Demystifying Variance in Circuit Discovery of LLMs

Frank Zhengqing Wu [1] **Francesco Tonin** [1] **Volkan Cevher** [1]

Abstract

Circuit discovery is a key technique in mechanistic interpretability to pinpoint the model components that are crucial for performing a given task. Although the current state-of-the-art method (EAP-IG) performs well on the metric of (un)faithfulness, it suffers from substantial variability. This includes *resampling variance*, where the circuit changes when we probe with a new batch of data from the same distribution; *rephrasing variance*, where the discovered circuit shifts when the prompts are rephrased; and *sample-wise variance*, where a circuit with low population unfaithfulness exhibits large fluctuations in unfaithfulness across individual samples.

This paper studies the roots of these variances. We demonstrate that CEAP, our new circuit discovery method that improves upon EAP-IG with a theoretical guarantee, can substantially lessen resampling variance. We further show that rephrasing variance arises because prompts with different templates tend to activate different circuits in the model. This leads us to argue that it may be challenging to find a comprehensive circuit that explains and controls the model's behavior on a task, which can be expressed in countless templates, suggesting that LLMs may be inherently hard to steer. We show that sparsity, which has been claimed to form more compact and interpretable task circuits, fails to solve this problem. Regarding sample-wise variance, we argue that it is largely benign: extremely poor unfaithfulness scores often stem from how unfaithfulness is defined, rather than from defects in the measured circuits. We show that the magnitude of unfaithfulness is affected by *selective contribution scaling*, a neural mechanism that accounts for the extremely poor scores sometimes observed.

[1]EPFL, Switzerland. Correspondence to: Frank Zhengqing Wu <zhengqing.wu@epfl.ch>, Volkan Cevher <volkan.cevher@epfl.ch>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Circuit discovery aims to find the key pathways a model uses to perform a task. It usually does this by first evaluating the importance of all components, then selecting the most important ones to form the circuit that is considered responsible for carrying out the task. In recent years, this area has advanced along two main axes: greater efficiency and broader generality. On the efficiency side, methods are becoming more automated and easier to parallelize. To eliminate the need for extensive manual work, Conmy et al. (2023) introduced an automated framework that evaluates the importance of model components by applying causal interventions to each in turn. As such a method scales poorly, Marks et al. (2024); Syed et al. (2024); Nanda (2022) proposed gradient-based circuit discovery approaches that leverage the parallelism provided by modern deep learning hardware. In particular, edge attribution patching (EAP) (Syed et al., 2024; Nanda, 2022) can evaluate all the components of a model in two forward and one backward passes. However, this level of efficiency comes at the cost of precision. To balance these two aspects, Hanna et al. (2024) interpolated the model behavior into multiple steps and applied EAP to each step, yielding EAP-IG (integrated gradients).

On the generality side, earlier work (Nanda, 2022; nostalgebraist, 2020) often focused on single-sample probing. This methodology was unable to provide much insight into how models handle a task as a whole (Nanda, 2022). Subsequently, researchers transitioned to methods that assess the significance of model components over a task distribution with many samples, typically through computing the average importance across those samples (Hanna et al., 2024; Wang et al., 2023; Hanna et al., 2023; Marks et al., 2024). By steering the discovered task circuits, Wang et al. (2025a); Cheng et al. (2026) enhanced the model's performance on those tasks, highlighting the practical value of circuit discovery. Nevertheless, several works revealed great variances in the circuit discovery algorithms, calling their reliability into question.

In this work, we discuss three forms of such variance. The first is *resampling variance* (Méloux et al., 2025), which means that the identified circuit may differ when the probing prompts are resampled from the same underlying distribution. We show that such variance can be reduced by our new

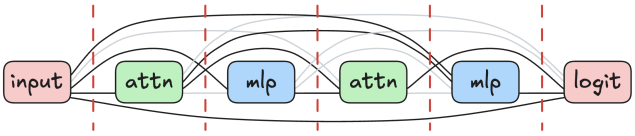


Figure 1. The computational graph of a mini Transformer. The residual addition is split into separate edges. The black edges are the ones included in the subgraph/circuit, while the grey ones are excluded. During a patched run, modules will take the results from previous ones through black edges but the grey edges will be fixed at activations corresponding to the corrupted input. Each red dotted line intercepts a set of edges that is necessary and sufficient for producing the output.

circuit discovery method, conductance-based EAP (CEAP), which achieves the same level of efficiency as EAP-IG.

The second type of variance is *rephrasing variance* (Méloux et al., 2025), where alternative phrasings of the same input can produce substantially different circuits. We show that this is because different templates activate distinct circuits in the model. A recent work by Franco et al. (2026) also observed template dependence for a different circuit-discovery algorithm. Together, our results suggest the ubiquity of this phenomenon. While Franco et al. (2026) proposed to shift the granularity of circuit discovery from tasks to templates, we take the stance that the dependence of circuits on templates reveals a profound shortcoming of current models and notions of tasks. Since it is impractical to exhaust all templates, the generality goal of circuit discovery might be unattainable: it is infeasible to find comprehensive task circuits, and one cannot reliably predict the outcome of task circuit steering when the model is faced with unseen templates. We investigate sparse training (Gao et al., 2025) as a potential remedy, as it tends to produce more compact task circuits, which might compel the model to rely on a single circuit for all templates. Nonetheless, we do not find strong evidence that sparsity adequately mitigates this issue.

The third form of variance is *sample-wise variance* (Miller et al., 2024), in which the circuit quality metric, unfaithfulness, varies sharply across samples. We show that the poor unfaithfulness scores arise from a negative correlation between the metric and the magnitude of the model's behavior, rather than from deficiencies in the discovered circuit itself. We account for this using a neural mechanism that we term selective contribution scaling.

Taken together, our contributions are the following:

- We introduce CEAP, whose underlying component selection strategy is more principled than that of EAP-IG. We support this claim by proving that conductance satisfies additive order preservation—an intuitive and desirable property that IG fails to meet. We further demonstrate that CEAP lowers resampling variance.
- We show that template-induced rephrasing variance is widespread and study its implications for circuit steering. We find that sparsity does not solve this problem, despite its prima facie appeal.
- We explain extreme sample-wise unfaithfulness through selective contribution scaling, showing that large unfaithfulness values reflect the metric geometry rather than catastrophic circuit failure.

2. Background: Evaluating Subgraphs With (Un)faithfulness

Circuit discovery aims to identify a subgraph (circuit) from the model's full computational graph that best reproduces the full model's behavior (illustrated in Figure 1). A widely used measure of the circuit quality is faithfulness (Hanna et al., 2024). In the current setup, the *behavior* of a task is usually characterized by how a task-specific output metric changes when the input changes from a base point (referred to as corrupted input in previous literature) to the clean input. For instance, when examining subject-verb agreement (SVA) (Newman et al., 2021), we might quantify how much output probability shifts from plural verbs to singular verbs as the input changes from one that cues a plural verb (corrupted input) to one that cues a singular verb (clean input). To quantify this change in general, for a given input x, we use the metric `prob diff`

$$M(x; x_0, x_1) \coloneqq \mathbb{P}(C(x_1)|x) - \mathbb{P}(W(x_0)|x),$$

where x_1 is a clean input, like *"The book on the table"*, and x_0 is a corrupted input like *"The books on the table"*. $C(x_1)$ is the set of tokens that x_1 encourages, and $W(x_0)$ is the set of competing tokens encouraged by x_0 instead. Continuing our example of SVA, $C(x_1)$ is singular verbs, and $W(x_0)$ is plural verbs. In the following, when the clean-corrupted pair is clear from context, we shorten $M(x; x_0, x_1)$ to $M(x)$.

Besides `prob diff`, another commonly used metric is `logit diff` (Nanda, 2022; Hanna et al., 2024), which, in this case, is the summation of all the logits corresponding to singular verbs minus those of the plural ones. Hanna et al. (2024); Conmy et al. (2023) also recommended using KL divergence as a metric. Nevertheless, we observed numerical issues with this approach, detailed in Appendix C.

We use subscripts G and G' to denote evaluation on the full computational graph and selected subgraph, respectively. The *behavior* of the full model and the selected circuit can be written as a subtraction of metrics,

$$Q_\Gamma(x_0, x_1) \coloneqq M_\Gamma(x_1) - M_\Gamma(x_0), \tag{1}$$

where $\Gamma \in \{G, G'\}$. The metrics for the full model in Equation (1) can each be obtained via two standard forward passes. By contrast, $M_{G'}(\cdot)$ is obtained by patching:

edges outside G' are fixed to their activations under x_0, while edges inside G' remain responsive to the current input through the selected subgraph (Hanna et al., 2024), as visualized in Figure 1. By this design, $M_{G'}(x_0) = M_G(x_0)$.

Finally, the faithfulness (Hanna et al., 2024) of G' with respect to G is defined as

$$\Phi(G', G; x_0, x_1) = \frac{Q_{G'}(x_0, x_1)}{Q_G(x_0, x_1)}.^{1}$$

The best value for $\Phi(G', G; x_0, x_1)$ is 1, as it is the faithfulness yielded by choosing the entire graph. For the ease of discussion, in what follows, we use *unfaithfulness*, defined as

$$U(G', G; x_0, x_1) \coloneqq |1 - \Phi(G', G; x_0, x_1)|,$$

to evaluate the subgraph. The smaller this metric, the better, which clarifies the presentation.

3. Conductance-based EAP (CEAP)

To find a subgraph that best explains the model's behavior on a certain task, we first quantify the importance of the edges in the model with a scoring method, after which we select the most important edges, which constitute the subgraph. Naturally, the quality of the subgraph hinges upon the soundness of the scoring method. In this section, we discuss how the conductance-based scoring method, which CEAP builds on, improves upon the previous IG-based scoring method.

Suppose we are studying a metric function $M(\cdot) \in C(\mathbb{R}^{d_{\text{in}}}, \mathbb{R})$, where d_{in} is the input dimension. The value of the metric given any input can be written as a function of the activations on the edges, $M(x) = M(\{a_e(x)\}_{e \in E})$. Here, $a_e(x) \in \mathbb{R}^{d_e}$ are the activations on edge e, and E denotes the set of all edges. We study the process where we change the input from x_0 to x_1. The importance we attribute to each component i of the activation a_e is its conductance (Dhamdhere et al., 2019):

Definition 3.1. The *conductance* of a scalar activation a_{ei} for a scalar-valued function $M(\cdot)$, evaluated with respect to the corrupted-input pair (x_0, x_1), is given by:

$$\begin{aligned} I_{\text{Cond}}(a_{ei}) &= \int_\gamma \frac{\partial M}{\partial a_{ei}} \mathrm{d}a_{ei} \\ &= \sum_{j \in [d_{\text{in}}]} (x_{1j} - x_{0j}) \int_0^1 \frac{\partial M(x_0 + \alpha(x_1 - x_0))}{\partial a_{ei}} \frac{\partial a_{ei}}{\partial x_j} \mathrm{d}\alpha, \end{aligned} \tag{2}$$

where γ is the path that a_{ei} traverses when the input moves from x_0 to x_1 in a straight line. Then, we define the importance/conductance of the edge e to be: $I_{\text{Cond}}(e) = \sum_{i \in [d_e]} I_{\text{Cond}}(a_{ei})$.

Comparison with previous methods. The current state-of-the-art patching method (EAP-IG (Hanna et al., 2024)) uses integrated gradients (IG) (Sundararajan et al., 2017), which scores each activation component with:

$$I_{\text{IG}}(a_{ei}) = [a_{ei}(x_1) - a_{ei}(x_0)] \int_0^1 \frac{\partial M(x_0 + \alpha(x_1 - x_0))}{\partial a_{ei}} \mathrm{d}\alpha. \tag{3}$$

Then, the IG method scores an edge with $I_{\text{IG}}(e) = \sum_{i \in [d_e]} I_{\text{IG}}(a_{ei})$.

When evaluating edges across multiple samples, the edge scores are computed as the average over all samples. After all edges have been scored, *circuit discovery* proceeds by retaining only those edges whose scores have the largest absolute values. In this paper, we use the greedy method (Hanna et al., 2024). This method first finds a circuit with a target edge number and prunes the childless or parentless edges.[2] The excluded edges are patched as described in the previous section. Different scoring methods, IG or conductance, result in different patching: EAP-IG and CEAP. Overall, to obtain the most informative graph, we need to give higher absolute-value scores to edges whose patching causes greater changes at the network output, which motivates us to study the following property.

Definition 3.2 (Additive order preservation)**.** Consider a function that can be decomposed additively into several branches: $\mathbb{R} \ni F(x; B) = \sum_{b \in B} f_b(y_b) = \sum_{b \in B} f_b(g_b(x))$, where B is the set of branches, and x and y_b may be vectors. We also denote $\mathbb{R} \ni F(x; B') = \sum_{b \in B'} f_b(y_b)$ for any $B' \subseteq B$. Suppose we study the process where x moves from x_0 to x_1. For convenience, we denote $\Delta F(B') = F(x_1; B') - F(x_0; B')$ and $\Delta f_b = f_b(x_1) - f_b(x_0)$. For a scoring function that assigns a scalar to an edge, $I(y_b) \in \mathbb{R}$, we say it satisfies *additive order preservation* if for any $b \in B$, with

$$\mathcal{E}_b \coloneqq |\Delta F(B) - \Delta F(B \setminus \{b\})| = |\Delta f_b|,$$

we have

$$\mathcal{E}_{b_1} > \mathcal{E}_{b_2} \iff |I(y_{b_1})| > |I(y_{b_2})|.$$

Remark 3.3. Note that $|\Delta F(B) - \Delta F(B \setminus \{b\})|$ exactly equals to the importance of "activation" y_b measured with activation patching (Nanda, 2022), which freezes y_b to be $y_b(x_0)$ while letting other activations freely move with upstream inputs. Additive order preservation conveys that, if the patching of one branch causes the whole function to

[1] This value was previously called normalized faithfulness in previous literature (Hanna et al., 2024; Zhang et al., 2025). As the "unnormalized" faithfulness is not the emphasis of this paper, we will refer to the normalized faithfulness as faithfulness for brevity.

[2] The target edge number and the final edge number are usually close. We do not distinguish between them in our discussion.

change its behavior more than the patching of another, then the former branch should be assigned a score higher in its absolute value than that of the latter, so that the former can be prioritized to be admitted into the subgraph. This property is particularly desirable when we study unfaithfulness, which evaluates a subgraph by how much the behavior is affected when the complement of the subgraph is patched.

Remark 3.4. The sum-of-branches type of functions discussed in Definition 3.2 can be of significant relevance if we wish to interpret the residual stream of currently prevalent architectures, as the residual stream is always a summation of previous branches. More broadly, additive order preservation is a minimal desideratum for scoring when applied to general functions. General functions may have other ways to mix the outputs of branches. The appropriate desiderata for more complicated mixings may not be immediately intuitive. However, the first-order Taylor expansion of any mixing always appears in the form of sum-of-branches functions, so studying the latter still provides insight.

To show that conductance is a more principled scoring method, we prove:

Theorem 3.5. *Conductance satisfies additive order preservation, while IG does not.*

The proof is in Appendix E. In the following, we derive the discretized version of conductance and IG, which shows how the two quantities are computed numerically. The discretized computation also sheds light on why IG appears less principled.

Let us break down the input movement $x_0 \to x_1$ into K evenly spaced segments and denote the endpoints of each segment by $x^k := x_0 + k/K(x_1 - x_0)$, where $k \in \{0, 1, \cdots, K\}$. We also denote the activations corresponding to these input endpoints by $a_{ei}^k := a_{ei}(x^k)$. The discretized IG can be written as

$$I_{\text{IG}}^{\text{dis}}(a_{ei}) = \frac{a_{ei}^K - a_{ei}^0}{K} \sum_{k=0}^{K-1} \frac{\partial M(x^k)}{\partial a_{ei}}. \tag{4}$$

One can interpret this formula as a refined version of `gradient×input` (Nanda, 2022). The latter scores the edge with $\left(a_{ei}^K - a_{ei}^0\right) \frac{M(x^0)}{a_{ei}}$. Equation (4), in comparison, computes the average gradient instead of using one gradient at one endpoint. Another more fine-grained interpretation of Equation (4) is that it is a weighted sum of the gradients $\frac{\partial M(x^k)}{\partial a_{ei}}$, with all weights equal to $\frac{a_{ei}^K - a_{ei}^0}{K}$. Nevertheless, it may be undesirable for all the weights to be the same. After all, the rate of change of M with respect to e_{ei} is roughly $\frac{\partial M(x^k)}{\partial a_{ei}}$ only when a_{ei} lies within the segment of $\left(a_{ei}^k, a_{ei}^{k+1}\right)$, whose length is not necessarily $\frac{a_{ei}^K - a_{ei}^0}{K}$ due to the nonlinearity of $a_{ei}(x)$. This motivates a new formula that rectifies the weightings of $\frac{\partial M(x^k)}{\partial a_{ei}}$:

$$I_{\text{Cond}}^{\text{dis}}(a_{ei}) = \sum_{k=0}^{K-1} \frac{\partial M(x^k)}{\partial a_{ei}} \left(a_{ei}^{k+1} - a_{ei}^k\right), \tag{5}$$

which is exactly the discrete form of conductance (Shrikumar et al., 2018). The "weighted-sum of gradients" perspective of Equations (4) and (5) also suggests that IG can be taken as an approximation to conductance, which explains its efficacy shown by Hanna et al. (2024).

3.1. CEAP Reduces Circuit Variance Under Data Resampling

Méloux et al. (2025) noted that EAP-IG finds significantly different circuits when we resample the dataset from the same distribution. We will show that such an instability is ameliorated by CEAP. Following the setup of Hanna et al. (2024), we conducted experiments on GPT-2 small, GPT-2 XL, Pythia-160M, and Pythia-2.8B over three datasets: SVA (Newman et al., 2021), IOI (Wang et al., 2023), and greater-than (Hanna et al., 2023).[3] The templates for generating these datasets are shown in Appendix D. Experimental details are in Appendix A.

In our experiments, for each task, we used a full dataset of roughly 10000 samples and subsampled 4 smaller datasets (without replacement), each with 1000 samples. We performed circuit discovery for the 4 datasets and measured the stability of the found circuits using pairwise Jaccard index (PJI). More concretely, suppose we have two graphs G_1 and G_2, with edge sets E_{G_1} and E_{G_2}, respectively. The PJI for these two graphs is defined as $|E_{G_1} \cap E_{G_2}| \,/\, |E_{G_1} \cup E_{G_2}|$. With 4 sub-datasets, for the same target edge number, we obtained 4 graphs and thus 6 PJIs. We report the mean and standard deviation of these PJIs in Figure 2 for GPT-2 XL on SVA. The x-axis shows the number of edges we admitted during circuit discovery. We separate experiments for different templates, as we will show in Section 4 that different templates activate distinct circuits in the model and thus their respective circuit discovery should be done separately. Due to space constraints, we only show the first 6 templates here. The results for other models, datasets, and templates are shown in Appendix F. *Overall, we find that CEAP achieves PJI values that are higher than, or at least on par with, those of EAP-IG.* We believe this is because CEAP can find important edges in a more principled manner as shown in Theorem 3.5, while EAP-IG can only be viewed as a noisy version of CEAP.

Note that CEAP attains unfaithfulness comparable to EAP-IG, so its increased stability in circuit selection does not arise from choosing task-irrelevant circuits. Further details

[3] Hanna et al. (2024) also used three other datasets. They have relatively few samples and are unsuitable for our purpose.

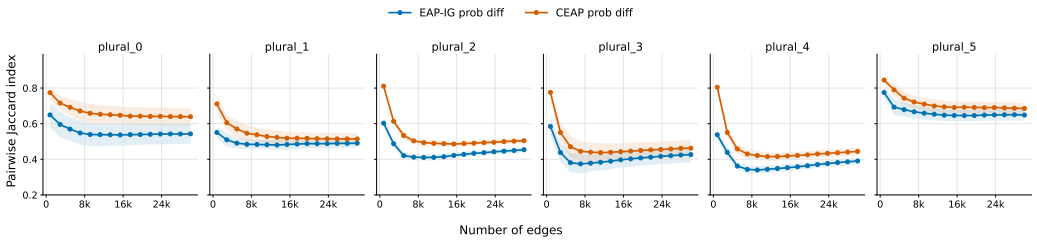


Figure 2. Comparison of PJI yielded by CEAP and EAP-IG (using GPT-2 XL for a subset of SVA)

are given in Appendix F.

4. Template-induced Circuit Variance

Méloux et al. (2025) observed that performing circuit discovery with paraphrased clean-corrupted prompt pairs can yield dramatically different circuits, even when a human judges the paraphrased prompts as representing the same underlying task. We demonstrate that this occurs because the templates influence the circuits the models use to perform the same task. To show this, we take 1000 samples from the same task but in different templates, apply CEAP to evaluate edge importance, and identify the critical circuit for each sample. Then, we illustrate the similarity of the scores across different samples, as well as the similarity of the graphs identified for these samples.

For the similarity of the scores, we investigate the ranking of edges based on the absolute values of their scores. More precisely, consider all edges in the model arranged in a single vector $\left(e_1, e_2, \cdots, e_{|E|}\right)$. Their associated scores for a given sample are given by $\left(s_1, s_2, \cdots, s_{|E|}\right)$, where $s_i = I_{\text{Cond}}(e_i)$. We then take the absolute values of these scores,[4] $\left(|s_1|, |s_2|, \cdots, |s_{|E|}|\right)$, and sort them in descending order to obtain the absolute-score rank vector $\left(r_1, r_2, \cdots, r_{|E|}\right)$, where r_i denotes the rank of $|s_i|$. We then applied UMAP to the absolute-score rank vectors for all 1000 samples to project them onto 2 dimensions. The resulting visualization for GPT-2 small on SVA is shown in Figure 3a. Additionally, we directly visualize the similarity between the graphs obtained for different samples. We select the graph size for which the mean unfaithfulness across all 1000 samples falls below 0.2, and then compute the PJI among all graphs found for each sample at that size; these results are shown in Figure 3b. We produced the same visualization for GPT-2 small and Pythia-160M on SVA, IOI, and greater-than. Across all figures, it is evident that the model relies on different circuits to process different templates. The complete set of visualizations is provided in Appendix G.

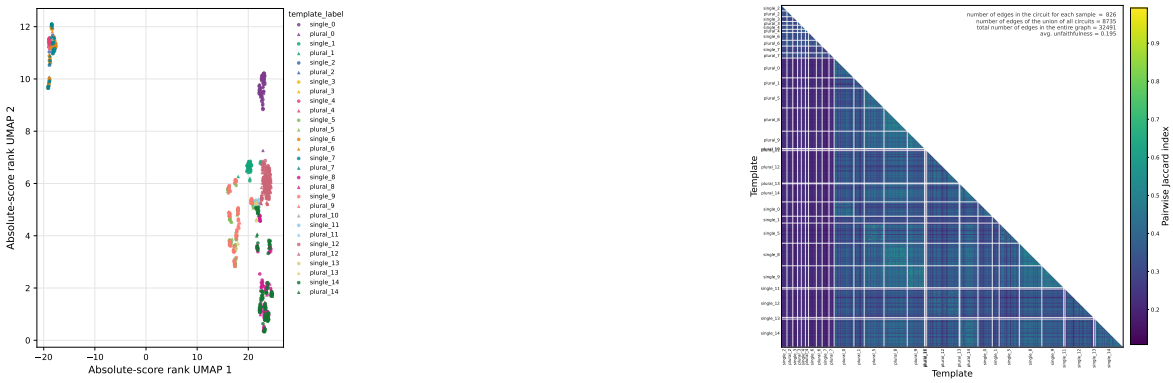
Numerous prior studies (Hanna et al., 2024; Méloux et al., 2025; Wang et al., 2025b; Miller et al., 2024; Nanda, 2022; Gao et al., 2025) have operated under the assumption that a single circuit within a given model is responsible for performing a specific task, and that, by evaluating edge importance via averaging over a sufficient number of samples, usually generated with multiple templates (Hanna et al., 2024; Méloux et al., 2025), we can comprehensively pinpoint the circuit responsible for performing the task. However, we have shown that such an assumption is flawed. Different templates, in fact, trigger different circuits within the models. Since the space of possible templates is effectively unbounded, it follows that *identifying a comprehensive circuit for a particular task may be an unattainable goal*.

This implication challenges one of the central aspirations of mechanistic interpretability: that, by adjusting the identified task circuits, we can reliably steer the model's behavior on that task in our preferred direction (Zhang et al., 2026), including boosting math capability (Wang et al., 2025a), moderating refusal (Cheng et al., 2026), editing knowledge (Meng et al., 2022; 2023; Yao et al., 2024), and many more (Gao et al., 2025; Lindsey et al., 2025; Wang et al., 2023). Although previous works showed that interventions on circuits identified from available samples can be effective (Gao et al., 2025; Wang et al., 2023) and may improve benchmark performance (Wang et al., 2025a; Cheng et al., 2026), our observations here suggest that such a methodology does not guarantee reliable behavior in the wild, where the model may encounter unseen templates.[5] In practice, the degree of concentration among template circuits can serve as a proxy for our confidence in the effect of a model intervention.

To deepen the discussion, we ask whether it is achievable or meaningful to pursue concentrated task circuits independent of templates, since "tasks" may seem an artificial concept, while templates are the concrete data the models process. It may be quite demanding to require models to develop task-focused circuits rather than template-focused ones. Nevertheless, a task can be thought of informally as a collection of prompts that a human is able to address using

[4] This follows the convention of Hanna et al. (2024), who performed circuit discovery using the absolute score values.

[5] In Appendix H, we present a case that, after encouraging plural verbs for a sample by patching a circuit obtained for this purpose from another sample, the model actually encourages singular verbs.



(a) Absolute-score rank UMAP.

(b) Pairwise Jaccard index.

Figure 3. Template-dependent sample circuits for GPT-2 small on SVA. The UMAP embeds samples by their absolute-score rank vectors, and the Pairwise Jaccard index matrix captures the overlap between the corresponding circuits. In the right panel, two clearly separated template groups emerge. Circuits within each group exhibit substantial overlap, while overlap between the two groups is almost zero. These two groups correspond to the two clusters in the left panel: one in the upper-left corner and the other containing the rest. The statistics in the upper-right of the right panel offer quantitative insight into how little overlap there is among the circuits for different templates.

a single, unified algorithm,[6] which indicates that it should be feasible to implement it using just one circuit. The fact that current models resort to different circuits for different templates implies that the models leverage redundant algorithms.[7] A well-known principle in AI holds that intelligence can be understood as a form of compression (Wolff, 2013; Schmidhuber, 2008). In particular, from the perspective of Kolmogorov complexity (Hutter, 2000), intelligence can be viewed as the ability to find short descriptions of data that enable accurate prediction. In this sense, models that rely on redundant algorithms to perform tasks have suboptimal intelligence, and the aim of training models that rely on a template-independent circuit for a given task aligns with the broader objective of discovering more compressive, thus intelligent, models.

