CoT}} - 1)$ of the largest possible distance $T_{\text{CoT}} - 1$. To obtain the curves in the figure, the importance-by-distance values for each dataset sample are first interpolated onto a common grid of normalized distances before averaging over samples.

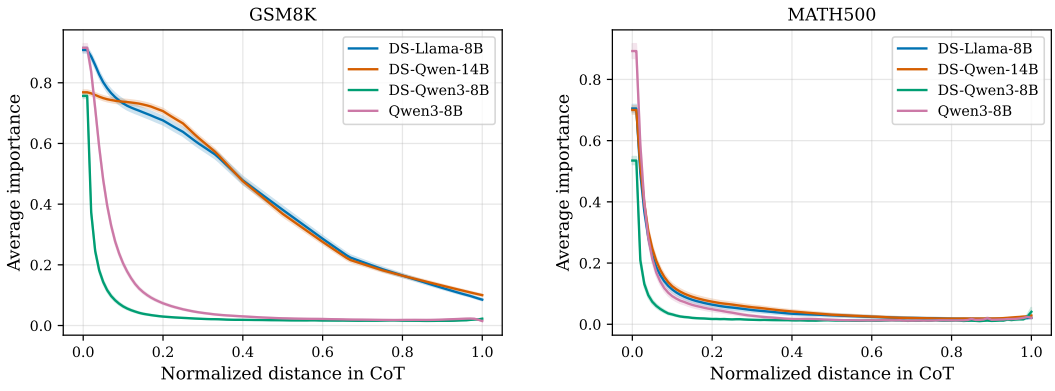


(a) GSM8K (b) MATH500

(c) MMLU-Pro (d) GPQA

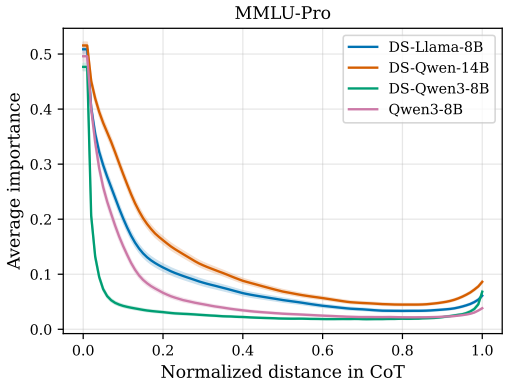
(e) ZebraLogic

Figure 12: Average importance of CoT units to later CoT units, estimated by AttriCoT, as a function of the distance in steps between the units.

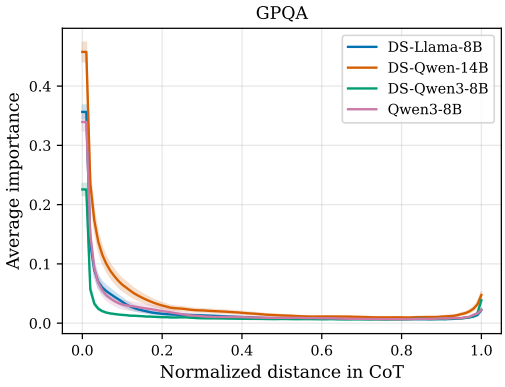


(a) GSM8K

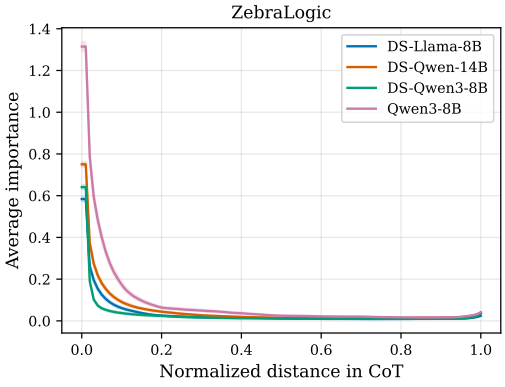
(b) MATH500



(c) MMLU-Pro



(d) GPQA



(e) ZebraLogic

Figure 13: Average importance of CoT units to later CoT units, estimated by AttriCoT, as a function of the normalized distance between the units.