A recent effort by Gao et al. (2025) to train more interpretable transformers showed that imposing sparsity during training shrinks the task circuits. Prima facie, this seems like a potential solution to mitigate the unsteerability of models that we discussed above. *Could it be that sparsity forces the model to merge the circuits for different templates, thereby causing the task circuits to shrink?* Our experiments, based on the models from Gao et al. (2025), suggest that the benefit from sparsity is limited.

In our experiments, we adapted `transformer-lens` (Nanda & Bloom, 2022), the interpretability infrastructure that we used for CEAP, to the specialized architectures of the sparse models built by Gao et al. (2025), which were customized for coding. We also created two datasets suitable for these models, single-double-quote and else-elif, summarized in Appendix D. Across these experiments, we do not find that sparsity causes the model to merge template circuits: sparse models still deploy different circuits for different templates. Full visual comparisons and summary statistics are presented in Appendices I and J.

In addition, we analyze how the mean PJI between circuit pairs obtained from different templates varies with increasing sparsity, while keeping all other settings fixed. As shown in Figure 4, the overlap of different template circuits increases substantially when we allow fewer nonzero parameters. Nevertheless, the average PJI for circuit pairs across different tasks (cross-task) also increases to an extent comparable to the mean cross-template PJI for single-double-quote. This suggests that the level of polysemanticity increases with sparsity, which leads to the possibility that the increasing overlap between cross-template circuits for one task might be a result of the model using a larger portion of polysemantic neurons to implement redundant algorithms, rather than truly compressing for a single, unified algorithm for the task. This complicates task steering. Determining whether sparsity primarily enhances polysemanticity or compression requires

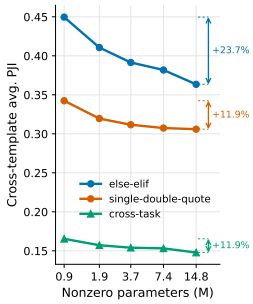


Figure 4. Cross-template average PJI increases with more sparsity, but so does cross-task average PJI.

[6]For example, for IOI the algorithm is: find all names, remove duplicates, and output the rest (Wang et al., 2023).

[7]A redundant algorithm may look like: `if handling template_A: use algorithm_A, elif handling template_B: use algorithm_B...` A good example of this for IOI is shown in Figure 3 of Franco et al. (2026).

detailed model-level investigation, which lies beyond the scope of this variance-focused paper. We leave such an analysis for future work. Our current observations do not indicate that sparsity resolves the issue discussed in this section.

5. Variance of Unfaithfulness Across Samples

Miller et al. (2024) noted that averaging edge scores over samples to identify circuits can yield large variance in per-sample unfaithfulness, even if the resulting circuit is faithful at the population level.[8] One might think this is because the averaged scores better match some samples while fitting others less well. However, it turns out that this is not the primary source of variance. In our experiments, we observed that even if we perform circuit discovery with CEAP using a fixed circuit size *for each sample separately*, thereby eliminating influence from other samples, the resulting unfaithfulness still fluctuates greatly across samples. Figure 6a shows such an example, which we use throughout this section. We see that the unfaithfulness U of a sample (sample 78) may go as high as 20, while most of the samples from the same template (`plural_1`) already achieve low unfaithfulness.

In this section, we show that under a reasonable circuit discovery scheme, poor unfaithfulness scores typically stem from how unfaithfulness interacts with the distribution of edge scores, rather than from missing important edges. Comparing unfaithfulness values between samples does not reveal which sample's circuit is more sufficient for recovering the full model's behavior.

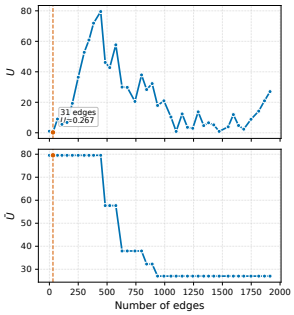


Figure 5. Normal and pessimistic unfaithfulness for sample 78 of SVA.

It is important to recognize that unfaithfulness measured on an individual sample is often hard to interpret, because it can vary in a highly non-monotonic way as the circuit size increases. Figure 5 shows how U changes with the circuit size. It reaches 0.267 with just 31 edges but shoots up to around 80 when we increase the circuit size. Ideally, however, unfaithfulness should consistently decrease when we admit more edges into the circuit. Single-sample unfaithfulness fails to meet this because it evaluates the behavior of the model at a single point in the input space. We detail this in Appendix L. Highly fluctuating unfaithfulness can introduce noise to our analysis and lower statistical significance. To reduce this noise, we define pessimistic unfaithfulness $\bar{U}$:

Definition 5.1. Let $U(m)$ denote the unfaithfulness of the circuit found by the algorithm with target edge number m. Pessimistic unfaithfulness is defined as $\bar{U}(m) = \max_{m \le \hat{m} \le |E|} U(\hat{m})$.

In experiments, since we only evaluate a subset of all possible target edge counts, we estimate $\bar{U}(m)$ by taking the maximum unfaithfulness observed over the evaluated counts.

Next, we show that extremely poor U should not be interpreted as a dramatic discrepancy between the circuit and the model's behavior. We define the unnormalized unfaithfulness $U' := |Q_{G'}(x_0, x_1) - Q_G(x_0, x_1)| = |M_{G'}(x_1) - M_G(x_1)|$ to directly measure the behavior gap between the circuit and the full model. We also define the pessimistic unnormalized unfaithfulness $\bar{U}'$ by enforcing monotonicity. In our experiments, we found that extremely high $\bar{U}$ is usually associated with near-zero $|Q_G|$,[9] as shown in Figure 6b, where we obtained a negative ρ for Spearman rank correlation (SRC). Since Q_G is the normalization for computing (un)faithfulness, it is natural to suspect whether the extremely small normalization values are what cause $\bar{U}$ to blow up for certain samples. This suspicion is corroborated by a positive SRC coefficient ρ between the unnormalized $\bar{U}'$ and $|Q_G|$, illustrated in Figure 6c. *These observations suggest that the extremely large $\bar{U}$ does not mean the logits produced by the chosen circuit differ wildly from those of the full model; rather, it is merely a consequence of normalization.*

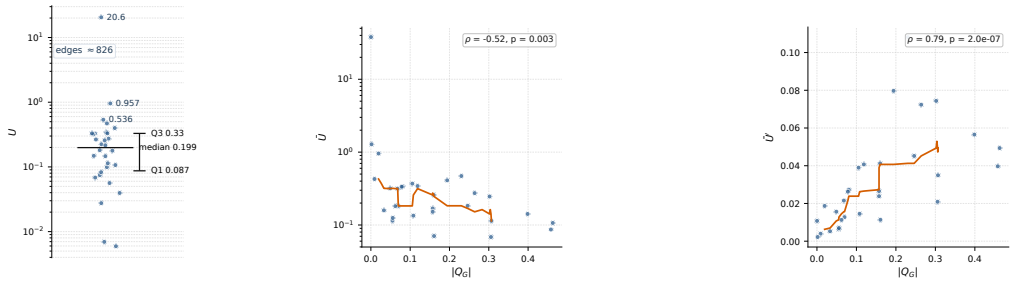
5.1. The Correlation Between $\bar{U}$ and $|Q_G|$

What remains unclear is why, given the same number of allowed edges, the circuits identified for samples with small $|Q_G|$ achieve much worse $\bar{U}$ than the circuits identified for samples with large $|Q_G|$. Intuitively, this may be because as $|Q_G|$ decreases, the relative significance of the edges left out of the circuit, compared with those that are included, becomes greater. The following analysis supports this explanation.

Given a model with each edge scored on sample n, we arrange the absolute-valued scores of all its edges into a nonincreasing vector s^n. Let $S^n := \sum_{\ell=1}^{|E|} s^n_\ell$, and $u^n = s^n / S^n$ so that all u^n components sum up to 1. We define the normalized momentum to be $\mu^n := \sum_{\ell=1}^{|E|} u^n_\ell \times \ell$, which quantifies how heavy-tailed s^n is. From here on, for clarity, we introduce a superscript of sample index n for Q_G. We found that the samples with lower $|Q^n_G|$ have larger

[8] The notion of population-level faithfulness has varied in previous works (Miller et al., 2024; Hanna et al., 2024; Wang et al., 2023), which we discuss in Appendix K.

[9] In very rare cases, we find negative Q_G, meaning the full model is more likely to output the wrong answer.



(a) U distribution. *(b)* $\bar{U}$ vs. $|Q_G|$. *(c)* $\bar{U}'$ vs. $|Q_G|$.

Figure 6. For GPT-2 small on SVA template `plural_1` with target edge number 826. Here, we perform circuit discovery for each sample separately. The corresponding U distribution for scores obtained via averaging within the template is in Figure 52. Samples with smaller $|Q_G|$ tend to show larger pessimistic unfaithfulness $\bar{U}$, while the pessimistic unnormalized unfaithfulness $\bar{U}'$ tends to increase with $|Q_G|$. Orange curves show rolling-median trends as visual guides only. Boxes report Spearman rank correlation ρ and its p-value.

μ^n (Figure 7a). Suppose we consider two samples such that $|Q_G^{n_1}| > |Q_G^{n_2}|$. Then u^{n_1} and u^{n_2} would appear as illustrated in the lower panel of Figure 8. Given an edge number m, our circuit discovery algorithm approximately selects the edges with the highest absolute scores to form the circuit. Define $C^n(m) \coloneqq \sum_{\ell \in L} s_\ell^n$ where L is the set of indices of s^n that correspond to the edges admitted into the circuit. Then, $C^n(m)/S^n \coloneqq R^n(m)$ can be interpreted as the ratio of score mass included in the circuit. Naturally, μ^n and $R^n(m)$ should have a negative correlation (Figure 7b). Moreover, as $R^n(m)$ carries the meaning of the aggregate importance of edges included in the circuit *relative* to total importance of all edges, it should have a negative correlation with unfaithfulness $\bar{U}$, which reflects the change of logit behavior caused by including only the edges in the chosen circuit *relative* to the original logit behavior caused by the full model (Figure 7c). Based on these correlations, we derive the following relationships that account for the trend observed in Figure 6b.

$$|Q_G^n| \searrow \quad \longrightarrow \quad \mu^n \nearrow \quad \longrightarrow \quad R^n(m) \searrow \quad \longrightarrow \quad \bar{U} \nearrow \tag{6}$$

While the last two arrows in the above appear intuitive, the mechanism underlying the first one is unclear. We show that this is due to a neural mechanism that we term *selective contribution scaling*, which we cover in Section 5.2.

We verify all the correlation trends in this section, including the two in Figure 6 and the four in Figure 7, for all templates of SVA/IOI/greater-than, for both GPT-2 small and Pythia-160M, and for all graph sizes that we swept. The trends hold up well. Moreover, we check these trends using U in place of $\bar{U}$, in case our design of $\bar{U}$ introduces undesirable artifacts. U-based plots generally show the same trends, despite being noisier. Note that the discussion of this section so far is based on scoring samples separately. We also produce all the visualizations for scores obtained by averaging samples within their own templates, which is closer to common practice (Hanna et al., 2024; Méloux et al., 2025; Miller et al., 2024). Those experiments are well captured by the conclusion drawn in this section, as the circuits found within a template are highly similar in most cases. All the results are presented in Appendix M.

5.2. A Mental Image for Selective Contribution Scaling

Our conductance score satisfies that, if a set of edges is necessary and sufficient to produce the output (Figure 1), their scores must sum up to Q_G^n (known as partition consistency (Dhamdhere et al., 2019)). In this sense, one can interpret the score as the contribution of the edge to the output. Suppose we have a pair of samples with $|Q_G^{n_1}| > |Q_G^{n_2}|$. It is reasonable to expect that the entries in s^{n_1} are larger than those in s^{n_2}, given partition consistency. Since we know in hindsight that u^{n_2} is more heavy-tailed than u^{n_1} (Figure 7a), we can infer that s^{n_1} is considerably higher than s^{n_2} at low indices, while the differences between them at high indices are mild. Namely, if we move the input from sample n_2 to n_1 to increase the output signal amplitude from $|Q_G^{n_2}|$ to $|Q_G^{n_1}|$, the important edges located at low indices will contribute more than the less important ones at high indices.[10] This is what we call selective contribution scaling, as illustrated in the upper

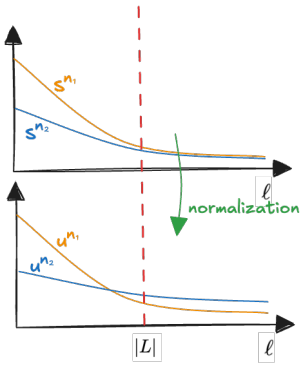
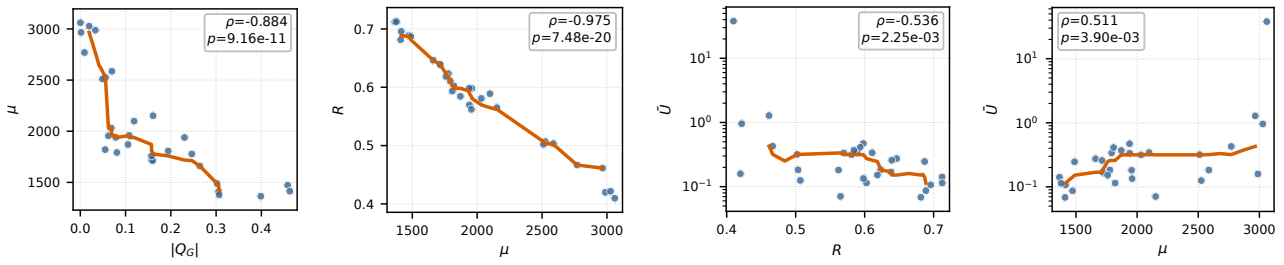


Figure 8. Mental image of selective contribution scaling.

[10]Note that we are implicitly assuming that the edge orderings in s^{n_1} and s^{n_2} coincide. While this is not strictly true, they should be very similar, because, as we show in Section 4, samples sharing the same template tend to exhibit similar scoring patterns.



(a) μ vs. $|Q_G|$. *(b)* R vs. μ. *(c)* $\bar{U}$ vs. R. *(d)* $\bar{U}$ vs. μ.

Figure 7. Spearman rank correlations that explain Figure 6b. The setup is exactly the same as in Figure 6. The last panel, though not discussed, adds supporting evidence: it suggests that the heavy-tailedness of s^n is indicative of the unfaithfulness $\bar{U}$, combining the last two correlations in Equation (6).

panel of Figure 8. We verify the mental image of Figure 8 with a concrete sample pair in Appendix N.

That there exist specialized parts of the model reacting to structurally specific stimuli (tasks expressed in templates) and other parts being indifferent to them aligns with the notion of sparse coding, a widely endorsed framework of information processing in neuroscience (Barlow, 1961; Olshausen & Field, 1996; 1997; Vinje & Gallant, 2000; Lonnqvist et al., 2023). Nonetheless, in our case, it links unfaithfulness to the insignificant magnitude of Q_G, which weakens unfaithfulness as a definitive marker of performance recovery. By the same token, we also anticipate great cross-sample variance in faithfulness-style normalization-dependent metrics measured on systems that exhibit sparse coding.

6. Conclusion and Discussion

In this paper, we systematically investigate the three types of variance in circuit discovery algorithms. While resampling variance can be mitigated with more principled scoring, and sample-wise variance of unfaithfulness is largely benign, rephrasing variance is hard to diminish and suggests the impossibility of finding comprehensive task circuits.

In discussing the above, we develop methods and guidance that can inform future practice. First, conductance is a more principled way of scoring compared to IG, as it satisfies additive order preservation. Second, although evaluating (un)faithfulness is relatively straightforward, and the notion itself is intuitively plausible (Hanna et al., 2024), there is a caveat: comparisons of this metric across different samples are subject to an intrinsic sample-wise variance induced by selective contribution scaling. We further discuss in Appendix K that population-level (un)faithfulness should explicitly account for this effect and be more lenient toward samples whose behavioral magnitude is small. Nonetheless, (un)faithfulness remains informative for tracking how circuit quality improves as we increase the permitted circuit size. In this regard, pessimism serves as a useful complement to reduce interpretational noise.

Furthermore, our discussion on template-based rephrasing variance sheds new light on the generalizability of *task* circuits (Sharkey et al., 2025). On the one hand, it suggests that statements involving task circuits should be carefully tested for robustness under changes in templates,[11] and that template-agnostic benchmarking can be inconclusive. On the other hand, it opens up avenues for future exploration. Let us define a task as the collection of samples that engage the same circuit. This definition is useful because it is operational and may provide insight into circuit steering. At present, despite certain post hoc heuristics (Franco et al., 2026), we lack a principled way to determine which templates engage the same circuit and which rely on distinct circuits. As a result, the notion of a task remains elusive. To address this, we must systematically characterize how templates influence the algorithmic procedures that models use to process them. Put differently, it is important to formalize a model-centered notion of a task, in contrast to the current notion, in which humans judge whether different instances should be considered part of the same task (Hanna et al., 2024; Wang et al., 2023; Hanna et al., 2023; Newman et al., 2021). Furthermore, as the circuits currently appear to be template-dependent, an important question is whether models can be trained to use the same circuit for tasks defined in human terms, which may, in turn, lead to greater compression and potentially higher intelligence.

Acknowledgements

We thank Dr. Paul Rolland for valuable discussions. This work was funded by the Swiss National Science Foundation (SNSF) under grant number 2000-1-240094. This work was also supported under project ID # 37 as part of the Swiss

[11] Some previous studies have considered robustness to paraphrasing (Meng et al., 2022; 2023). However, it has not yet become a standard practice.

AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Barlow, H. Possible principles underlying the transformations of sensory messages. *Sensory Communication*, 1, 01 1961. doi: 10.7551/mitpress/9780262518420.003.0013.

Chatterji, N. S., Long, P. M., and Bartlett, P. L. When does gradient descent with logistic loss find interpolating two-layer networks?, 2021. URL https://arxiv.org/abs/2012.02409.

Cheng, S., Wiegreffe, S., and Manocha, D. What drives representation steering? a mechanistic case study on steering refusal. *arXiv preprint arXiv:2604.08524*, 2026.

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

Dhamdhere, K., Sundararajan, M., and Yan, Q. How important is a neuron. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SylKoo0cKm.

Franco, G., Tassis, L. M., Rohr, A., and Crovella, M. Finding highly interpretable prompt-specific circuits in language models, 2026. URL https://arxiv.org/abs/2602.13483.

Gao, L., Rajaram, A., Coxon, J., Govande, S. V., Baker, B., and Mossing, D. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.

Hanna, M., Liu, O., and Variengien, A. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=p4PckNQR8k.

Hanna, M., Pezzelle, S., and Belinkov, Y. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL https://openreview.net/forum?id=grXgesr5dT.

Hutter, M. A theory of universal artificial intelligence based on algorithmic complexity, 2000. URL https://arxiv.org/abs/cs/0004001.

Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Lonnqvist, B., Wu, Z., and Herzog, M. H. Latent noise segmentation: How neural noise leads to the emergence of segmentation and grouping. *arXiv preprint arXiv:2309.16515*, 2023.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Marr, D. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. The MIT Press, 07 2010. ISBN 9780262514620. doi: 10.7551/mitpress/9780262514620.001.0001. URL https://doi.org/10.7551/mitpress/9780262514620.001.0001.

Méloux, M., Portet, F., and Peyrard, M. Mechanistic interpretability as statistical estimation: A variance analysis of eap-ig. *arXiv preprint arXiv:2510.00845*, 2025.

Meng, K., Bau, D., Andonian, A. J., and Belinkov, Y. Locating and editing factual associations in GPT. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=-h6WAS6eE4.

Meng, K., Sharma, A. S., Andonian, A. J., Belinkov, Y., and Bau, D. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=MkbcAHIYgyS.

Miller, J., Chughtai, B., and Saunders, W. Transformer circuit evaluation metrics are not robust. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=zSf8PJyQb2.

Nanda, N. Attribution patching: Activation patching at industrial scale. urlhttps://www.neelnanda.io/mechanistic-interpretability/attribution-patching, February 2022. Last accessed July 25, 2025.

Nanda, N. and Bloom, J. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens, 2022.

Newman, B., Ang, K.-S., Gong, J., and Hewitt, J. Refining targeted syntactic evaluation of language models. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3710–3723, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.290. URL https://aclanthology.org/2021.naacl-main.290/.

nostalgebraist. Interpreting gpt: The logit lens. https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens, 2020. LessWrong blog post, accessed 2026-05-07.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996. URL https://api.semanticscholar.org/CorpusID:4358477.

Olshausen, B. A. and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. ISSN 0042-6989. doi: https://doi.org/10.1016/S0042-6989(97)00169-7. URL https://www.sciencedirect.com/science/article/pii/S0042698997001697.

Schmidhuber, J. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. In *Workshop on anticipatory behavior in adaptive learning systems*, pp. 48–76. Springer, 2008.

Sharkey, L., Chughtai, B., Batson, J., Lindsey, J., Wu, J., Bushnaq, L., Goldowsky-Dill, N., Heimersheim, S., Ortega, A., Bloom, J. I., Biderman, S., Garriga-Alonso, A., Conmy, A., Nanda, N., Rumbelow, J. M., Wattenberg, M., Schoots, N., Miller, J., Saunders, W., Michaud, E. J., Casper, S., Tegmark, M., Bau, D., Todd, E., Geiger, A., Geva, M., Hoogland, J., Murfet, D., and McGrath, T. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=91H76m9Z94. Survey Certification.

Shrikumar, A., Su, J., and Kundaje, A. Computationally efficient measures of internal neuron importance, 2018. URL https://arxiv.org/abs/1807.09946.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3319–3328. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

Syed, A., Rager, C., and Conmy, A. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416, 2024.

Vinje, W. and Gallant, J. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science (New York, N.Y.)*, 287:1273–6, 03 2000. doi: 10.1126/science.287.5456.1273.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Wang, X., Hu, Y., Du, W., Cheng, R., Wang, B., and Zou, D. Towards understanding fine-tuning mechanisms of LLMs via circuit analysis. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025a. URL https://openreview.net/forum?id=Z9qzta1yiK.

Wang, Y., Liu, Y., Shi, Y., Li, C., Pang, A., Yang, S., Yu, J., and Ren, K. Discovering influential neuron path in vision transformers. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=WQQyJbr5Lh.

Weiß, C. H., Zhu, F., and Hoshiyar, A. Softplus ingarch models. *Statistica Sinica*, 32(2):pp. 1099–1120, 2022. ISSN 10170405, 19968507. URL https://www.jstor.org/stable/27118810.

Wolff, J. G. Computing as compression: the sp theory of intelligence, 2013. URL https://arxiv.org/abs/1303.2013.

Yao, Y., Zhang, N., Xi, Z., Wang, M., Xu, Z., Deng, S., and Chen, H. Knowledge circuits in pretrained transformers. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 118571–118602. Curran Associates, Inc., 2024. doi: 10.52202/079017-3765. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d6df31b1be98e04be48af8bedb95b499-Paper-Conference.pdf.

Zhang, H., Zhang, Z., Wang, M., Su, Z., Wang, Y., Wang, Q., Yuan, S., Nie, E., Duan, X., Han, F., Xue, Q., Yu, Z., Shang, C., Liang, X., Xiong, J., Shen, H., Tao, C., Liu, Z., Jin, S., Xi, Z., Zhang, D., Ananiadou, S., Gui, T., Xie, R., So, H. K.-H., Schütze, H., Huang, X., Zhang, Q., and Wong, N. Locate, steer, and improve: A practical survey of actionable mechanistic interpretability in large language models, 2026. URL https://arxiv.org/abs/2601.14004.

Zhang, L., Dong, W., Zhang, Z., Yang, S., Hu, L., Liu, N., Zhou, P., and Wang, D. EAP-GP: Mitigating saturation effect in gradient-based automated circuit identification. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=lGyXq0LOeQ.

A. Experimental Details

Following previous conventions (Hanna et al., 2024; Nanda, 2022), we used `logit_diff` for tasks whose correct and incorrect answers are single logits (IOI, single-double-quote). For other tasks, we used `prob_diff`.

For GPT-2 and Pythia experiments, we used integration step $K = 200$ for both EAP-IG and CEAP. For sparse-transformer experiments (Gao et al., 2025), we used 150.

B. Code and Data Availability

A ready-to-use implementation of CEAP, along with code to reproduce the figures presented in this paper, is available at https://github.com/LIONS-EPFL/Circuit-Variance.

C. Numerical Issue Caused by KL Divergence

Hanna et al. (2024); Conmy et al. (2023) also recommended using the KL divergence as a metric, as it is in principle applicable to all tasks. In practice, however, we found KL divergence to be restrictive: it cannot be used for models that assign zero probability to some tokens, since this leads to an infinite KL value, which we observe *e.g.*, for Pythia-160M. While $\mathrm{Softmax}(\cdot)$ should theoretically ensure that every token has nonzero probability, numerical underflow (which occurs for both `fp32` and `fp64`) frequently produces exact zeros in the probabilities. To keep the exposition uncluttered, we therefore report only the `logit`/`prob diff` metrics in the main paper. The experiments that we carried out for KL divergence do not show qualitatively different results from the conclusions we present here.

D. Dataset Templates

We use template-generated datasets so that clean and corrupted prompts differ in a controlled way. Tables 1 to 5 summarize the templates used in our experiments. Placeholders in square brackets are filled with names, nouns, variable names, or code literals during dataset construction. For the tasks customized for sparse-transformers (Gao et al., 2025), each base template generates a paired contrast. For `single_double_quote`, we show both paired directions explicitly using the `_s` and `_d` suffixes. For `else_elif`, we show one representative direction in the table and omit the `_else`/`_elif` suffixes from the displayed template names for space; the generated data and visualizations keep those suffixes.

Table 1. SVA dataset templates. Each row shows a prompt contrast where the correct answer is either a singular or plural verb form. In the placeholders, sg and pl denote singular and plural forms, respectively.

Template	Clean prompt template example	Corrupted prompt template example	Correct	Incorrect
single_0	The [Noun_{sg}] that [Verb_{sg}] the [Noun]	The [Noun_{pl}] that [Verb_{pl}] the [Noun]	singular verb	plural verb
plural_0	The [Noun_{pl}] that [Verb_{pl}] the [Noun]	The [Noun_{sg}] that [Verb_{sg}] the [Noun]	plural verb	singular verb
single_1	The [Noun_{sg}] from the [Noun_{pl}]	The [Noun_{pl}] from the [Noun_{pl}]	singular verb	plural verb
plural_1	The [Noun_{pl}] from the [Noun_{pl}]	The [Noun_{sg}] from the [Noun_{pl}]	plural verb	singular verb
single_2	The [Noun_{pl}] that the [Noun_{sg}]	The [Noun_{pl}] that the [Noun_{pl}]	singular verb	plural verb
plural_2	The [Noun_{pl}] that the [Noun_{pl}]	The [Noun_{pl}] that the [Noun_{sg}]	plural verb	singular verb
single_3	The [Noun_{pl}] the [Noun_{sg}]	The [Noun_{pl}] the [Noun_{pl}]	singular verb	plural verb
plural_3	The [Noun_{pl}] the [Noun_{pl}]	The [Noun_{pl}] the [Noun_{sg}]	plural verb	singular verb
single_4	The [Noun_{pl}] said the [Noun_{sg}]	The [Noun_{pl}] said the [Noun_{pl}]	singular verb	plural verb
plural_4	The [Noun_{pl}] said the [Noun_{pl}]	The [Noun_{pl}] said the [Noun_{sg}]	plural verb	singular verb
single_5	The [Noun_{sg}] the [Noun_{pl}] [Verb]	The [Noun_{pl}] the [Noun_{pl}] [Verb]	singular verb	plural verb
plural_5	The [Noun_{pl}] the [Noun_{pl}] [Verb]	The [Noun_{sg}] the [Noun_{pl}] [Verb]	plural verb	singular verb
single_6	The [Noun_{pl}] that the [Noun_{sg}]	The [Noun_{pl}] that the [Noun_{pl}]	singular verb	plural verb
plural_6	The [Noun_{pl}] that the [Noun_{pl}]	The [Noun_{pl}] that the [Noun_{sg}]	plural verb	singular verb
single_7	The [Noun_{pl}] the [Noun_{sg}]	The [Noun_{pl}] the [Noun_{pl}]	singular verb	plural verb
plural_7	The [Noun_{pl}] the [Noun_{pl}]	The [Noun_{pl}] the [Noun_{sg}]	plural verb	singular verb
single_8	The [Noun_{sg}] that the [Noun_{pl}] [Verb]	The [Noun_{pl}] that the [Noun_{pl}] [Verb]	singular verb	plural verb
plural_8	The [Noun_{pl}] that the [Noun_{pl}] [Verb]	The [Noun_{sg}] that the [Noun_{pl}] [Verb]	plural verb	singular verb
single_9	The [Noun_{sg}] that the [Noun_{pl}] [Verb]	The [Noun_{pl}] that the [Noun_{pl}] [Verb]	singular verb	plural verb
plural_9	The [Noun_{pl}] that the [Noun_{pl}] [Verb]	The [Noun_{sg}] that the [Noun_{pl}] [Verb]	plural verb	singular verb
single_10	The [Noun_{sg}]	The [Noun_{pl}]	singular verb	plural verb
plural_10	The [Noun_{pl}]	The [Noun_{sg}]	plural verb	singular verb
single_11	The [Noun_{sg}] [Verb_{sg}] [VP] and	The [Noun_{pl}] [Verb_{pl}] [VP] and	singular verb	plural verb
plural_11	The [Noun_{pl}] [Verb_{pl}] [VP] and	The [Noun_{sg}] [Verb_{sg}] [VP] and	plural verb	singular verb
single_12	The [Noun_{sg}] next to the [Noun_{pl}]	The [Noun_{pl}] next to the [Noun_{pl}]	singular verb	plural verb
plural_12	The [Noun_{pl}] next to the [Noun_{pl}]	The [Noun_{sg}] next to the [Noun_{pl}]	plural verb	singular verb
single_13	The [Noun_{sg}] [Verb_{sg}] and	The [Noun_{pl}] [Verb_{pl}] and	singular verb	plural verb
plural_13	The [Noun_{pl}] [Verb_{pl}] and	The [Noun_{sg}] [Verb_{sg}] and	plural verb	singular verb
single_14	The [Noun_{sg}] the [Noun_{pl}] [Verb]	The [Noun_{pl}] the [Noun_{pl}] [Verb]	singular verb	plural verb
plural_14	The [Noun_{pl}] the [Noun_{pl}] [Verb]	The [Noun_{sg}] the [Noun_{pl}] [Verb]	plural verb	singular verb

Table 2. IOI dataset templates. Each clean prompt is truncated immediately before the indirect-object answer; the corrupted prompt replaces the cueing name with a distractor [C].

Template	Clean prompt template example	Corrupted prompt template example	Correct	Incorrect
ABBA_00	Then, [A] and [B] went to the [PLACE]. [B] gave a [OBJECT] to	Then, [A] and [B] went to the [PLACE]. [C] gave a [OBJECT] to	[A]	[B]
ABBA_01	Then, [A] and [B] had a lot of fun at the [PLACE]. [B] gave a [OBJECT] to	Then, [A] and [B] had a lot of fun at the [PLACE]. [C] gave a [OBJECT] to	[A]	[B]
ABBA_02	Then, [A] and [B] were working at the [PLACE]. [B] decided to give a [OBJECT] to	Then, [A] and [B] were working at the [PLACE]. [C] decided to give a [OBJECT] to	[A]	[B]
ABBA_03	Then, [A] and [B] were thinking about going to the [PLACE]. [B] wanted to give a [OBJECT] to	Then, [A] and [B] were thinking about going to the [PLACE]. [C] wanted to give a [OBJECT] to	[A]	[B]
ABBA_04	Then, [A] and [B] had a long argument, and afterwards [B] said to	Then, [A] and [B] had a long argument, and afterwards [C] said to	[A]	[B]
ABBA_05	After [A] and [B] went to the [PLACE], [B] gave a [OBJECT] to	After [A] and [B] went to the [PLACE], [C] gave a [OBJECT] to	[A]	[B]
ABBA_06	When [A] and [B] got a [OBJECT] at the [PLACE], [B] decided to give it to	When [A] and [B] got a [OBJECT] at the [PLACE], [C] decided to give it to	[A]	[B]
BABA_00	Then, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to	Then, [B] and [A] went to the [PLACE]. [C] gave a [OBJECT] to	[A]	[B]
BABA_01	Then, [B] and [A] had a lot of fun at the [PLACE]. [B] gave a [OBJECT] to	Then, [B] and [A] had a lot of fun at the [PLACE]. [C] gave a [OBJECT] to	[A]	[B]
BABA_02	Then, [B] and [A] were working at the [PLACE]. [B] decided to give a [OBJECT] to	Then, [B] and [A] were working at the [PLACE]. [C] decided to give a [OBJECT] to	[A]	[B]
BABA_03	Then, [B] and [A] were thinking about going to the [PLACE]. [B] wanted to give a [OBJECT] to	Then, [B] and [A] were thinking about going to the [PLACE]. [C] wanted to give a [OBJECT] to	[A]	[B]
BABA_04	Then, [B] and [A] had a long argument, and afterwards [B] said to	Then, [B] and [A] had a long argument, and afterwards [C] said to	[A]	[B]
BABA_05	After [B] and [A] went to the [PLACE], [B] gave a [OBJECT] to	After [B] and [A] went to the [PLACE], [C] gave a [OBJECT] to	[A]	[B]
BABA_06	When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give it to	When [B] and [A] got a [OBJECT] at the [PLACE], [C] decided to give it to	[A]	[B]

Table 3. greater-than dataset template example. The task asks whether the next two-digit year suffix is greater than the suffix xy implied by the clean prompt.

Template	Clean prompt template example	Corrupted prompt template example	Correct	Incorrect
x	The [NOUN] lasted from the year 13xy to the year 13	The [NOUN] lasted from the year 1301 to the year 13	suffix $> xy$	suffix $\leq xy$

Table 4. single-double-quote dataset templates. Each base template is shown in both directions: _s expects the closing single-quote-parenthesis token, and _d expects the closing double-quote-parenthesis token.

Template	Clean prompt template example	Corrupted prompt template example	Correct	Incorrect
append_call_s	if [date bounds fail]: [RESULT_LIST].append('Invalid Input	if [date bounds fail]: [RESULT_LIST].append("Invalid Input	')	")
append_call_d	if [date bounds fail]: [RESULT_LIST].append("Invalid Input	if [date bounds fail]: [RESULT_LIST].append('Invalid Input	")	')
constructor_call_s	[OBJECT_VAR] = Project('[PROJECT_NAME]	[OBJECT_VAR] = Project("[PROJECT_NAME]	')	")
constructor_call_d	[OBJECT_VAR] = Project("[PROJECT_NAME]	[OBJECT_VAR] = Project('[PROJECT_NAME]	")	')

for_loop_s	for [ITEM_VAR] in [ITEM_SET]:
	if len([ITEM_VAR]) == 0:
	print('invalid
		for [ITEM_VAR] in [ITEM_SET]:
		if len([ITEM_VAR]) == 0:
		print("invalid
			')	")

for_loop_d	for [ITEM_VAR] in [ITEM_SET]:
	if len([ITEM_VAR]) == 0:
	print("invalid
		for [ITEM_VAR] in [ITEM_SET]:
		if len([ITEM_VAR]) == 0:
		print('invalid
			")	')

function_call_s	[FUNC_NAME]('[FILE_NAME]	[FUNC_NAME]("[FILE_NAME]	')	")
function_call_d	[FUNC_NAME]("[FILE_NAME]	[FUNC_NAME]('[FILE_NAME]	")	')
if_clause_1_s	if [COND]: print('There is no topological sort, the graph has a cycle	if [COND]: print("There is no topological sort, the graph has a cycle	')	")
if_clause_1_d	if [COND]: print("There is no topological sort, the graph has a cycle	if [COND]: print('There is no topological sort, the graph has a cycle	")	')
if_clause_2_s	if degree not in range(3, [LIMIT_VAR]-1): print('Degree should be in the range [3, [LIMIT_VAR]-1]	if degree not in range(3, [LIMIT_VAR]-1): print("Degree should be in the range [3, [LIMIT_VAR]-1]	')	")
if_clause_2_d	if degree not in range(3, [LIMIT_VAR]-1): print("Degree should be in the range [3, [LIMIT_VAR]-1]	if degree not in range(3, [LIMIT_VAR]-1): print('Degree should be in the range [3, [LIMIT_VAR]-1]	")	')
if_clause_3_s	if [ALT_VAR] < [DIST_VAR].get([NODE_VAR], float('inf	if [ALT_VAR] < [DIST_VAR].get([NODE_VAR], float("inf	')	")
if_clause_3_d	if [ALT_VAR] < [DIST_VAR].get([NODE_VAR], float("inf	if [ALT_VAR] < [DIST_VAR].get([NODE_VAR], float('inf	")	')
method_call_s	[GRAPH_VAR].read_data('[GRAPH_FILE]	[GRAPH_VAR].read_data("[GRAPH_FILE]	')	")
method_call_d	[GRAPH_VAR].read_data("[GRAPH_FILE]	[GRAPH_VAR].read_data('[GRAPH_FILE]	")	')

nested_loop_s	for [OUTER] in [SET]:
	for [INNER] in [OUTER]:
	if ([INNER] == ()):
	print('No [INNER] at
	the current iteration
		for [OUTER] in [SET]:
		for [INNER] in [OUTER]:
		if ([INNER] == ()):
		print("No [INNER] at
		the current iteration
			')	")

nested_loop_d	for [OUTER] in [SET]:
	for [INNER] in [OUTER]:
	if ([INNER] == ()):
	print("No [INNER] at
	the current iteration
		for [OUTER] in [SET]:
		for [INNER] in [OUTER]:
		if ([INNER] == ()):
		print('No [INNER] at
		the current iteration
			")	')

print_stmt_1_s	print('[PRINT_TEXT]	print("[PRINT_TEXT]	')	")
print_stmt_1_d	print("[PRINT_TEXT]	print('[PRINT_TEXT]	")	')
print_stmt_2_s	print('[USAGE_TEXT]	print("[USAGE_TEXT]	')	")
print_stmt_2_d	print("[USAGE_TEXT]	print('[USAGE_TEXT]	")	')
print_stmt_3_s	print('[QUESTION_TEXT]	print("[QUESTION_TEXT]	')	")
print_stmt_3_d	print("[QUESTION_TEXT]	print('[QUESTION_TEXT]	")	')
print_stmt_4_s	print('[STATUS_TEXT]	print("[STATUS_TEXT]	')	")
print_stmt_4_d	print("[STATUS_TEXT]	print('[STATUS_TEXT]	")	')
raise_error_s	raise ValueError('[ERROR_TEXT]	raise ValueError("[ERROR_TEXT]	')	")
raise_error_d	raise ValueError("[ERROR_TEXT]	raise ValueError('[ERROR_TEXT]	")	')
string_split_s	def [FUNC]([ARG]): [LEFT], [RIGHT] = [ARG].split('-	def [FUNC]([ARG]): [LEFT], [RIGHT] = [ARG].split("-	')	")
string_split_d	def [FUNC]([ARG]): [LEFT], [RIGHT] = [ARG].split("-	def [FUNC]([ARG]): [LEFT], [RIGHT] = [ARG].split('-	")	')
strftime_call_s	[TIME_VAR] = datetime.datetime.now() .strftime('%Y%m%d_%H%M%S	[TIME_VAR] = datetime.datetime.now() .strftime("%Y%m%d_%H%M%S	')	")
strftime_call_d	[TIME_VAR] = datetime.datetime.now() .strftime("%Y%m%d_%H%M%S	[TIME_VAR] = datetime.datetime.now() .strftime('%Y%m%d_%H%M%S	")	')

Table 5. else-elif dataset templates. The task contrasts Python branches where `else` should be followed by `:\n`, while `elif` should be followed by any other token. In visualizations, template labels additionally carry `_else` or `_elif` suffixes for the two directions; these suffixes are omitted here for space, and the prompt examples show one representative direction.

Template	Clean prompt template example	Corrupted prompt template example	Correct	Incorrect

return_result	if [RESULT]:
	[CACHE] = [RESULT]
	return [RESULT]
	else
		if [RESULT]:
		[CACHE] = [RESULT]
		return [RESULT]
		elif
			:\n	other tokens

answer_check	if eval([PROBLEM]) == [ANSWER]:
	print([MSG])
	elif
		if eval([PROBLEM]) == [ANSWER]:
		print([MSG])
		else
			other tokens	:\n

append_guard	if [COUNT_MAP][[NODE].topic] >= [TARGET]:
	[LIST].append([NODE])
	else
		if [COUNT_MAP][[NODE].topic] >= [TARGET]:
		[LIST].append([NODE])
		elif
			:\n	other tokens

program_dir_chain	if [DIR] == 0:
	[DIR] = [NEG]
	elif [DIR] == 1:
	[DIR] = [POS]
	elif
		if [DIR] == 0:
		[DIR] = [NEG]
		elif [DIR] == 1:
		[DIR] = [POS]
		else
			other tokens	:\n

schedule_branch	if [TIME] > [LAST]:
	[START] = [LAST] + 1
	else
		if [TIME] > [LAST]:
		[START] = [LAST] + 1
		elif
			:\n	other tokens

strategy_branch	if [STRATEGY] == 'bfs':
	return bfs(...)
	elif [STRATEGY] == 'dfs':
	return dfs(...)
	elif
		if [STRATEGY] == 'bfs':
		return bfs(...)
		elif [STRATEGY] == 'dfs':
		return dfs(...)
		else
			other tokens	:\n

equality_case	if [CONSTRAINT].is_equality():
	return [Z3] == -[COEFF]['1']
	else
		if [CONSTRAINT].is_equality():
		return [Z3] == -[COEFF]['1']
		elif
			:\n	other tokens

stats_update	if [ROOT] not in [STATS]:
	[STATS][[ROOT]] =
	([AREA], [COUNT])
	elif
		if [ROOT] not in [STATS]:
		[STATS][[ROOT]] =
		([AREA], [COUNT])
		else
			other tokens	:\n

inline_threshold	if ([NUM] <= [LIMIT]):
	return [FUNC_CALL]
	else
		if ([NUM] <= [LIMIT]):
		return [FUNC_CALL]
		elif
			:\n	other tokens

string_align	if [CHAR] in [STRING]:
	[LEFT], [RIGHT] =
	[STRING].split([CHAR], 1)
	elif
		if [CHAR] in [STRING]:
		[LEFT], [RIGHT] =
		[STRING].split([CHAR], 1)
		else
			other tokens	:\n

binary_search	if [DATA][[INDEX]] < [VALUE]:
	[RESULT] = [INDEX]
	[LOW] = [INDEX] + 1
	else
		if [DATA][[INDEX]] < [VALUE]:
		[RESULT] = [INDEX]
		[LOW] = [INDEX] + 1
		elif
			:\n	other tokens

truthy_call	if [ROOM]:
	set_parameter([BOARD], [PARAM],
	[ROOM_NAME] + ' - ' + [ROOM])
	elif
		if [ROOM]:
		set_parameter([BOARD], [PARAM],
		[ROOM_NAME] + ' - ' + [ROOM])
		else
			other tokens	:\n

compare_objects	if isinstance([LEFT], (list, tuple)):
	if isinstance([RIGHT], (list, tuple)):
	compare_objects(...)
	else
		if isinstance([LEFT], (list, tuple)):
		if isinstance([RIGHT], (list, tuple)):
		compare_objects(...)
		elif
			:\n	other tokens

cell_toggle	if [BOARD][[ROW]][[COL]] == [OFF]:
	[BOARD][[ROW]][[COL]] = 1
	elif
		if [BOARD][[ROW]][[COL]] == [OFF]:
		[BOARD][[ROW]][[COL]] = 1
		else
			other tokens	:\n

random_split	if random.random() < 0.5:
	[OUT].append(
	f"ML,{[PEER]},{[ELEMENT]}")
	else
		if random.random() < 0.5:
		[OUT].append(
		f"ML,{[PEER]},{[ELEMENT]}")
		elif
			:\n	other tokens

none_default	if [END] is None:
	[GOAL] = ([ROWS]-1, [COLS]-1)
	elif
		if [END] is None:
		[GOAL] = ([ROWS]-1, [COLS]-1)
		else
			other tokens	:\n

E. Proof of Theorem 3.5

Proof. We first consider the CEAP scoring function. Given any branch $b \in B$, suppose the edge has size $y \in \mathbb{R}^{d_b}$, we have:

$$\begin{aligned}
I_{\text{Cond}}(y_b) &= \sum_{i\in[d_b]} \sum_{j\in[d_{in}]} (x_{1j} - x_{0j}) \int_0^1 \frac{\partial F(x_0 + \alpha(x_1 - x_0))}{\partial y_{b_i}} \frac{\partial y_{b_i}}{\partial x_j} \, \mathrm{d}\alpha \\
&= \sum_{i\in[d_b]} \int_0^1 \frac{\partial F(x_0 + \alpha(x_1 - x_0))}{\partial y_{b_i}} \underbrace{\sum_{j\in[d_{in}]} \frac{\partial y_{b_i}}{\partial x_j} \, \mathrm{d}\alpha (x_{1j} - x_{0j})}_{(=\mathrm{d}y_{b_i})} \\
&= \sum_{i\in[d_b]} \int_0^1 \frac{\partial f_b(y_{b_i}(\alpha))}{\partial y_{b_i}} \mathrm{d}y_{b_i}(\alpha) = \int_{\gamma_{y_b}} \sum_{i\in[d_b]} \frac{\partial f_b(y_{b_i})}{\partial y_{b_i}} \mathrm{d}y_{b_i} \\
&= \int_{\gamma_{y_b}} (\nabla_{y_b} f_b) \cdot \mathbf{d}\boldsymbol{r} = f_b(x_1) - f_b(x_0) = \Delta f_b.
\end{aligned}$$

In the above, γ_{y_b} is the trajectory that y_b follows when α goes from 0 to 1, and the second last equality is due to that the integrand is a gradient, and thus the result is path-independent. The above derivation shows that conductance satisfies additive order preservation, almost by design.

On the other hand, we can come up with the following counterexample for IG. Suppose we are studying a two-branched function: $F = g_1(y_1) + g_2(y_2) = g_1(f_1(x)) + g_2(f_2(x))$, where

$$f_1(x) = \begin{cases} \frac{1}{3}x, & x < 1, \\ x - \frac{2}{3}, & 1 \le x < 2, \\ \frac{1}{3}x + \frac{2}{3}, & x \ge 2, \end{cases} \qquad g_1(y_1) = \begin{cases} y_1, & y_1 < \frac{1}{3}, \\ \frac{1}{3}, & \frac{1}{3} \le y_1 < \frac{4}{3}, \\ y_1 - 1, & y_1 \ge \frac{4}{3}, \end{cases} \qquad \begin{aligned} f_2(x) &= x, \\ g_2(y_2) &= \frac{1}{3}y_2. \end{aligned}$$

We study the process where the input x is moved from 0 to 3. In this case, there are two branches, denoted by set $B = \{1, 2\}$. We can compute that $|\Delta f_1| = \frac{2}{3}$, and $|\Delta f_2| = 1$. We can also compute:

$$I_{\text{IG}}(y_1) = \frac{5}{3} \int_0^1 \frac{\partial F}{\partial y_1}\bigg|_{x=3\alpha} \mathrm{d}\alpha = \frac{5}{3} \int_0^3 \frac{\partial F}{\partial y_1}\bigg|_{x} \mathrm{d}x \tag{7}$$

$$= \frac{5}{3} \left(\int_0^1 \mathrm{d}x + \int_1^2 0 \mathrm{d}x + \int_2^3 \mathrm{d}x \right) = \frac{10}{3}. \tag{8}$$

$$I_{\text{IG}}(y_2) = 3 \int_0^1 \frac{\partial F}{\partial y_2}\bigg|_{x=3\alpha} \mathrm{d}\alpha = 3 \int_0^3 \frac{\partial F}{\partial y_2}\bigg|_{x} \mathrm{d}x = 3 \int_0^3 \frac{1}{3} \mathrm{d}x = 3.$$

Hence $|\Delta f_1| < |\Delta f_2|$ and yet $|I_{\text{IG}}(y_1)| > |I_{\text{IG}}(y_2)|$, breaking additive order preservation. □

Remark E.1. One might notice that $g_1(\cdot)$ above is not differentiable, and thus we are implicitly using the one-sided derivative in place of $\frac{\partial F}{\partial y_1}$. However, our statement should still hold for such non-smooth functions. It is easy to see that non-differentiability is not the root cause of IG breaking additive order preservation. First, in the counterexample, the non-differentiability occurs on a Lebesgue-null set and hence does not affect the integral. Second, we can smoothen the kinks in $g_1(\cdot)$ using techniques like huberized ReLU (Chatterji et al., 2021) or softplus (Weiß et al., 2022), where the kinks are replaced with smooth arcs whose span can be arbitrarily small. Thus, $I_{\text{IG}}(y_1)$ for the smoothened g_1 can be arbitrarily close to what we compute above. Constructing such g_1 might involve tedious notation and blur the main message. We settle for the current counterexample for its simplicity and clarity: The reason why the magnitude of $I_{\text{IG}}(y_1)$ relative to $I_{\text{IG}}(y_2)$ seems to lose its meaning is that the integrals receives the same weight $\frac{5}{3}$ in Equation (8), while, in fact, the weighting should reflect how much y_1 moves when x moves through each of the three integration intervals.

Remark E.2. The fact that conductance satisfies additive order preservation may be understood as a result of linearity and partition consistency (Dhamdhere et al., 2019) of conductance. Nevertheless, these two properties are not necessary to satisfy additive order preservation. In fact, if we engineer a new scoring method, which simply assigns to each edge the squared conductance score, additive order preservation is still satisfied, while linearity and partition consistency will be breached.

F. Complete Results for Pairwise Jaccard Index and Unfaithfulness

In this section, we compare unfaithfulness and pairwise Jaccard index yielded by CEAP and EAP-IG. Overall, we found that while CEAP does not have a consistent advantage over EAP-IG in terms of unfaithfulness, it clearly outperforms the latter for pairwise Jaccard index, or at the very least matches it. Specifically, for unfaithfulness, the quicker it drops with the increase of the number of edges, the better. This is based on the principle of minimality (Wang et al., 2023). For the pairwise Jaccard index, the higher, the better.

A circuit-finding scheme could in principle game the PJI metric by always selecting the same circuit with no regard to the task. We check that this is not the case for CEAP: it achieves low unfaithfulness across the tasks shown below. Nonetheless, we do not find compelling evidence that CEAP consistently attains lower unfaithfulness than EAP-IG. For instance, on GPT-2 XL, CEAP achieves lower unfaithfulness with smaller circuits for nearly all SVA templates, but the outcomes for IOI and greater-than are inconclusive. This is understandable: both CEAP and EAP-IG are path-integral methods that characterize trajectories in the activation space as inputs move from corrupted samples to their clean counterparts. However, once we patch the activations of the excluded edges, the resulting trajectory no longer aligns with the paths used to assign their scores. Compounded with the nonlinearity of the network, this introduces errors that are difficult to control or even quantify and can drown out the advantage of CEAP over EAP-IG.