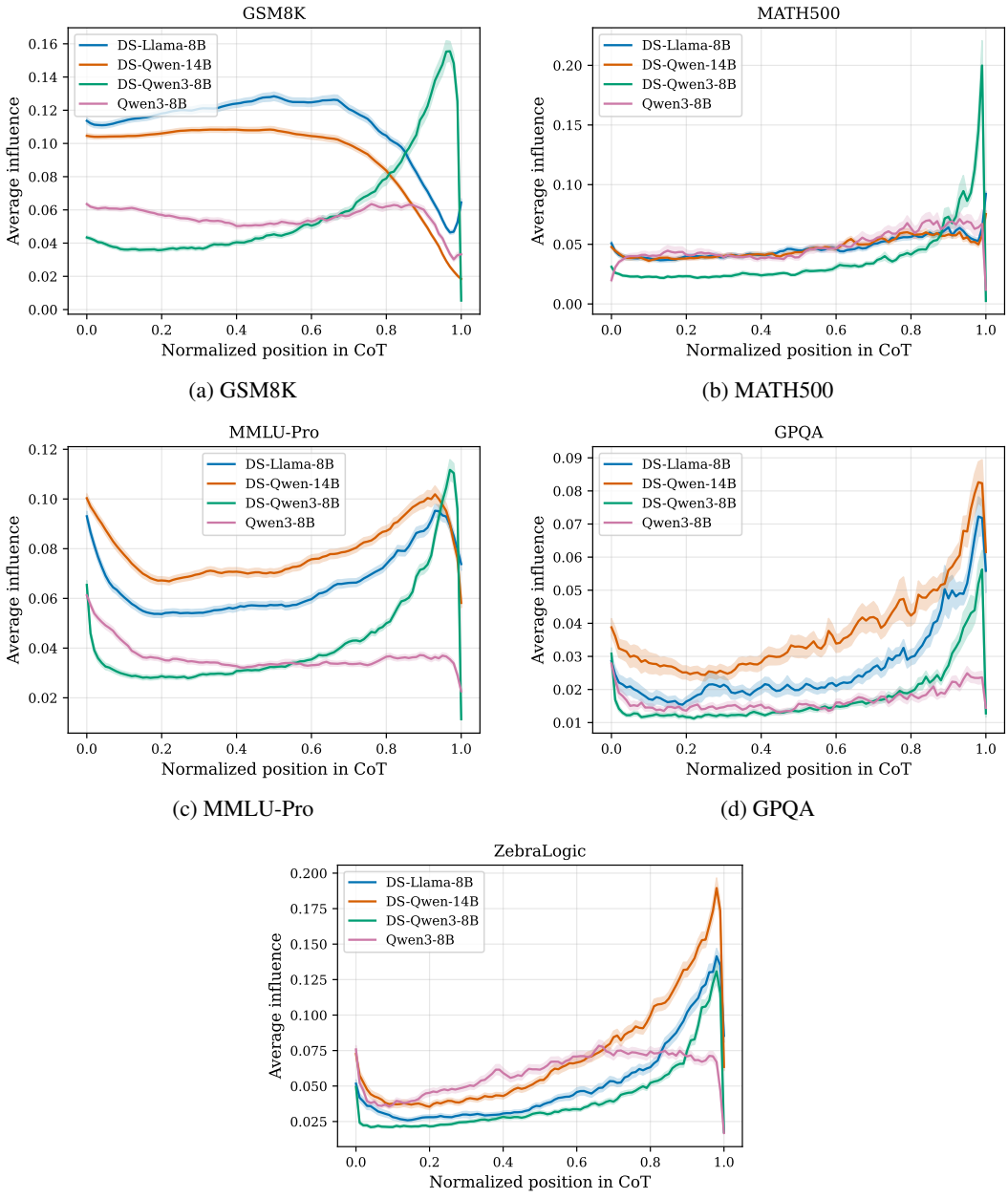
Unweighted average influence In Figure 14, we plot the average influence of CoT units on subsequent output units (now including final answer units), where we take the unweighted average over these subsequent units (13). The curves are plotted as a function of normalized CoT position $\tau/T_{\text{CoT}} \in [0, 1]$ in a similar manner as in Figure 13, i.e., by interpolating onto a common grid of positions, averaging, and computing standard errors. Many curves in Figure 14 (DS-Llama-8B and DS-Qwen-14B in Figures 14c–14e, and DS-Qwen3-8B on all datasets) exhibit a steep rise as the normalized position approaches 1.0. We believe that these steep rises are an artifact of taking the unweighted average. In (13), as τ decreases (going from right to left in the plots), the denominator $T - \tau$ increases linearly. At the same time, due to the decay of attribution scores with distance seen in Figure 12, the sum of absolute scores in (13) quickly saturates. The combination of these two effects may account for the $1/(T - \tau)$-like behavior in Figure 14.