F.1. GPT-2 XL

F.1.1. SVA

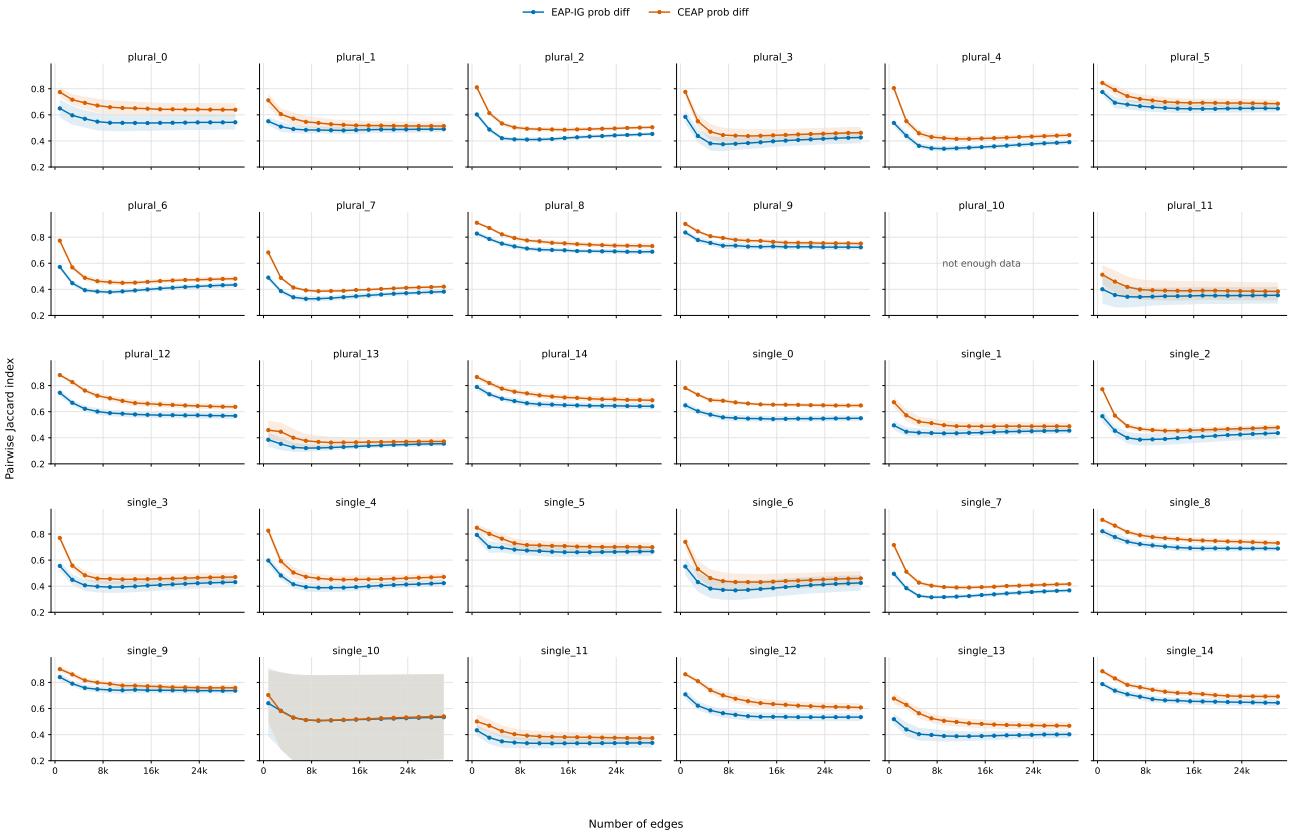


Figure 9. Pairwise Jaccard index vs. number of edges for GPT-2 XL on SVA.

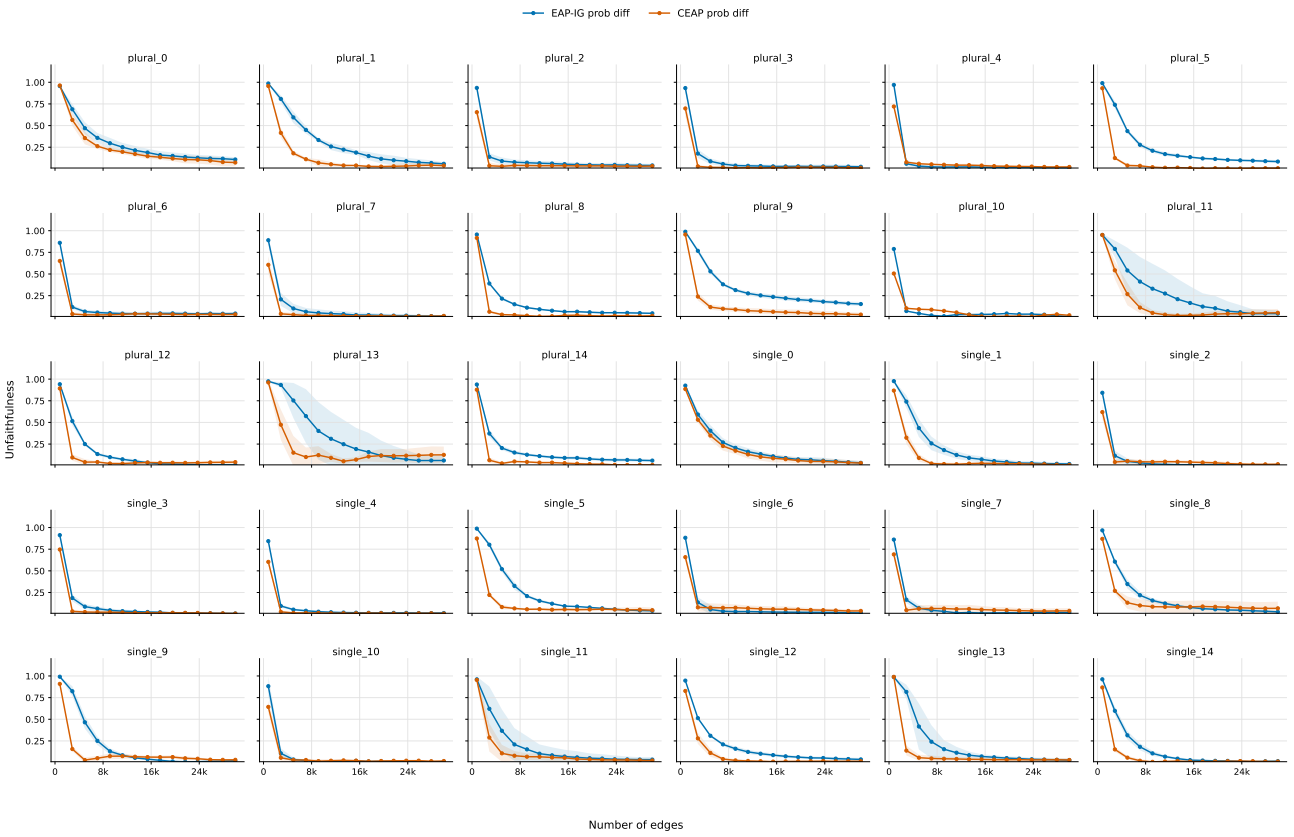


Figure 10. Unfaithfulness vs. number of edges for GPT-2 XL on SVA.

F.1.2. IOI

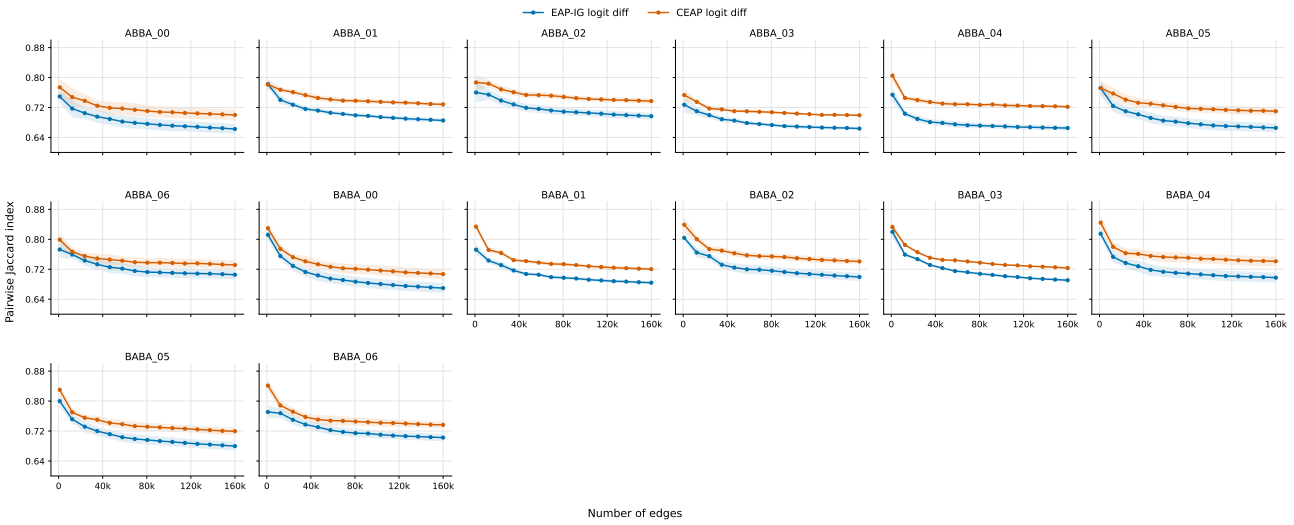


Figure 11. Pairwise Jaccard index vs. number of edges for GPT-2 XL on IOI.

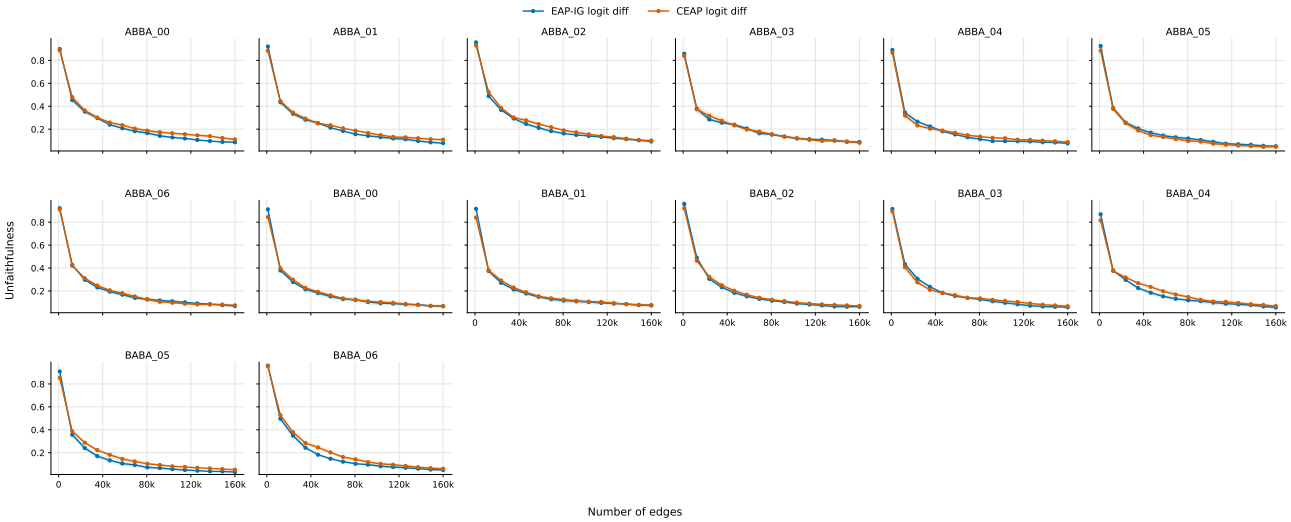


Figure 12. Unfaithfulness vs. number of edges for GPT-2 XL on IOI.

F.1.3. GREATER-THAN

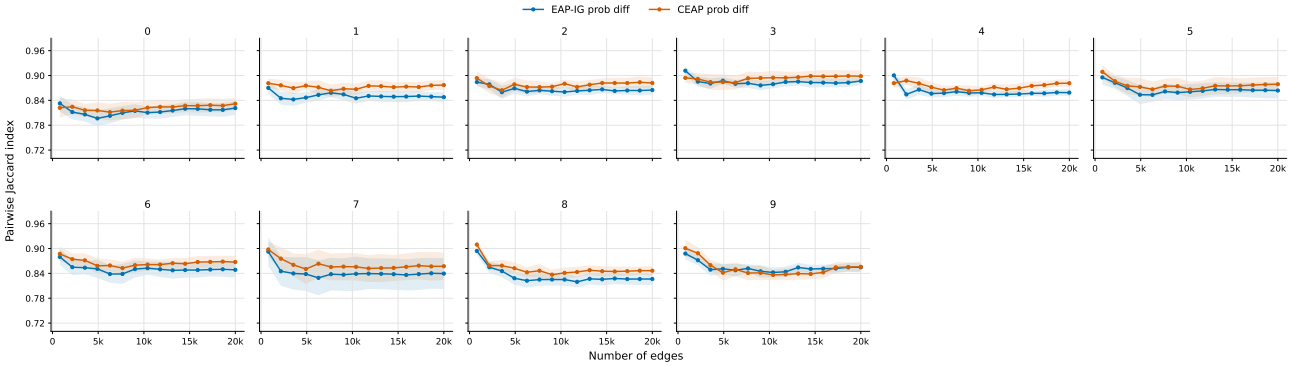


Figure 13. Pairwise Jaccard index vs. number of edges for GPT-2 XL on Greater-than.

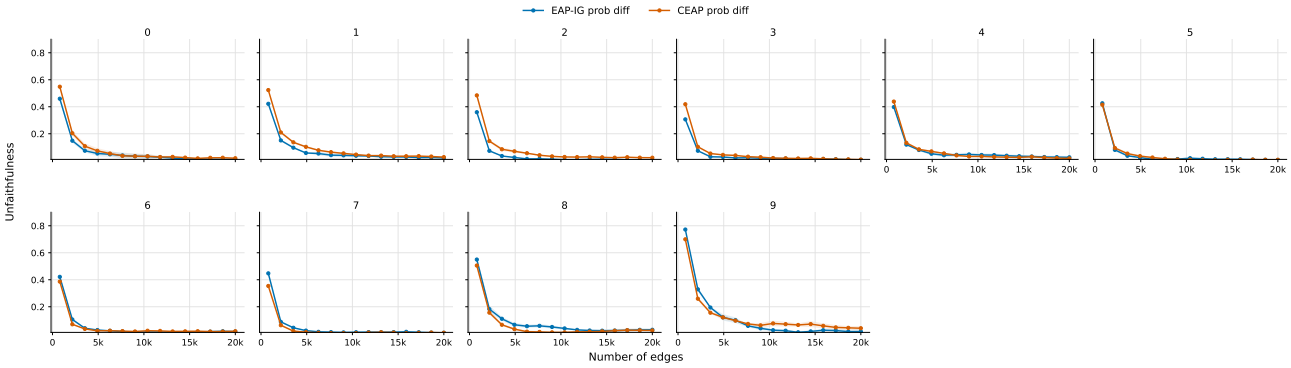


Figure 14. Unfaithfulness vs. number of edges for GPT-2 XL on Greater-than.

F.2. Pythia-2.8B

F.2.1. SVA

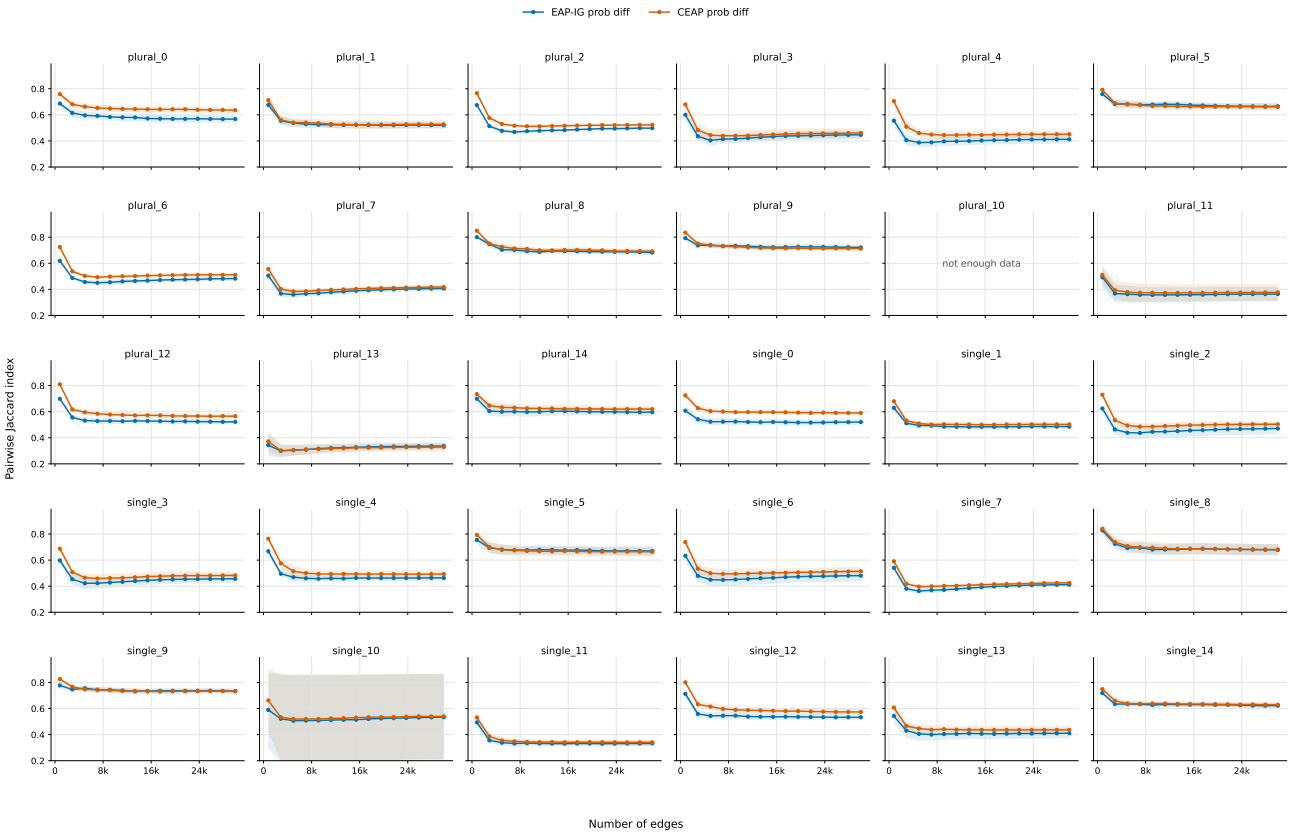


Figure 15. Pairwise Jaccard index vs. number of edges for Pythia-2.8B on SVA.

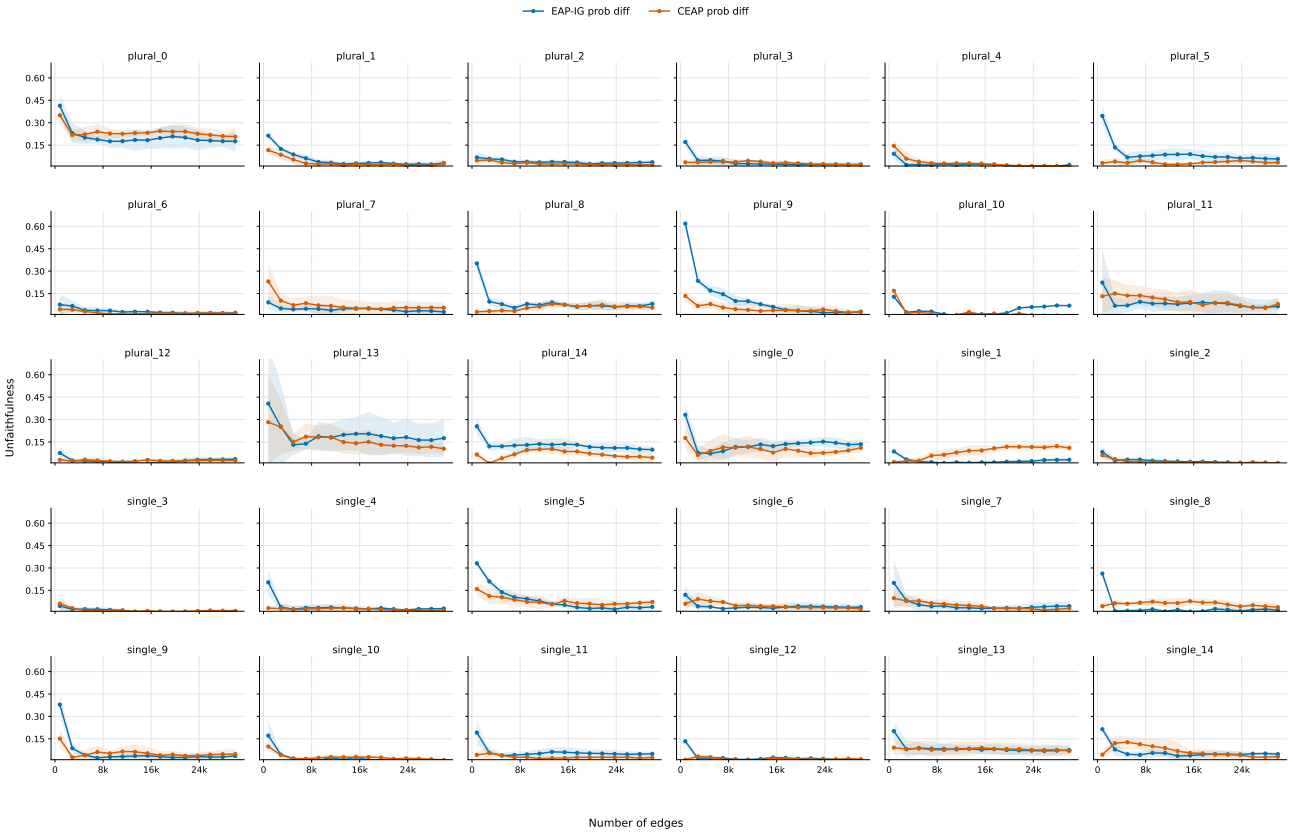


Figure 16. Unfaithfulness vs. number of edges for Pythia-2.8B on SVA.

F.2.2. IOI

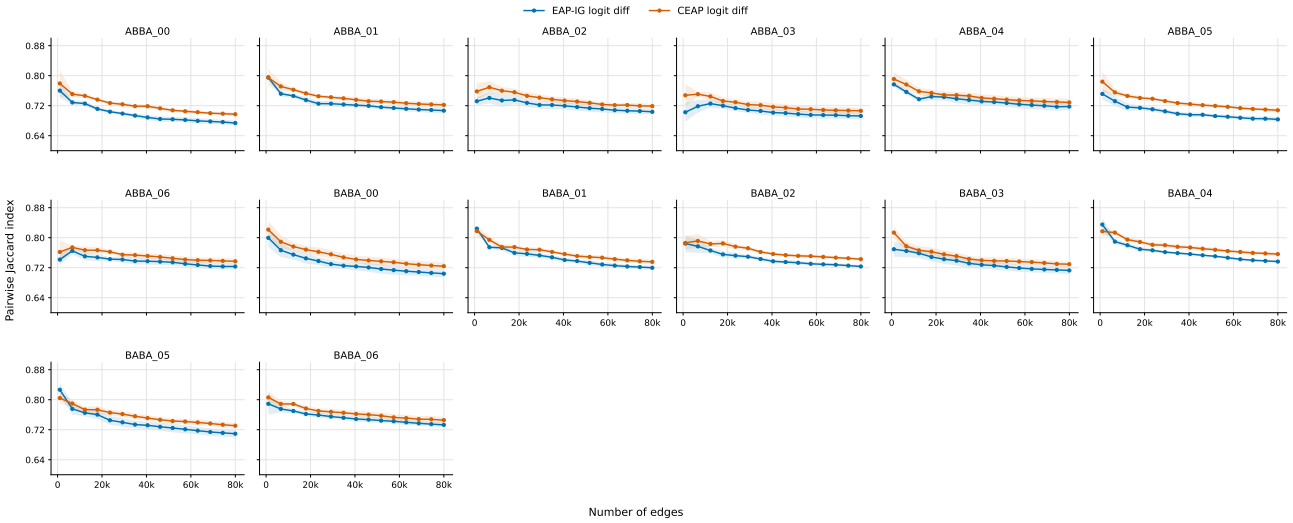


Figure 17. Pairwise Jaccard index vs. number of edges for Pythia-2.8B on IOI.

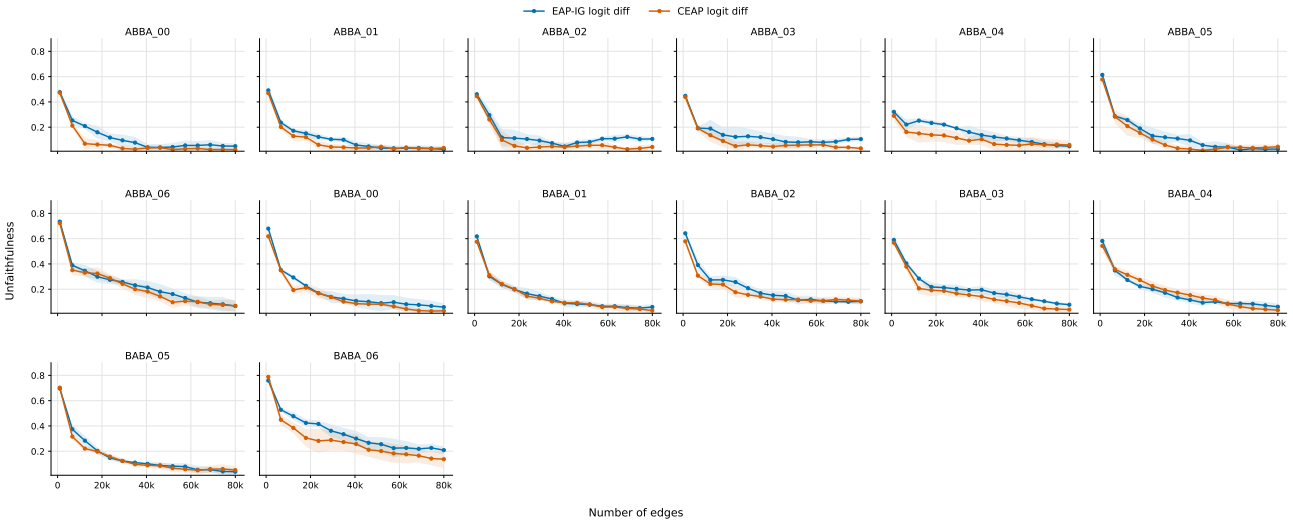


Figure 18. Unfaithfulness vs. number of edges for Pythia-2.8B on IOI.

F.2.3. GREATER-THAN

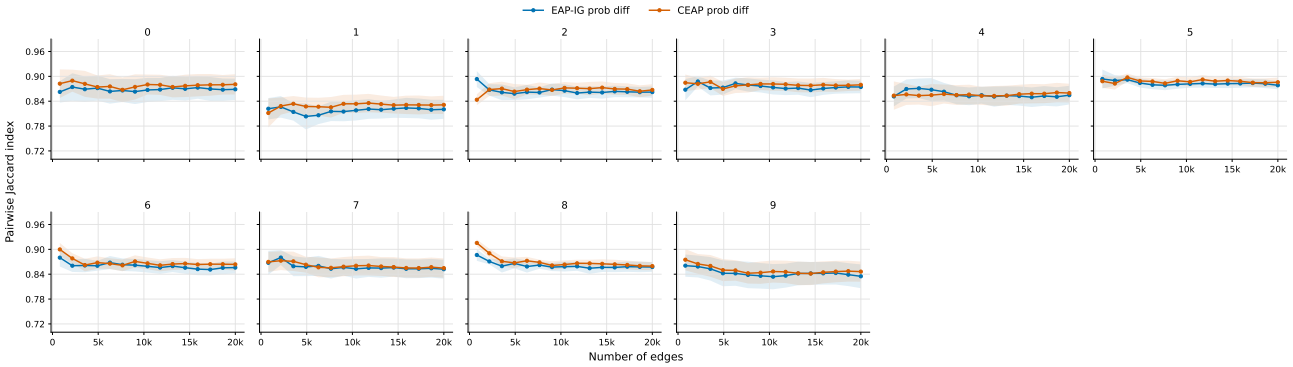


Figure 19. Pairwise Jaccard index vs. number of edges for Pythia-2.8B on Greater-than.

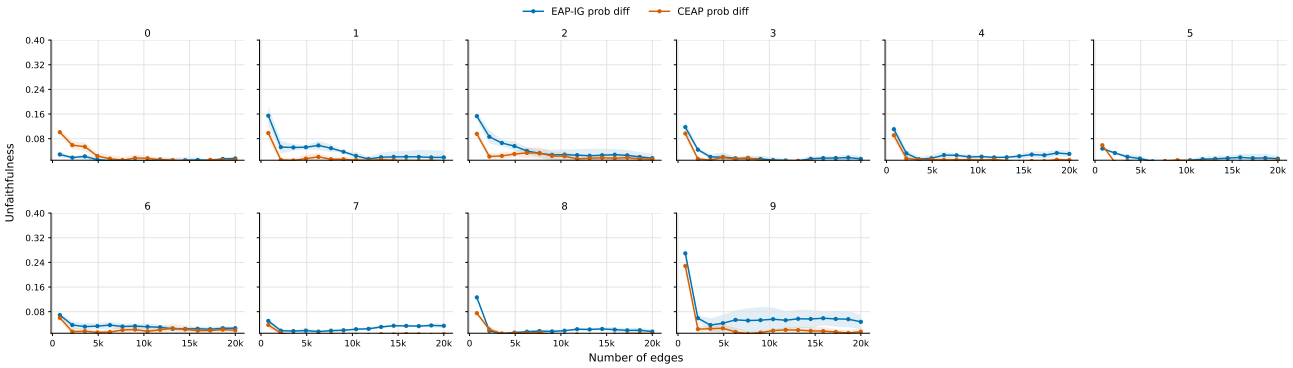


Figure 20. Unfaithfulness vs. number of edges for Pythia-2.8B on Greater-than.

F.3. GPT-2 small

F.3.1. SVA

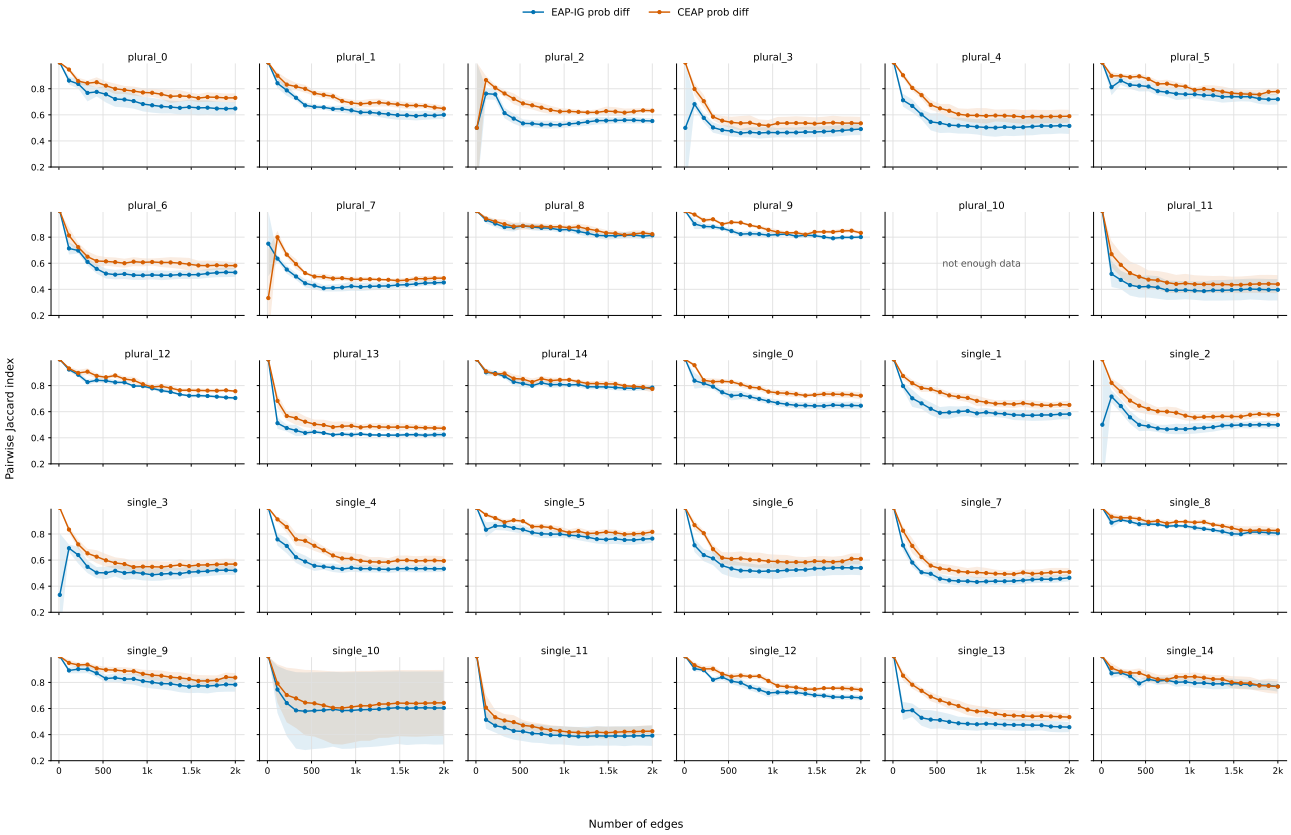


Figure 21. Pairwise Jaccard index vs. number of edges for GPT-2 small on SVA.

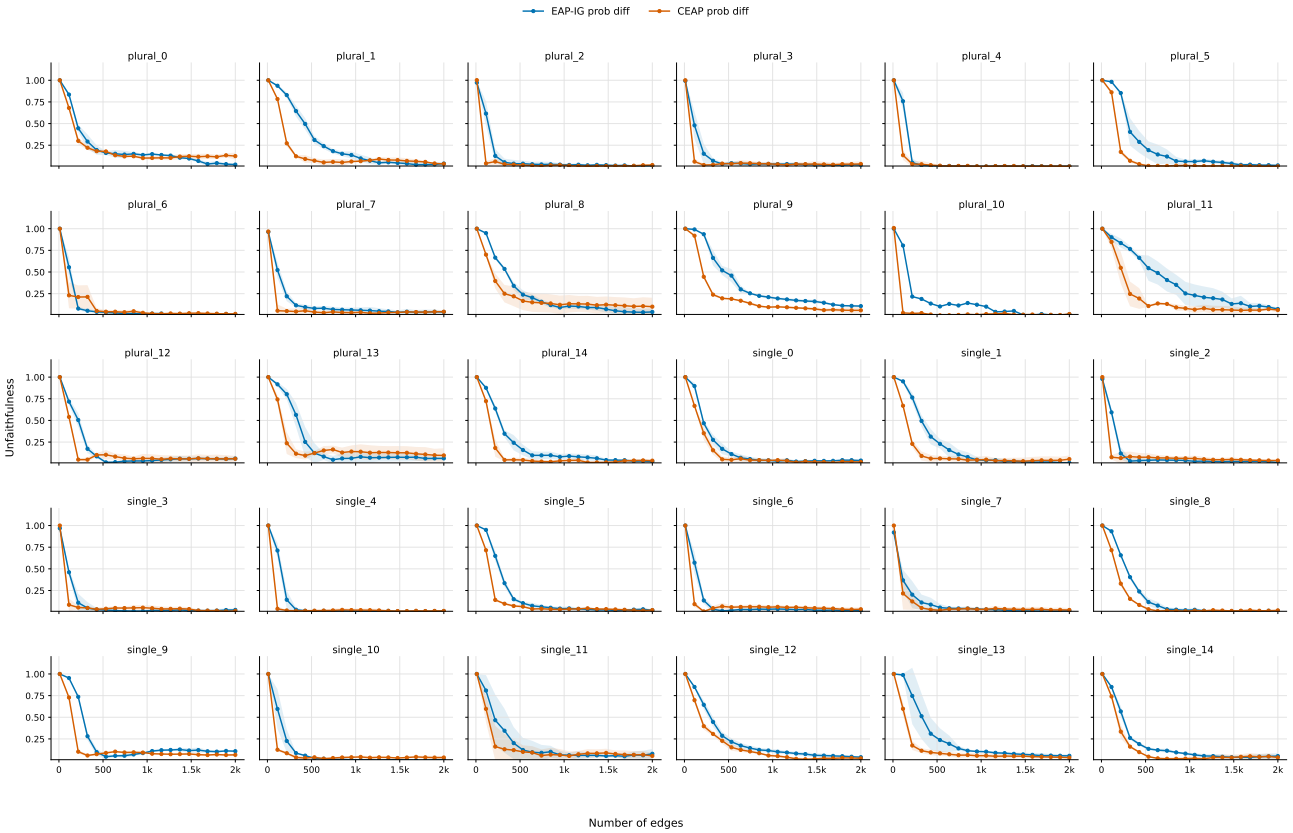


Figure 22. Unfaithfulness vs. number of edges for GPT-2 small on SVA.

F.3.2. IOI

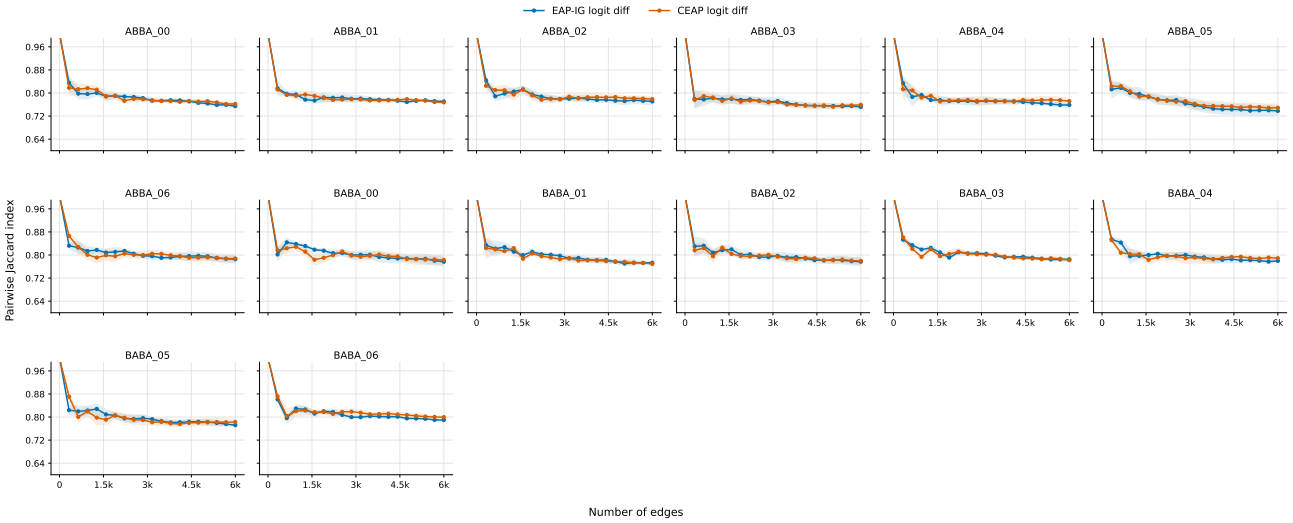


Figure 23. Pairwise Jaccard index vs. number of edges for GPT-2 small on IOI.

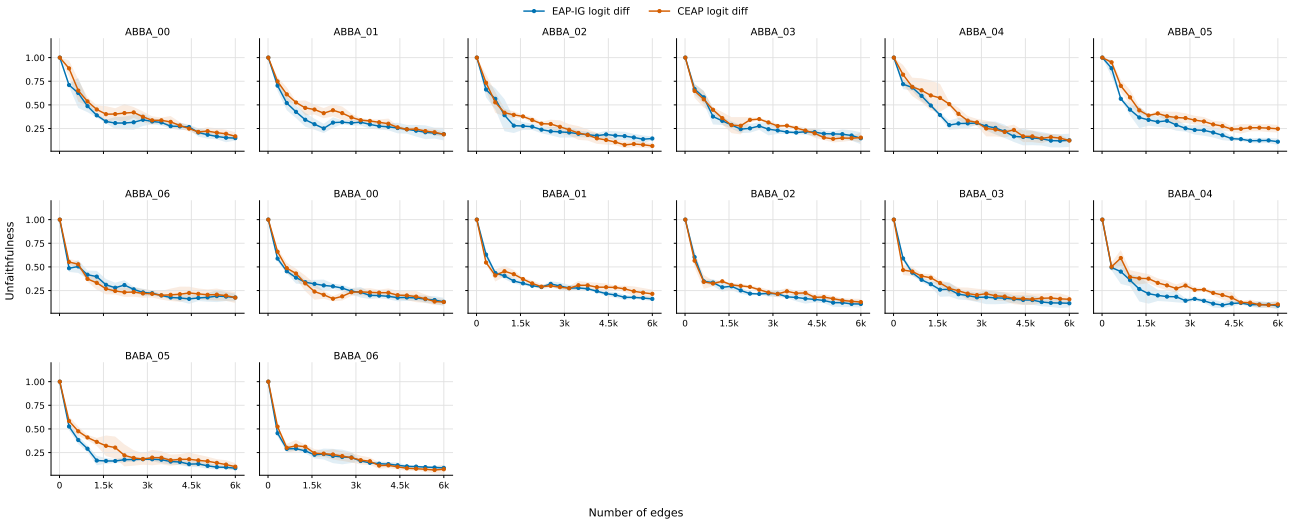


Figure 24. Unfaithfulness vs. number of edges for GPT-2 small on IOI.

F.3.3. GREATER-THAN

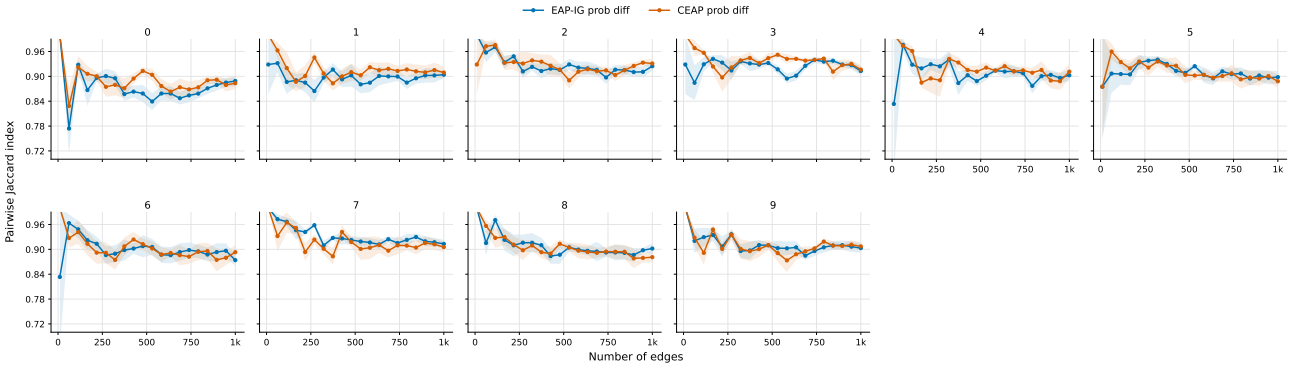


Figure 25. Pairwise Jaccard index vs. number of edges for GPT-2 small on Greater-than.

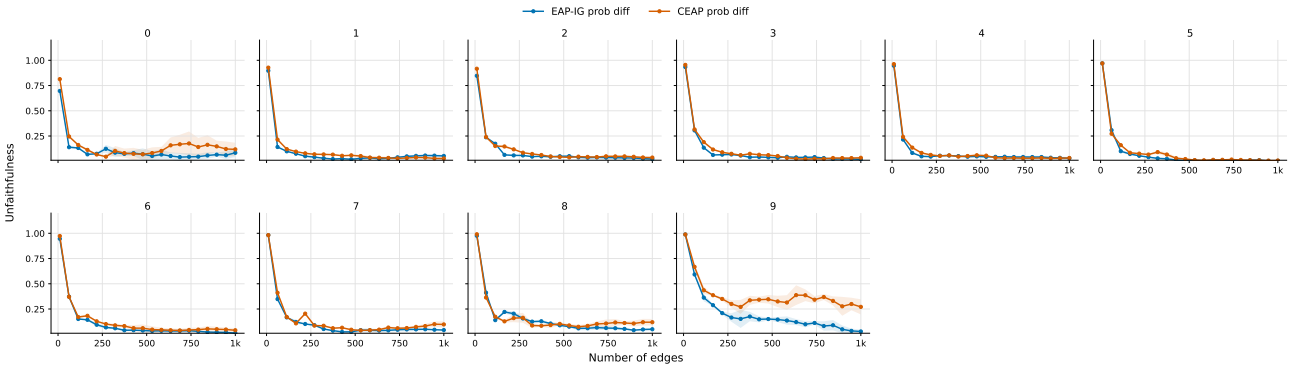


Figure 26. Unfaithfulness vs. number of edges for GPT-2 small on Greater-than.

F.4. Pythia-160M

F.4.1. SVA

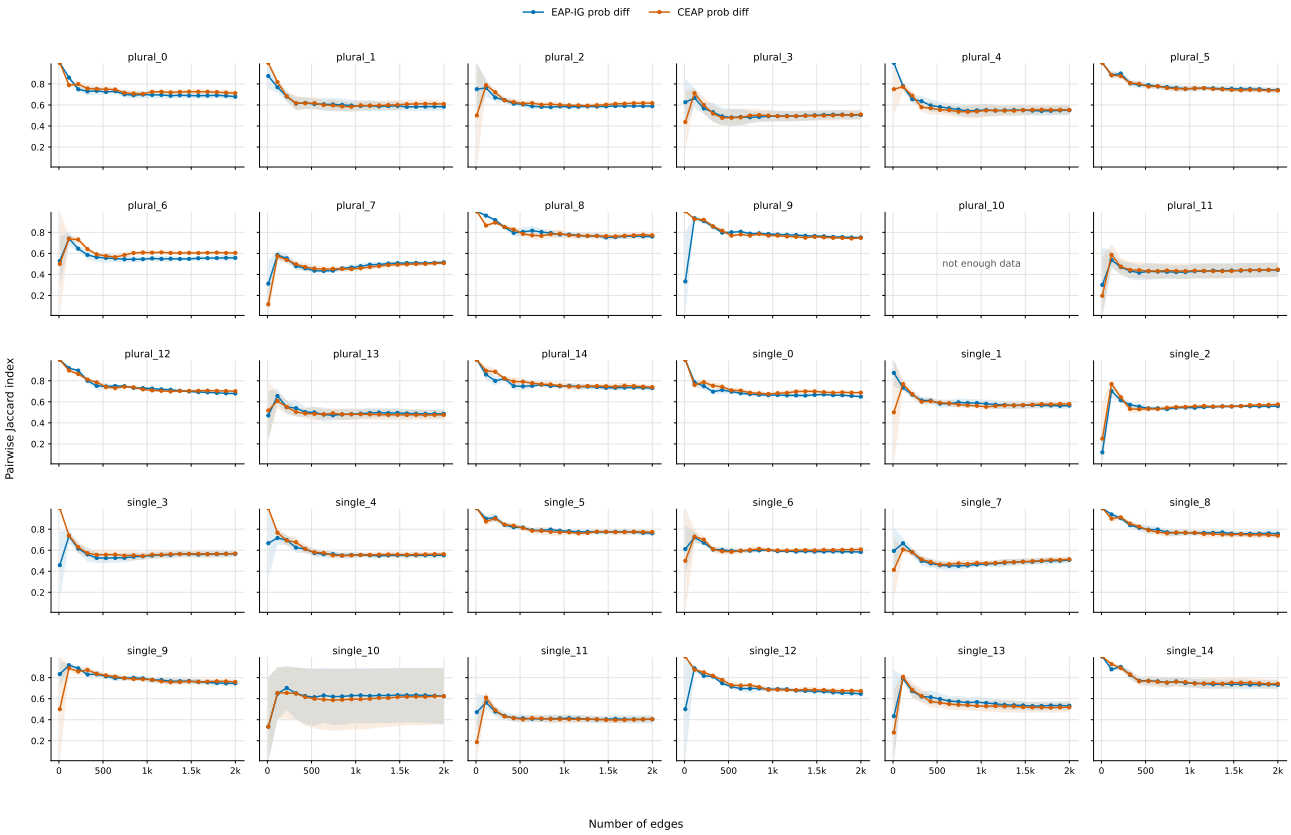


Figure 27. Pairwise Jaccard index vs. number of edges for Pythia-160M on SVA.

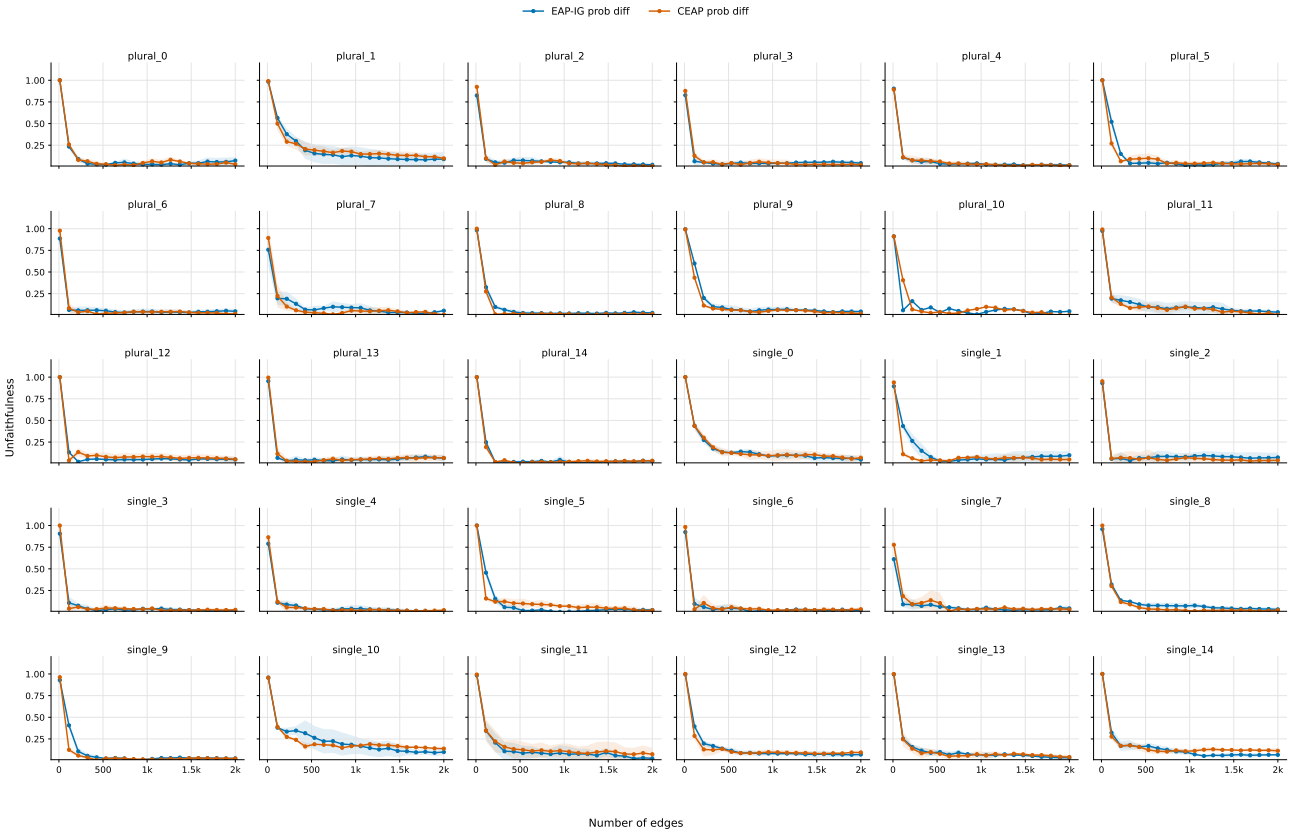


Figure 28. Unfaithfulness vs. number of edges for Pythia-160M on SVA.

F.4.2. IOI

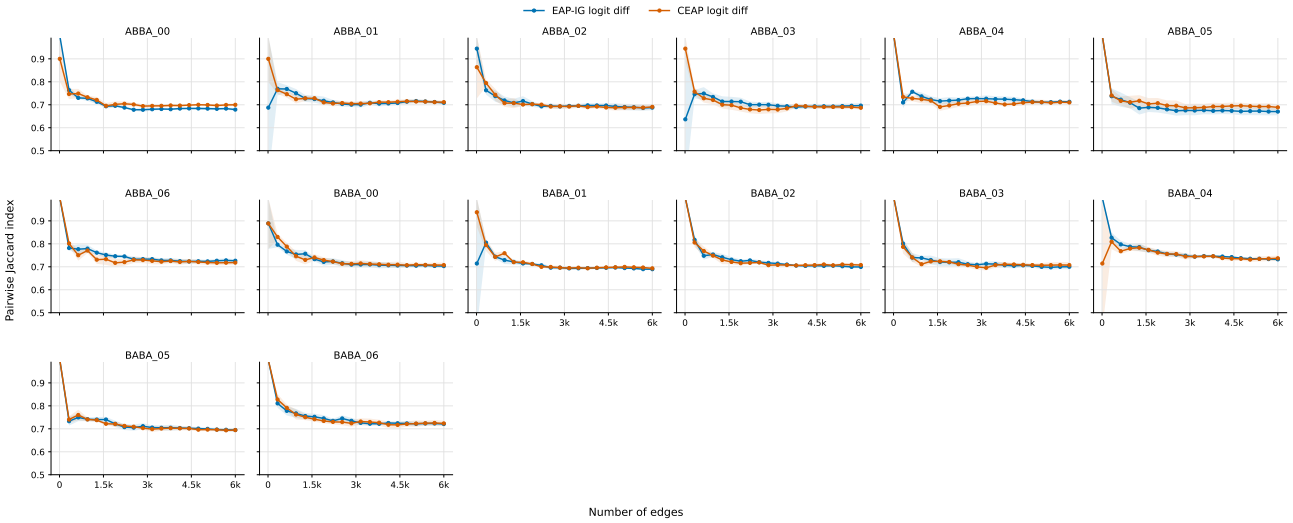


Figure 29. Pairwise Jaccard index vs. number of edges for Pythia-160M on IOI.