Exponentially-weighted average influence To address the apparent artifact in Figure 14, we also compute exponentially-weighted average influence (14) with decay rate $r \in (0, 1)$. The exponential weighting addresses the growing denominator problem in (13) by mostly confining the average to units within a *characteristic distance*, proportional to $1/\log(1/r)$. At the same time, exponential weighting does not completely exclude more distant output units like a sliding window would. We use this interpretation of characteristic distance to set r. Specifically, we note that the smallest number of downstream units occurs at $\tau = T_{\text{CoT}}$, where it equals the number of final answer units. Thus, for each dataset-model pair, we compute the average number $\bar{T}_{\text{FA}}$ of final answer units. We then set r so that 90% of the sum of the exponential weights occurs within the first $\bar{T}_{\text{FA}}$ weights, yielding the equations

$$1 - r^{\bar{T}_{\text{FA}}} = 0.9 \quad \Longrightarrow \quad r = (0.1)^{1/\bar{T}_{\text{FA}}}. \tag{19}$$

This setting of r implies that even for the last position $\tau = T_{\text{CoT}}$ in the CoT, most of the mass in the exponential weights falls before the end of the output sequence.

Figure 15 shows the curves for exponentially-weighted average influence with decay rate determined as described above. The exponential weighting removes the steeply rising artifact seen in Figure 14. Generally, the curves now have a wide middle range that is approximately flat, a sharp decrease at the end of the CoT, and often a significant decrease at the beginning as well. This shape suggests that early CoT steps have the most influence on the subsequent trace, middle steps have roughly constant influence, and influence declines quickly as the reasoning concludes. The curves for DS-Qwen3-8B follow the pattern for all five datasets. The three other models however deviate from the pattern on the math datasets (Figures 15a, 15b) at the beginning of the CoT: DS-Llama-8B and DS-Qwen-14B show no initial decrease, while for Qwen3-8B, there is a gradual decrease there on GSM8K but an increase on MATH500.

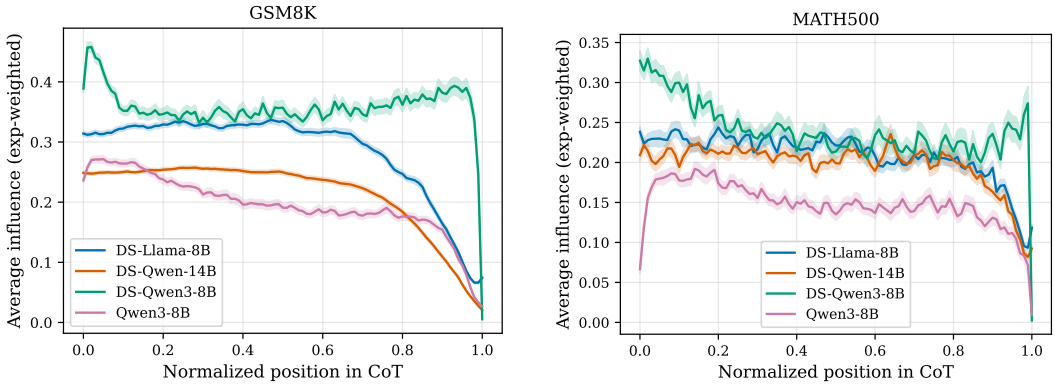


(a) GSM8K (b) MATH500

(c) MMLU-Pro (d) GPQA

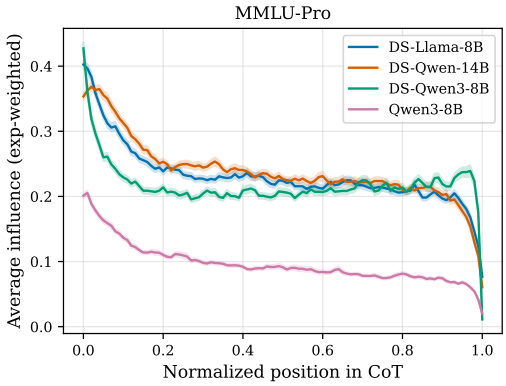
(e) ZebraLogic

Figure 14: Average influence of CoT units on subsequent output units, estimated by AttriCoT, as a function of normalized position in the CoT.