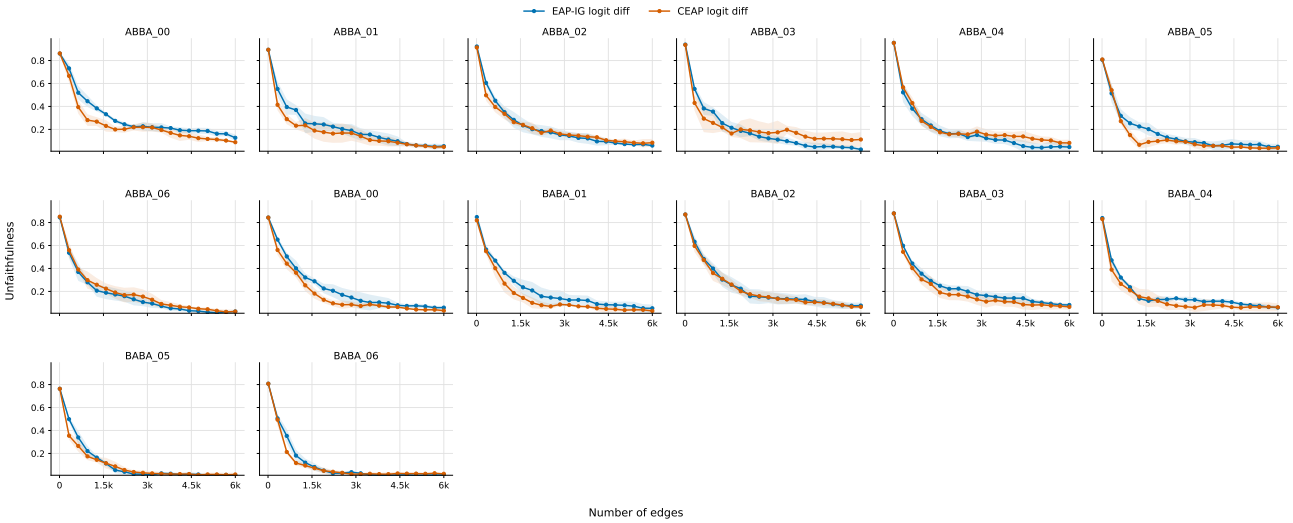


Figure 30. Unfaithfulness vs. number of edges for Pythia-160M on IOI.

F.4.3. GREATER-THAN

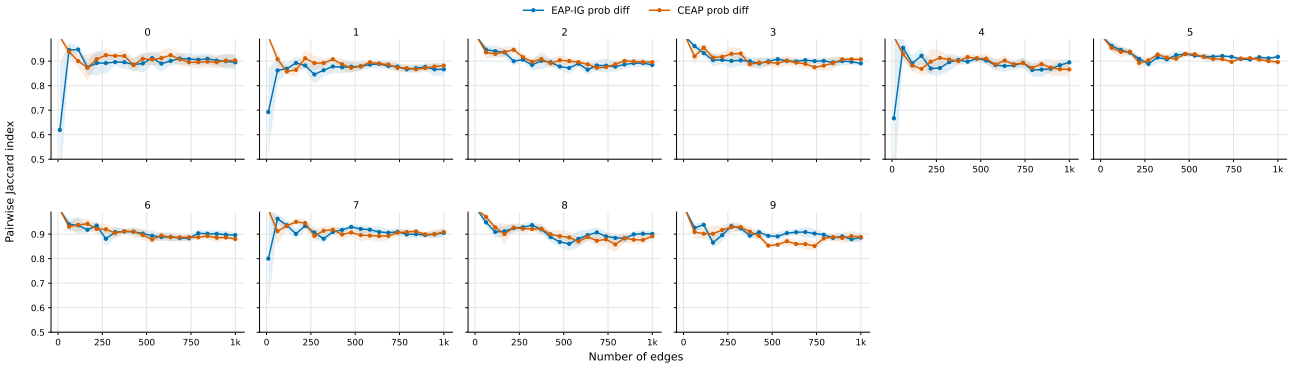


Figure 31. Pairwise Jaccard index vs. number of edges for Pythia-160M on Greater-than.

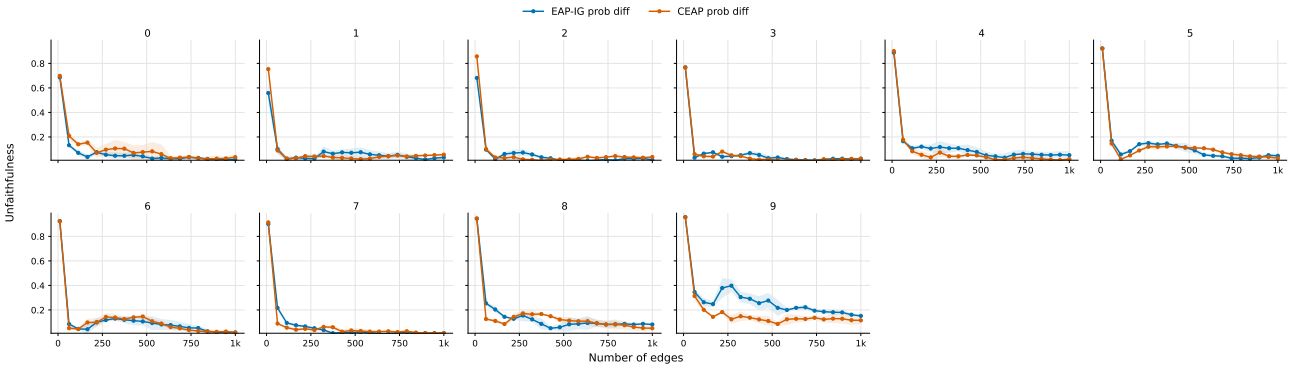


Figure 32. Unfaithfulness vs. number of edges for Pythia-160M on Greater-than.

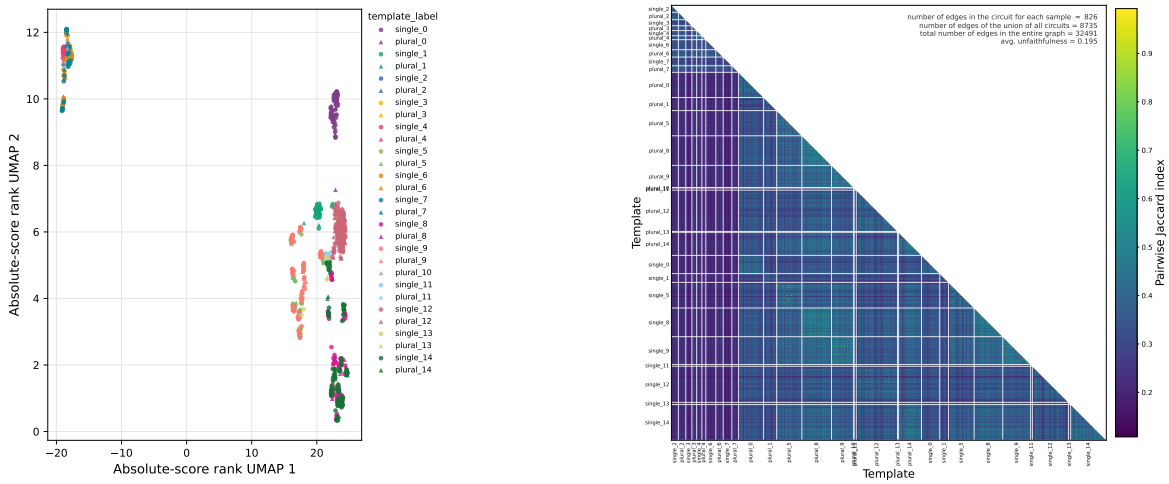
G. Template Absolute-Score Rank UMAP and Sample-Level Circuit Overlap Visualizations

This section pairs absolute-score rank vector UMAP embeddings of sample-level edge scores with sample-by-sample pairwise Jaccard index matrices for the greedily selected circuits. We include GPT-2 small and Pythia-160M on SVA, IOI, and greater-than.

For the PJI matrix visualizations of GPT-2 small and Pythia-160M, we choose graph sizes large enough so that the average unfaithfulness across all samples falls below 0.2. The only exception is GPT-2 small on IOI, where we fix the graph size to 6000 edges (about 20% of all possible edges), which is the maximum graph size we experimented with; even at this size, the unfaithfulness remains above 0.2.

G.1. GPT-2 small

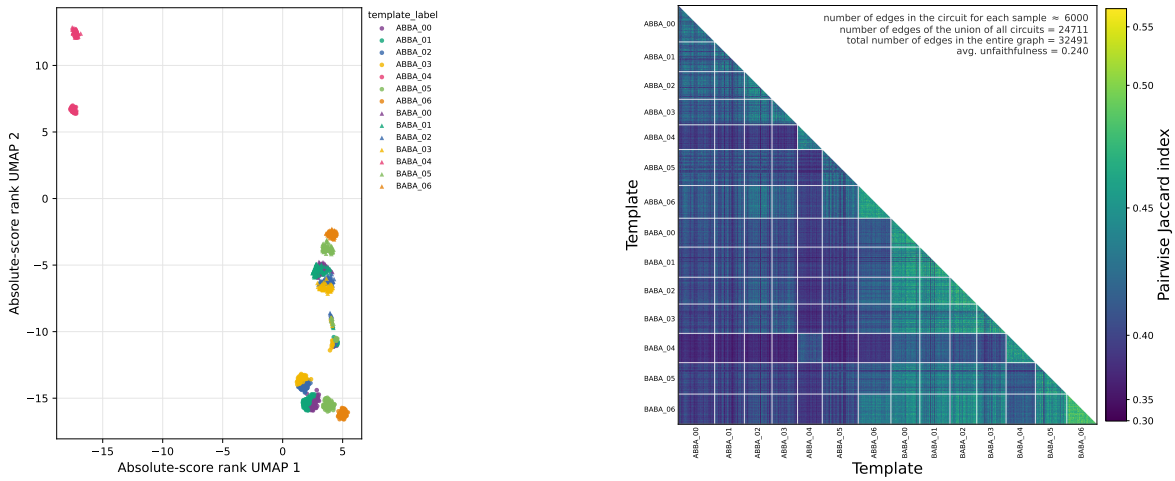
G.1.1. SVA



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 33. Template-induced circuit difference for GPT-2 small on SVA.

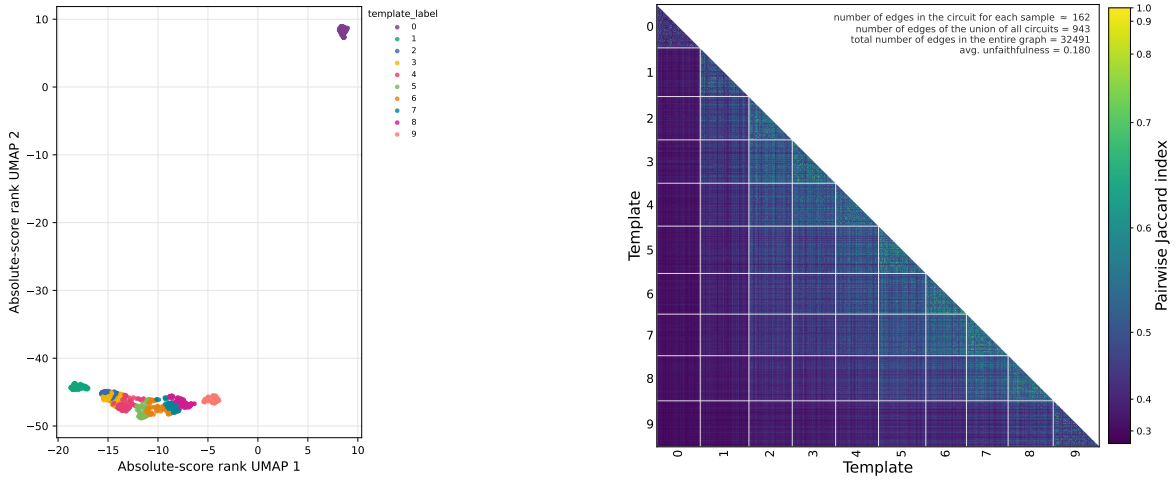
G.1.2. IOI



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 34. Template-induced circuit difference for GPT-2 small on IOI.

G.1.3. GREATER-THAN

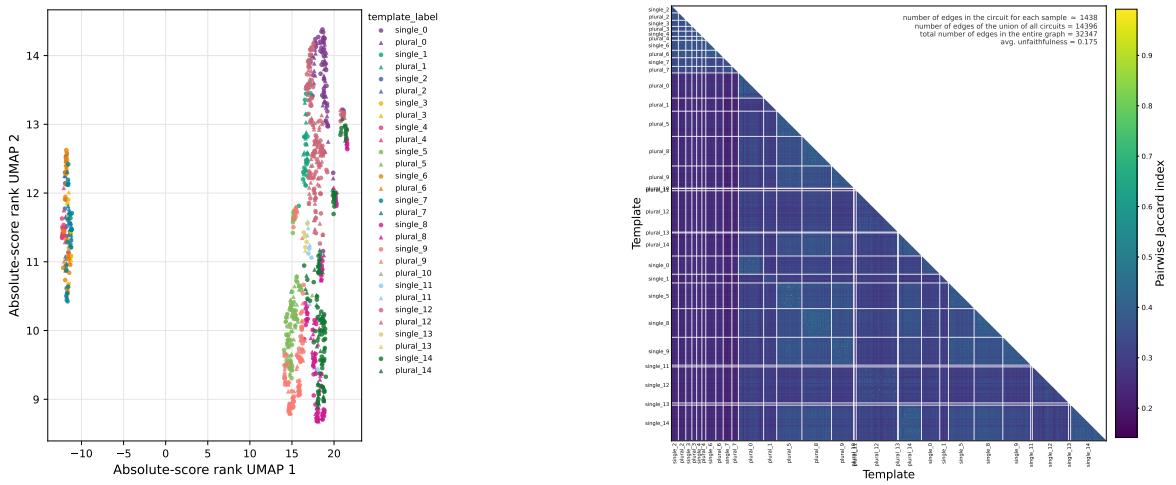


(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 35. Template-induced circuit difference for GPT-2 small on greater-than.

G.2. Pythia-160M

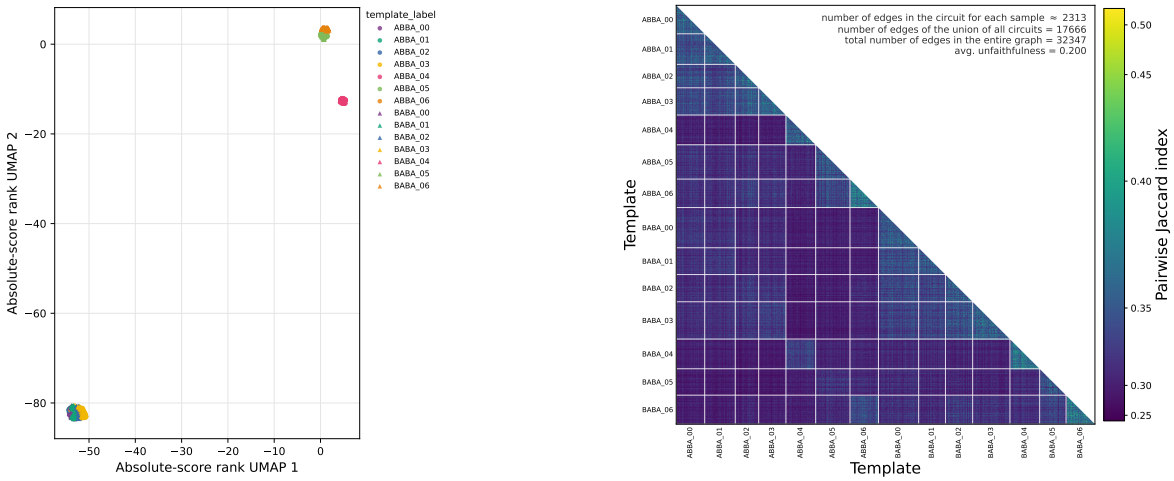
G.2.1. SVA



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 36. Template-induced circuit difference for Pythia-160M on SVA.

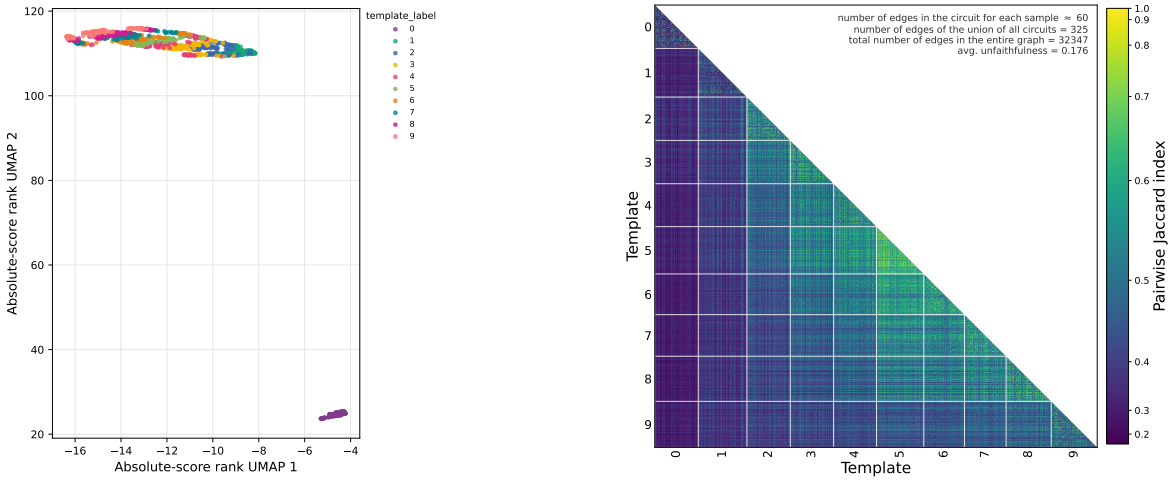
G.2.2. IOI



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 37. Template-induced circuit difference for Pythia-160M on IOI.

G.2.3. GREATER-THAN



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 38. Template-induced circuit difference for Pythia-160M on greater-than.

H. Patching Circuits Found for a Sample Might Act Opposite to Intention on Another Sample

Quantity	Sample 93	Sample 858
Clean prompt	*The senators like to watch television shows and*	*The teachers the parent*
Corrupted prompt	*The senator likes to watch television shows and*	*The teachers the parents*
Clean `prob_diff`	0.352503	0.119733
Own-circuit `prob_diff`	0.336449	0.072026
Cross-circuit `prob_diff`	0.256708	-0.223778

Table 6. Cross-sample circuit patching between samples 93 and 858 of IOI. The own-circuit rows evaluate each sample using the circuit discovered on that sample, whereas the cross-circuit rows evaluate each sample using the circuit discovered on the other sample.

We use the saved per-sample greedy circuits for GPT-2 on the SVA task, with the `prob_diff` metric and target circuit size 826. For a given sample here, the standard evaluation patches activations in the circuit so as to recover the movement from the pro-singular corrupted prompt toward the pro-plural prompt. The cross-sample condition keeps the input pair fixed but replaces the circuit with the circuit discovered on the other sample, thereby testing whether the learned pathway transfers across examples.

Although the two circuits are comparable in size after pruning,[12] with 712 edges for sample 93 and 754 edges for sample 858, their edge-level overlap is small (IoU $= 0.125960$). This low overlap is accompanied by a qualitative failure of transfer: when the circuit from sample 93 is used to patch sample 858, the `prob_diff` moves from 0.119733 under the clean full-model evaluation to -0.223778 under cross-circuit evaluation. Thus the transferred circuit does not merely recover less of the intended effect; it drives the logit movement in the opposite direction from the intended pro-plural intervention.

I. Template Absolute-Score Rank UMAP and Sample-Level Circuit Overlap Visualizations for Models by Gao et al. (2025)

We repeat the visualization for three models open-sourced by Gao et al. (2025):

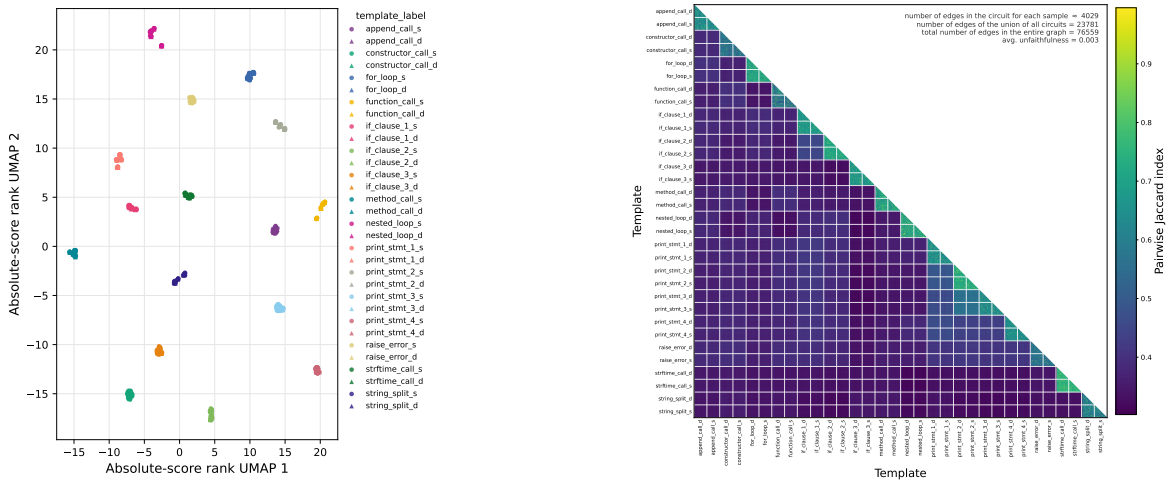
- `dense1_4x`,
- `csp_sweep1_4x_3.7Mnonzero_afrac1.000`,
- `csp_sweep1_4x_3.7Mnonzero_afrac0.500`.

Among these, `dense1_4x` and `csp_sweep1_4x_3.7Mnonzero_afrac1.000` form the closest dense-sparse pair by test loss; the latter has 3.7M nonzero parameters out of 110M total parameters. The difference between the last two models (as indicated by name: `afrac1.000` versus `afrac0.500`) is that the latter imposed activation sparsity on top of weight sparsity. Since the unfaithfulness of all models on both tasks is very low (see Tables 7 and 8) but can still differ by two orders of magnitude, we use circuits containing roughly 5.2% of all graph edges for the PJI plots, rather than choosing an edge count based on unfaithfulness. Tables 7 and 8 shows that cross-template overlap remains low, and the visualizations below show that sparse models still deploy different circuits for different templates.

[12] A procedure implemented by Hanna et al. (2024) to remove headless and tailless branches.

I.1. single-double-quote

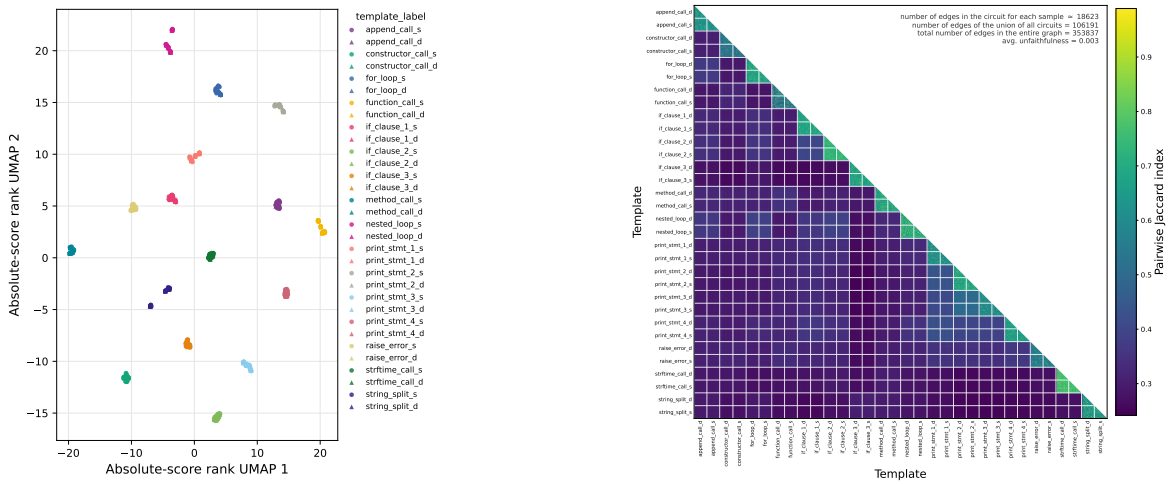
I.1.1. dense1_4x



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 39. Template-induced circuit difference for `dense1_4x` on single-double-quote.

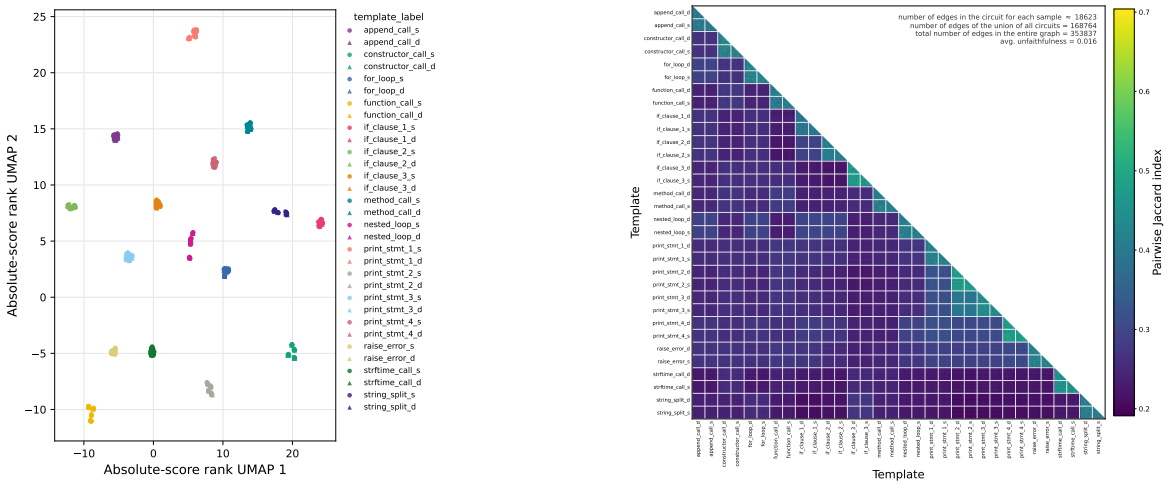
I.1.2. csp_sweep1_4x_3.7Mnonzero_afrac1.000



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 40. Template-induced circuit difference for `csp_sweep1_4x_3.7Mnonzero_afrac1.000` on single-double-quote.