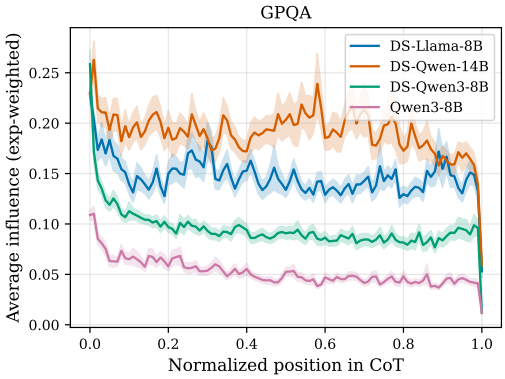


(a) GSM8K

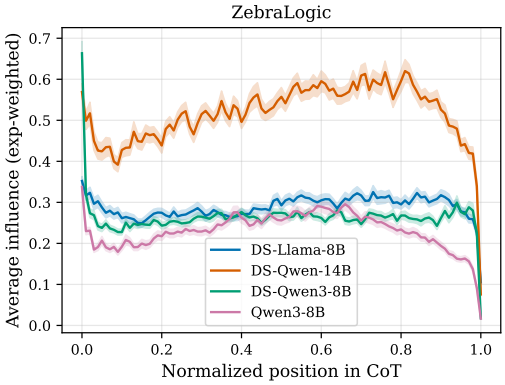
(b) MATH500



(c) MMLU-Pro



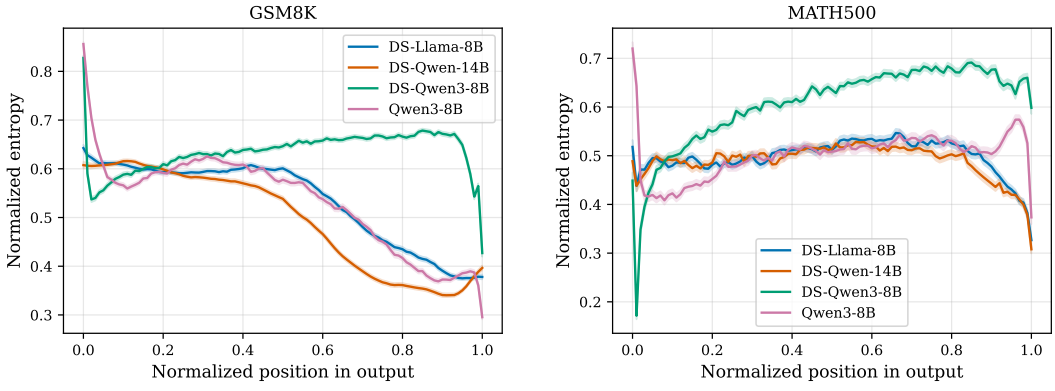
(d) GPQA



(e) ZebraLogic

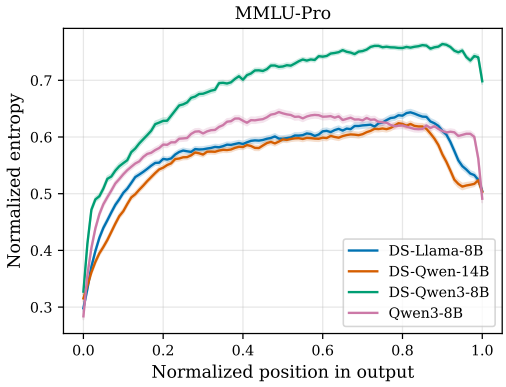
Figure 15: Exponentially-weighted average influence of CoT units on subsequent output units, estimated by AttriCoT, as a function of normalized position in the CoT.

Entropy Figure 16 shows the normalized entropy of output units' attribution scores (normalized to the entropy of a uniform distribution) as a function of the output unit's position. Among the four models, DS-Qwen3-8B's attribution scores generally have the highest entropy, implying that its attributions are more diffuse, even after correcting for the model's longer CoTs by normalizing the entropy. In most cases on the math datasets (GSM8K, MATH500) as well as ZebraLogic, the entropy is initially higher but quickly drops; this behavior is not present for the general/multiple-domain datasets (MMLU-Pro, GPQA). As the position in the output increases, entropy tends to gradually increase or stay the same (except for the non-DS-Qwen3-8B models on GSM8K), before sharply decreasing at the end. For ZebraLogic in particular, entropy builds to a peak before the final decrease.