I.1.3. `csp_sweep1_4x_3.7Mnonzero_afrac0.500`

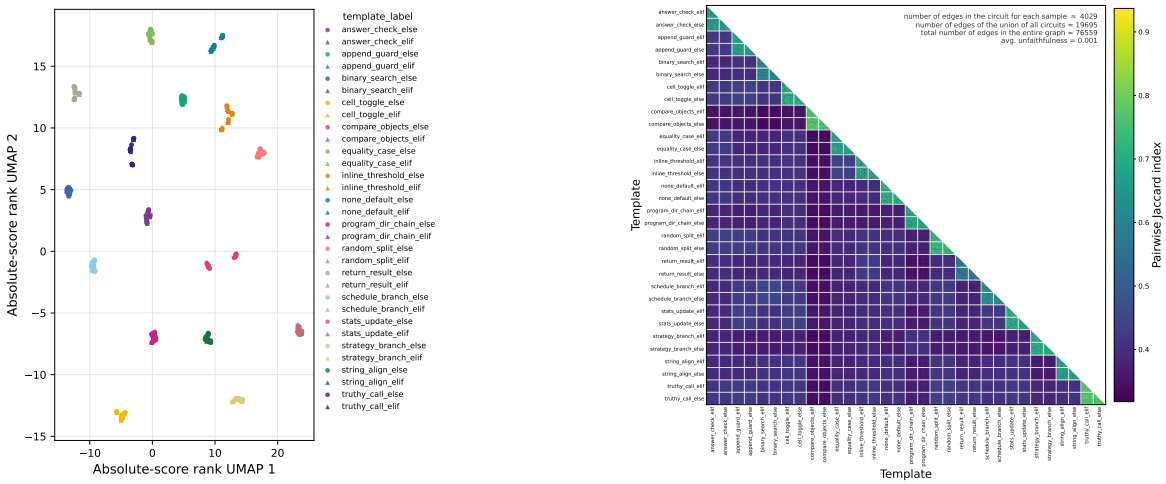


(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 41. Template-induced circuit difference for `csp_sweep1_4x_3.7Mnonzero_afrac0.500` on single-double-quote.

I.2. else-elif

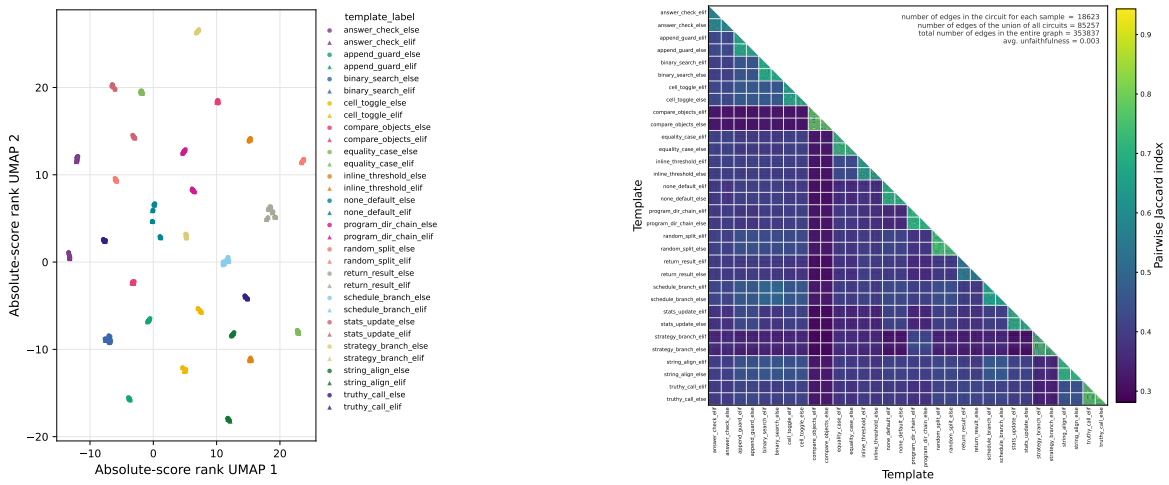
I.2.1. `dense1_4x`



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 42. Template-induced circuit difference for `dense1_4x` on else-elif.

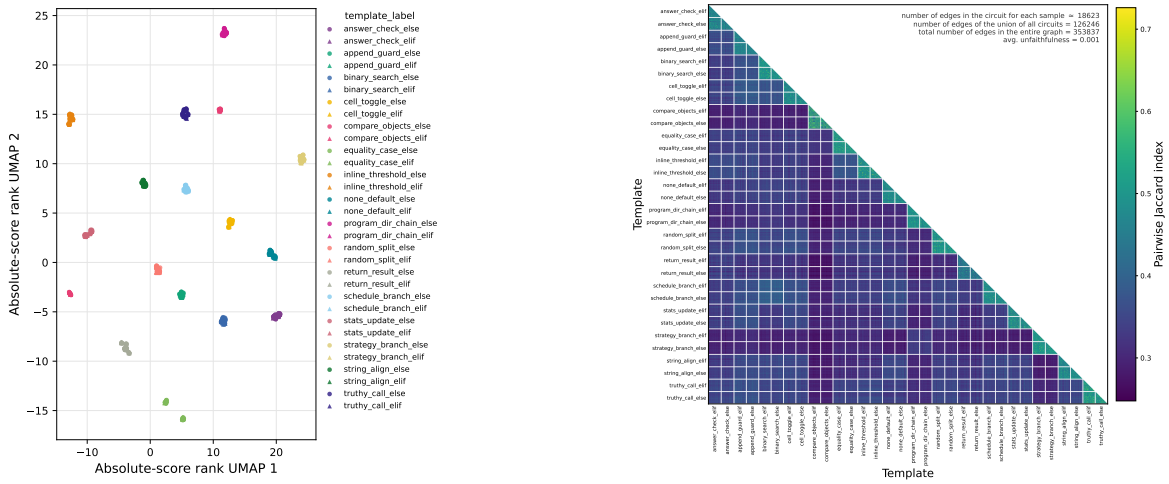
I.2.2. `csp_sweep1_4x_3.7Mnonzero_afrac1.000`



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 43. Template-induced circuit difference for `csp_sweep1_4x_3.7Mnonzero_afrac1.000` on else-elif.

I.2.3. `csp_sweep1_4x_3.7Mnonzero_afrac0.500`



(a) Absolute-score rank UMAP. *(b)* Pairwise Jaccard index.

Figure 44. Template-induced circuit difference for `csp_sweep1_4x_3.7Mnonzero_afrac0.500` on else-elif.

J. Cross-Template Circuit Overlap Summary

Tables 7 and 8 summarize the cross-template sample-level circuit overlap for the dense and circuit-sparsity models. For each model and task, we use circuits containing roughly 5.2% of all graph edges. The PJI average excludes not only the elementwise diagonal, but also pairs of samples from the same normalized template; for example, `A_s` and `A_d` are treated as the same single-double-quote template, and `A_else` and `A_elif` are treated as the same else-elif template. Note that for models with `afrac` less than 1, Gao et al. (2025) use $\mathrm{topk}(\cdot)$ to enforce activation sparsity, which introduces discontinuities into the network. In this setting, conductance can only capture the contribution of each edge along the continuous portions of the computation. Any effect an edge has by causing the output to jump across a discontinuity is not reflected in the conductance score. In principle, this could introduce inaccuracies. However, as demonstrated in the tables below, all circuits achieve low unfaithfulness, providing empirical evidence that conductance remains effective.

Type	Model	Cross-template avg. PJI	Unfaithfulness
sparse	csp_sweep1_4x_0.9Mnonzero_afrac1.000	0.3424	0.0022
	csp_sweep1_4x_1.9Mnonzero_afrac1.000	0.3195	0.0025
	csp_sweep1_4x_3.7Mnonzero_afrac1.000	0.3116	0.0025
	csp_sweep1_4x_7.4Mnonzero_afrac1.000	0.3074	0.0024
	csp_sweep1_4x_14.8Mnonzero_afrac1.000	0.3059	0.0034
	csp_sweep1_4x_3.7Mnonzero_afrac0.062	0.2129	0.0270
	csp_sweep1_4x_3.7Mnonzero_afrac0.125	0.2436	0.0151
	csp_sweep1_4x_3.7Mnonzero_afrac0.250	0.2547	0.0139
	csp_sweep1_4x_3.7Mnonzero_afrac0.500	0.2565	0.0157
	csp_bridges2	0.2642	0.0184
dense	dense1_1x	0.3752	0.0090
	dense1_4x	0.3686	0.0026

Table 7. Cross-template pairwise Jaccard index and unfaithfulness for single-double-quote. Values use circuits containing roughly 5.2% of all graph edges; sample pairs from the same normalized template are excluded from the PJI average.

Type	Model	Cross-template avg. PJI	Unfaithfulness
sparse	csp_sweep1_4x_0.9Mnonzero_afrac1.000	0.4496	0.0004
	csp_sweep1_4x_1.9Mnonzero_afrac1.000	0.4106	0.0159
	csp_sweep1_4x_3.7Mnonzero_afrac1.000	0.3915	0.0034
	csp_sweep1_4x_7.4Mnonzero_afrac1.000	0.3818	0.0014
	csp_sweep1_4x_14.8Mnonzero_afrac1.000	0.3634	0.0008
	csp_sweep1_4x_3.7Mnonzero_afrac0.062	0.3496	0.0026
	csp_sweep1_4x_3.7Mnonzero_afrac0.125	0.3206	0.0101
	csp_sweep1_4x_3.7Mnonzero_afrac0.250	0.3330	0.0006
	csp_sweep1_4x_3.7Mnonzero_afrac0.500	0.3292	0.0009
	csp_bridges2	0.3471	0.0106
dense	dense1_1x	0.4161	0.0320
	dense1_4x	0.3929	0.0013

Table 8. Cross-template pairwise Jaccard index and unfaithfulness for else-elif. Values use circuits containing roughly 5.2% of all graph edges; sample pairs from the same normalized template are excluded from the PJI average.

K. Population-level (Un)faithfulness

Let N denote the set of samples and (x_0^n, x_1^n) denote the corrupted-clean pair for sample $n \in N$. Simplifying the notation from Section 2, we write $M^n := M(x; x_0^n, x_1^n)$, $Q_\Gamma^n := Q_\Gamma(x_0^n, x_1^n)$ for $\Gamma \in \{G, G'\}$ and write the corresponding sample-level faithfulness and unfaithfulness as $\Phi^n := \Phi(G', G; x_0^n, x_1^n) = Q_{G'}^n / Q_G^n$ and $U^n := U(G', G; x_0^n, x_1^n) = |1 - \Phi^n|$.

Following the two notions of population faithfulness discussed by Miller et al. (2024), we note that there can be two types of population unfaithfulness:

Outer expectation: $U_O^N := \frac{1}{|N|} \sum_{n \in N} U^n$.

Inner expectation: The population faithfulness used by Wang et al. (2023); Hanna et al. (2024) is defined as $\Phi_I^N := \frac{\sum_{n \in N} Q_{G'}^n}{\sum_{n \in N} Q_G^n}$. Then the inner expectation form of unfaithfulness naturally follows

$$U_I^N := |1 - \Phi_I^N| = \left| \frac{\sum_{n \in N} (Q_G^n - Q_{G'}^n)}{\sum_{n \in N} Q_G^n} \right|.$$

At first glance, outer expectation appears to be the more reasonable option. Its interpretation as an "average level of unfaithfulness" is straightforward. Nevertheless, we advocate the inner expectation form as the more principled measure, for the following two reasons.

First, the current circuit discovery algorithms score the edges by averaging (or, equivalently, summing) the scores across all the samples (Hanna et al., 2024; Franco et al., 2026; Méloux et al., 2025). For gradient-based methods, by the linearity of the gradient, this approach is equivalent to substituting the metric M in Equation (2) with $\sum_{n \in N} M^n$. This implies that the network behavior accounted for is the summation of behavior over all the samples $\sum_{n \in N} Q^n$ when the model is confronted with all corrupted-clean pairs in N, analogous to the sample-wise scoring of Equation (2) accounting for the behavior of Q_G. From this perspective, Φ_I^N and U_I^N are more apt metrics to measure the quality of circuit discovery, since they directly account for the summation in their form. Admittedly, one can instead substitute M with the weighted sum $\sum_{n \in N} \frac{M^n}{Q_G^n}$, where Q_G^n is treated as a constant, in Equation (2), so that the behavior accounted for by the scoring is more aligned with the outer expectation form. Nevertheless, we also have the following reason to further advocate the inner expectation form.

The outer expectation treats the unfaithfulness of all samples equally. However, as we see in Section 5.2, the high unfaithfulness of samples with small $|Q_G^n|$ is an artifact of the metric itself and does not indicate major circuit defects. Hence, we should be more forgiving of the unfaithfulness of samples with small $|Q_G^n|$, which the inner expectation form captures. The inaccuracy of each n caused by G' is factored into this form through $(Q_G^n - Q_{G'}^n)$: If two samples contribute the same $(Q_G^n - Q_{G'}^n)$ to the inner expectation, then the sample with smaller $|Q_G^n|$ would have much worse U^n, which reflects a higher level of tolerance for it. In contrast, the outer expectation form might yield results that are drowned out by the extreme unfaithfulness values of samples with small $|Q_G^n|$.

L. The Defect of Unfaithfulness for a Single Sample

Unfaithfulness is a metric intended to capture how differently the chosen circuit behaves from the whole model. Nonetheless, the difference could lie in multiple aspects. Marr (2010) proposed three levels of analysis: computational, algorithmic, and implementational. The implementational level concerns the hardware-specific mechanisms to realize AI models and is largely irrelevant to interpretability. The algorithmic level characterizes the program that the model uses to handle tasks (Wang et al., 2023; Franco et al., 2026). While it is most appealing to characterize unfaithfulness at this level, there is currently no unified framework for doing so. The closest proxy for this is completeness (Wang et al., 2023), but it is computationally intractable. Thus, we are left with characterizing unfaithfulness at the computational level—that is, how much the input-output function implemented by the circuit over the relevant region of the input space (which defines a given task) deviates from the corresponding function implemented by the full model. Achieving low computational unfaithfulness is a necessary condition for attaining low algorithmic unfaithfulness (Marr, 2010). In this light, the form of unfaithfulness we investigate (computational unfaithfulness) should be viewed as only a crude surrogate for the genuine notion of unfaithfulness (algorithmic unfaithfulness). This provides us with a rationale to refine our notion of unfaithfulness.

Within computational unfaithfulness, there is also the difference between single-sample computational unfaithfulness and functional computational unfaithfulness. In terms of a single sample, the agreement between the full model's output for it and that of the selected circuit provides very limited information regarding how similar the underlying algorithms implemented by the full model and the circuit actually are. The circuit could instantiate a vastly different function from the full model, only coinciding in value at the point of the sample itself (see Figure 45). As we include more edges into the circuit, the circuit function evaluated at the sample could deviate from the full model, but conceptually, the underlying algorithm is more developed with more edges and thus the function as a whole should be closer to the full model function. Hence, it is reasonable to adopt the pessimistic unfaithfulness as a more canonical notion of single sample unfaithfulness to enforce monotonicity, which reflects the underlying improvement of algorithmic faithfulness as more edges are added.

On the other hand, population unfaithfulness involves many points in the input space, so it is more informative as a measure of unfaithfulness at the function level and already better satisfies the desideratum of monotonicity, which can be observed in Appendix F. This explains the efficacy of (un)faithfulness in numerous previous works (Hanna et al., 2024; Wang et al., 2025a; Méloux et al., 2025).

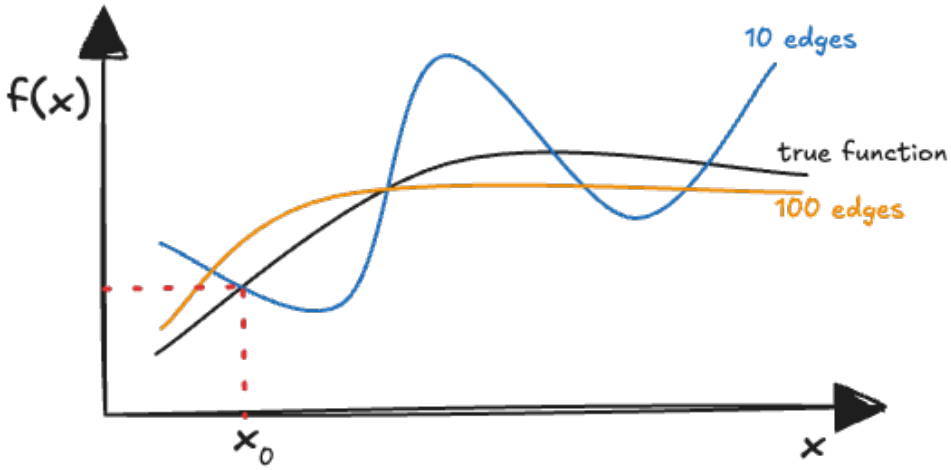


Figure 45. A schematic of why single sample faithfulness is highly non-monotonic. Suppose we apply circuit discovery to recover the true function (which is the prerequisite for achieving algorithmic faithfulness), while the sample of interest is at x_0. When we include 10 edges, the circuit function might be quite far away from the true function, but they coincide at x_0. With more edges included, the circuit function is closer to the true function as a whole, but it may deviate further from the true function at x_0.

M. Extended Correlation Diagnostics Across Templates, Models, and Circuit Sizes

We extend the six rank-correlation diagnostics shown in Figures 6 and 7 from one GPT-2 small SVA template and one circuit size to all templates of the SVA/IOI/greater-than tasks, GPT-2 small and Pythia-160M, and the full greedy edge-count sweep. Solid curve segments indicate statistically significant Spearman rank correlations at $p < 0.05$; dotted segments indicate non-significant correlations. We present the diagnostics in the same order as the corresponding panels in the main text.

When the curves are mostly above zero, it indicates a positive correlation; otherwise, a negative correlation. There are many curve segments that are not statistically significant. This is because certain templates reach low unfaithfulness easily. Once all the samples in one template are highly faithful, the trends that involve unfaithfulness will disappear. For example, if we include all edges in the graph, then all unfaithfulness will be zero, and the heavy-tailedness of s^n will have no effect on it. However, this does not nullify our explanation, which aims mainly at explaining cases where extremely poor unfaithfulness scores exist.

Note that U'/U graphs are, in general, noisier and less statistically significant than $\bar{U}'/\bar{U}$. However, the general trend does not change.

M.1. Individual Scoring

M.1.1. FIGURE 7(B): $\bar{U}/U$ VS. $|Q_G|$

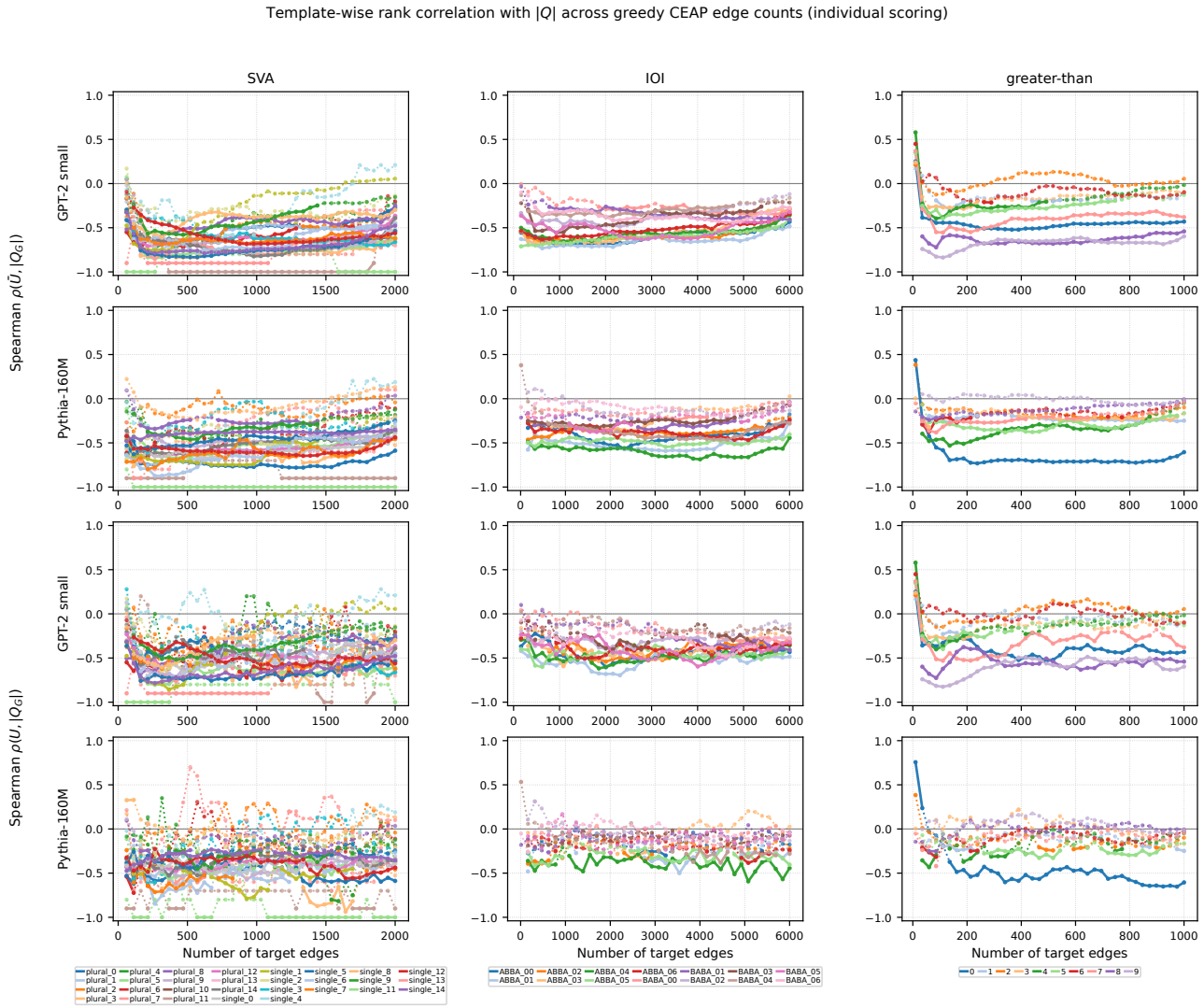


Figure 46. Extension of the $\bar{U}/U$ vs. $|Q_G|$ diagnostic. All the curves that are statistically significant are below zero, except in a few cases where the circuit is too small for the unfaithfulness to be informative yet. Moreover, if we constrain the circuit to be of a small size, the circuit discovery algorithm needs to accommodate more for the connectivity of the subgraph, rather than picking the edges with higher score absolute values, which deviates from the mental image of Figure 8.

M.1.2. FIGURE 7(C): $\bar{U}'/U'$ VS. $|Q_G|$

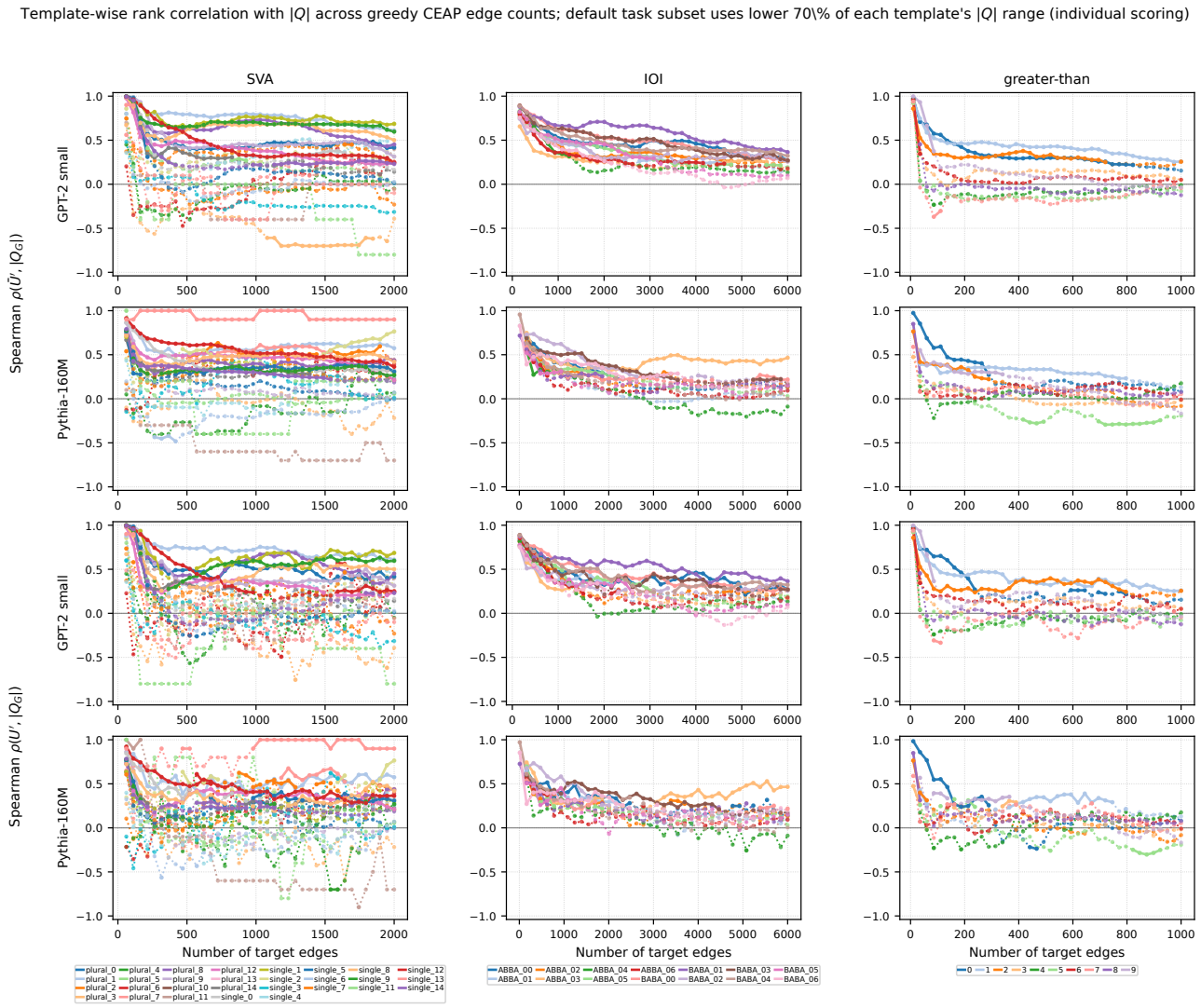


Figure 47. Extension of the unnormalized $\bar{U}'/U'$ vs. $|Q_G|$ diagnostic. It generally shows positive correlation, with a few exceptions, such as `plural_3`. There, we have only 11 samples, and the correlation readings may be unreliable. Greater-than may show negative correlations here, because we see that $\bar{U}'/U'$ normally first goes up with $|Q_G|$, then goes down. To account for this, for each template we remove all points beyond the first 70% of that template's $|Q_G|$ range. Note that this does not contradict our main point that points with small $|Q_G|$ in general do not have large $\bar{U}'/U'$.