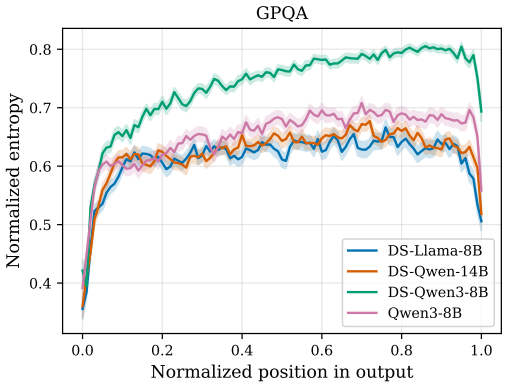


(a) GSM8K

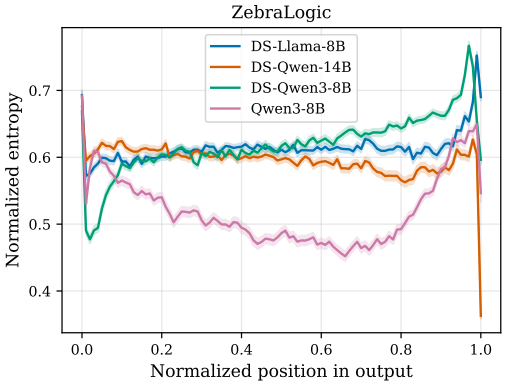
(b) MATH500



(c) MMLU-Pro

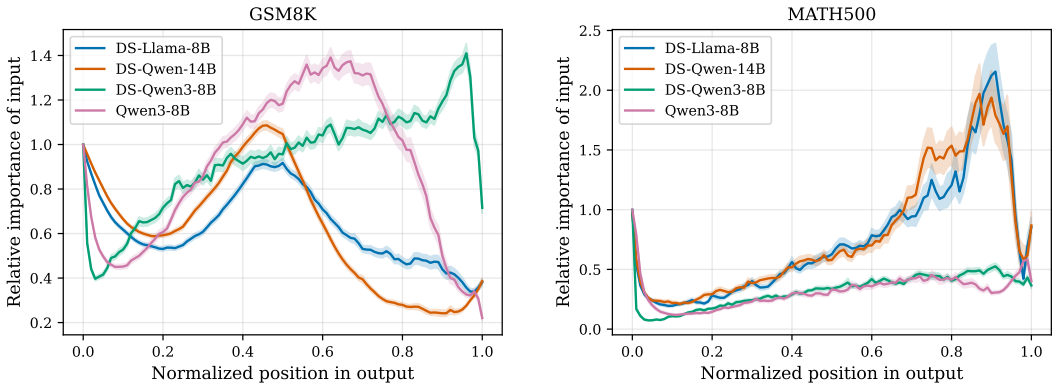


(d) GPQA



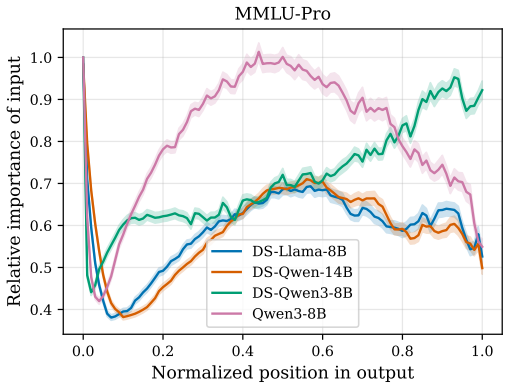
(e) ZebraLogic

Figure 16: Normalized entropy of AttriCoT's attribution scores for output units as a function of their position in the output.

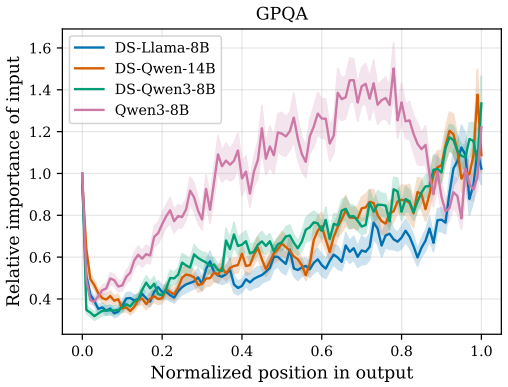


(a) GSM8K

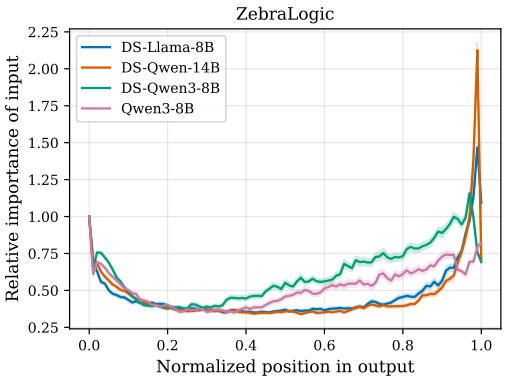
(b) MATH500



(c) MMLU-Pro



(d) GPQA



(e) ZebraLogic

Figure 17: Relative importance of input units as a function of normalized position in output.

Input ratio. Figure 17 plots the input ratio metric for output units, including for the MATH500 and GPQA datasets that were omitted from Figure 3 in Section 6.2. The patterns for MATH500 and GPQA are as described in Section 6.2.