M.1.3. FIGURE 9(A): μ VS. $|Q_G|$

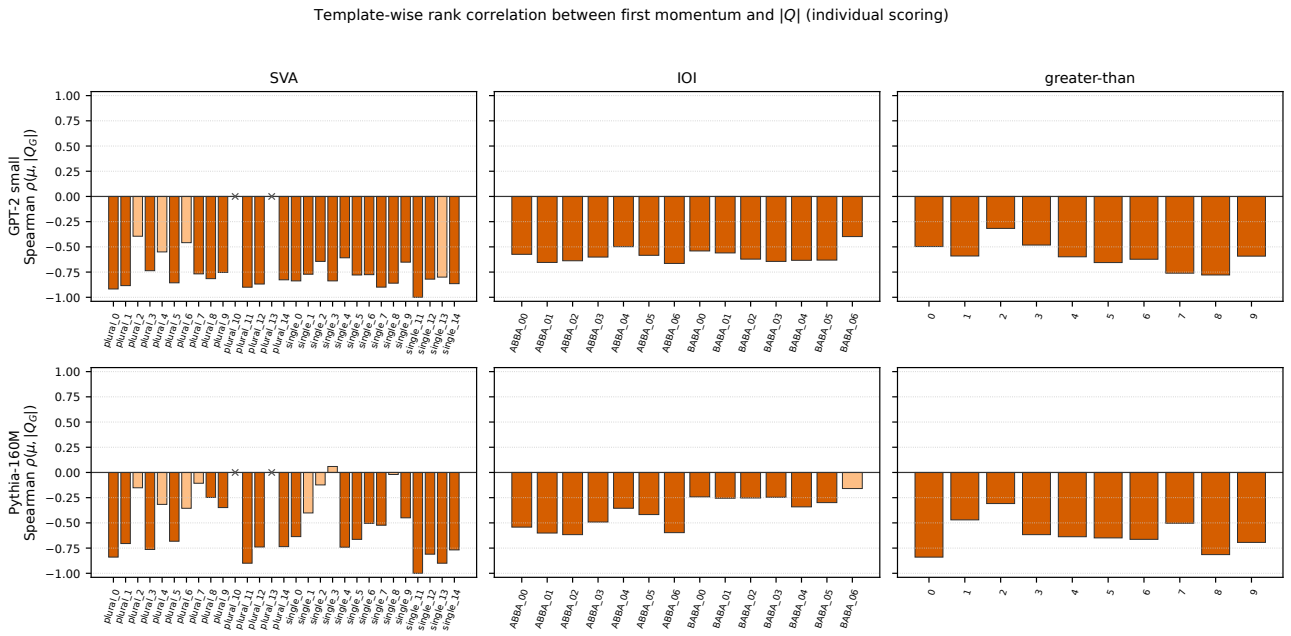


Figure 48. Extension of the μ vs. $|Q_G|$ diagnostic. There is a clearly negative correlation in general. A darker color suggests the Spearman ρ is statistically significant, and a lighter color suggests otherwise. The x's here indicate that there are too few samples for those templates in the original dataset and we did not get any data for them when sampling the dataset for our probing here.

M.1.4. FIGURE 9(B): R VS. μ

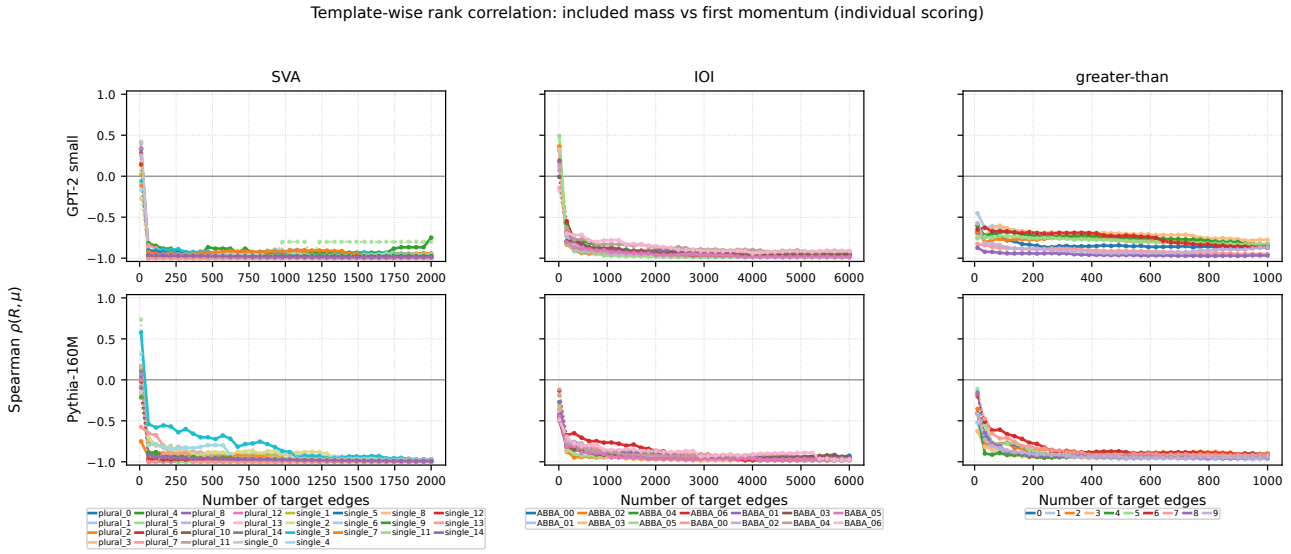


Figure 49. Extension of the R vs. μ diagnostic. There is a clearly negative correlation in general.

M.1.5. FIGURE 9(C): $\bar{U}/U$ VS. R

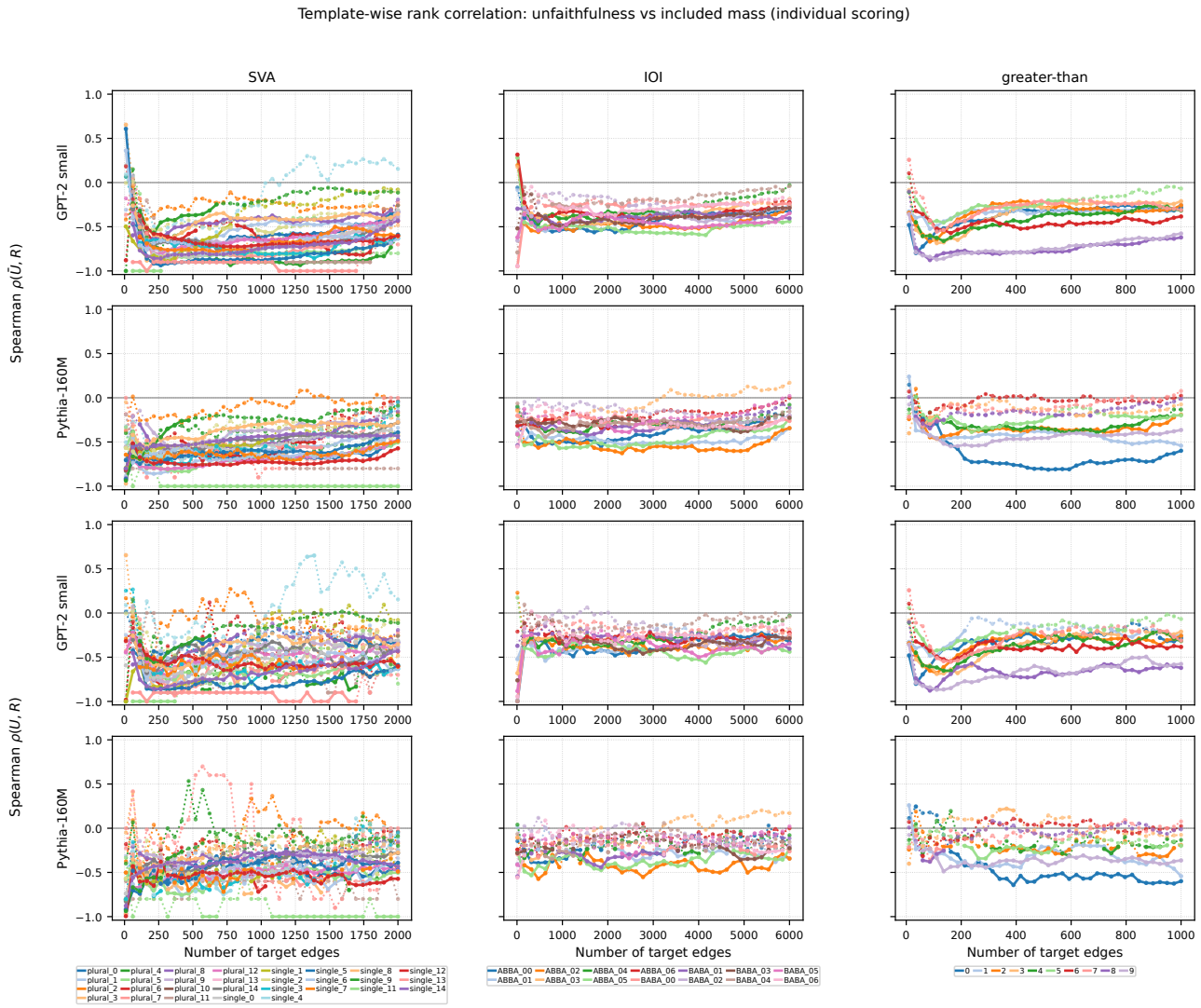


Figure 50. Extension of the $\bar{U}/U$ vs. R diagnostic. There is a clearly negative correlation in general.

M.1.6. FIGURE 9(D): $\bar{U}/U$ VS. μ

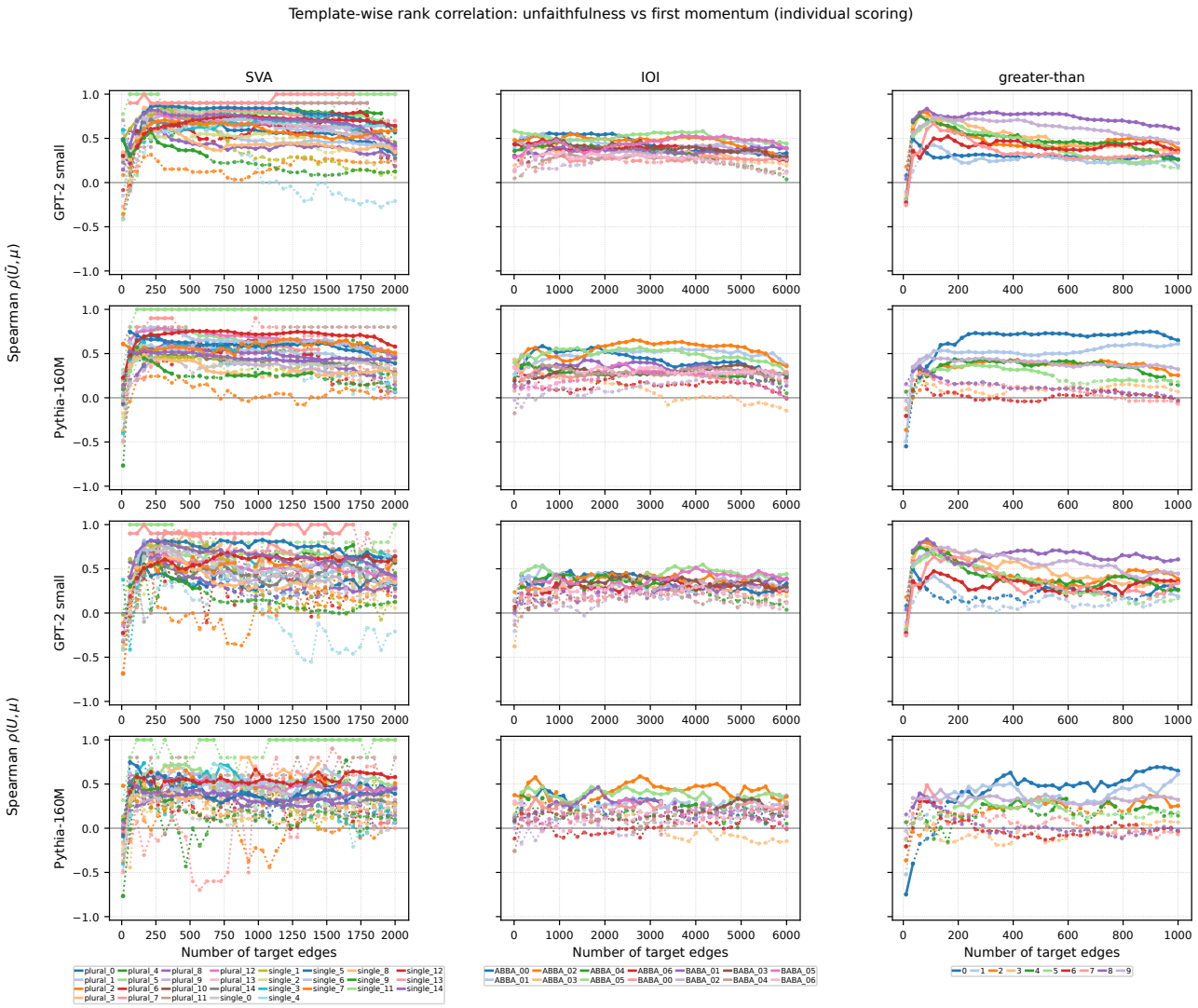


Figure 51. Extension of the $\bar{U}/U$ vs. μ diagnostic. There is a clearly positive correlation in general.

M.2. Template-averaged Scoring

All figures follow the trend in the last section, with a few exceptions for greater-than. We believe this is due to the diffuse nature of the greater-than circuits. As shown in Appendix G, the circuits underlying templates 1 through 9 form a smooth continuum rather than cleanly separated clusters, meaning that circuits within these templates are not particularly similar. Consequently, the results from the previous section provide only limited predictive power for this setup, because we require the greedy circuit discovery to approximately select the edges with the top absolute value scores for the mental image of Figure 8 to work. If the circuit is chosen via averaging across circuits that are not similar, then this assumption does not hold.

M.2.1. FIGURE 5(A): U DISTRIBUTION

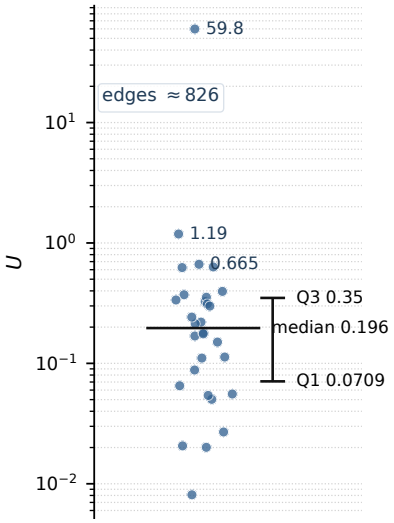


Figure 52. Template-averaged scoring version of the U distribution diagnostic for GPT-2 small on SVA template `plural_1`.

M.2.2. FIGURE 7(B): $\bar{U}/U$ VS. $|Q_G|$

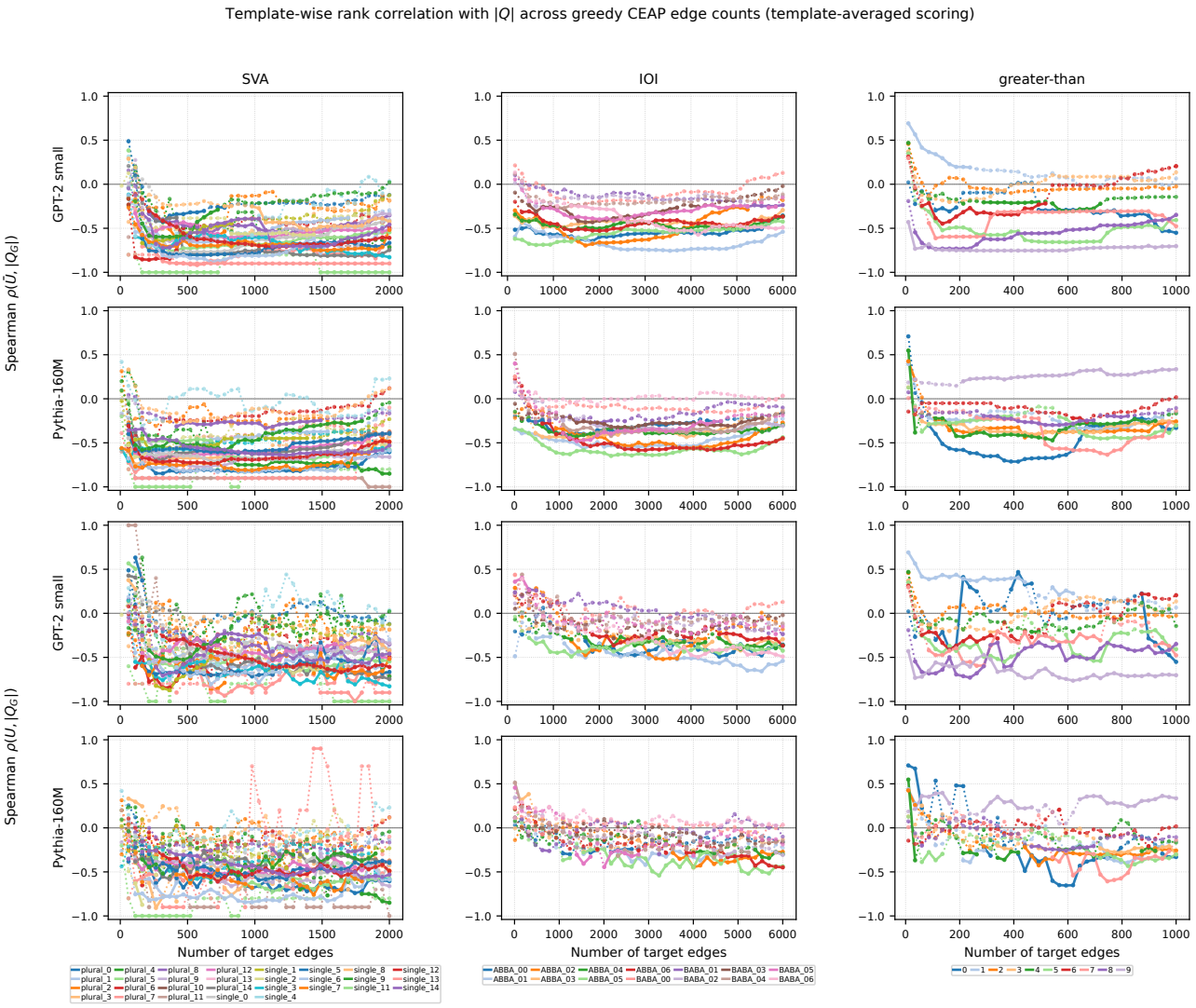


Figure 53. Template-averaged scoring version of the $\bar{U}/U$ vs. $|Q_G|$ diagnostic.

M.2.3. FIGURE 7(C): $\bar{U}'/U'$ VS. $|Q_G|$

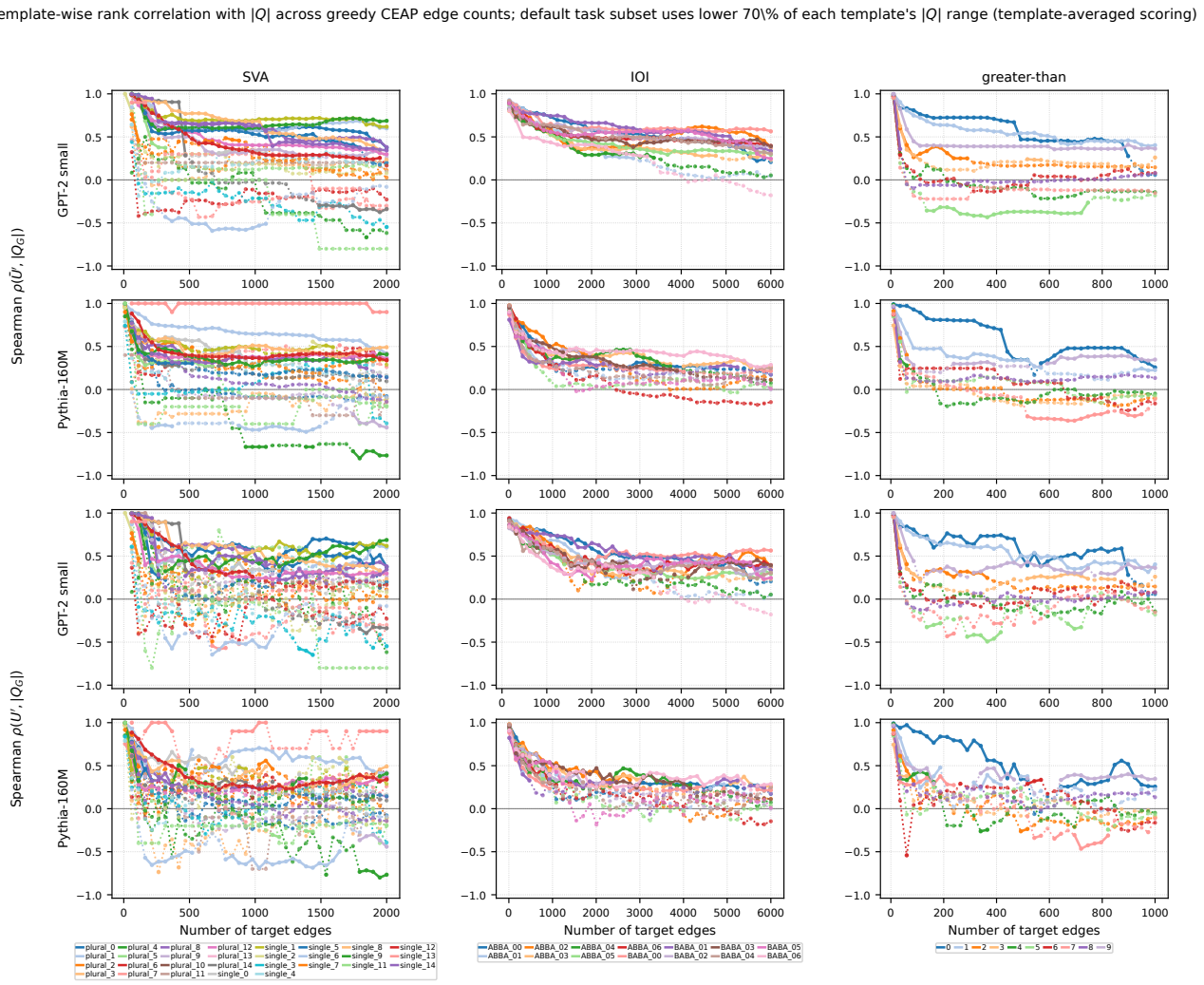


Figure 54. Template-averaged scoring version of the unnormalized $\bar{U}'/U'$ vs. $|Q_G|$ diagnostic.

M.2.4. FIGURE 9(A): μ VS. $|Q_G|$

This plot is exactly the same as Figure 48, as it reflects a property of each sample and does not depend on whether we average over all samples.

M.2.5. FIGURE 9(B): R VS. μ

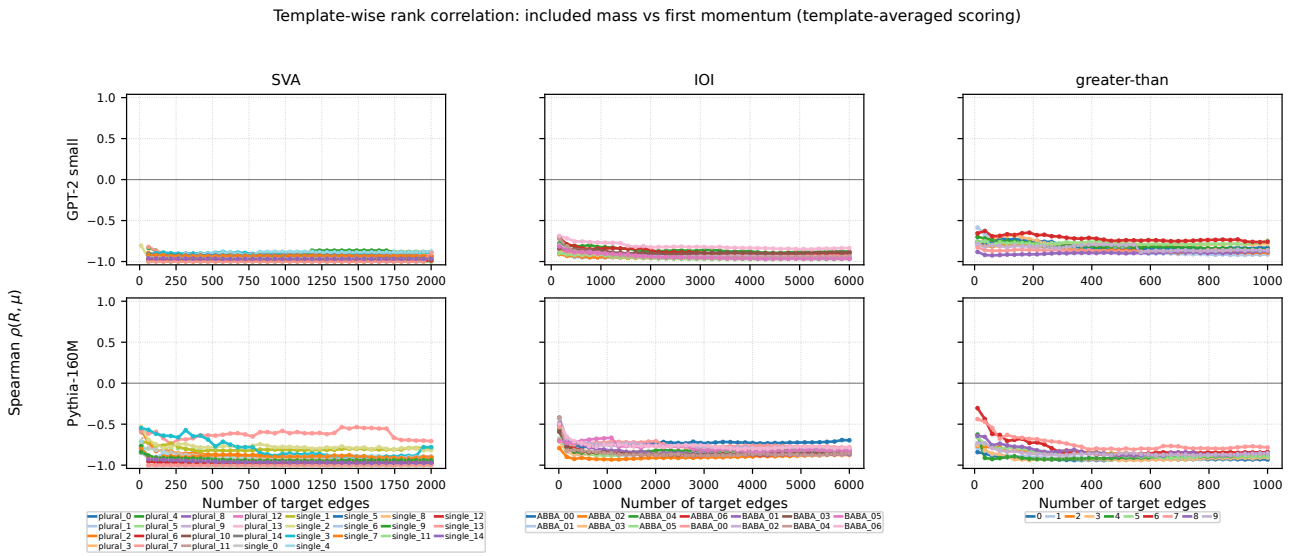


Figure 55. Template-averaged scoring version of the R vs. μ diagnostic.

M.2.6. FIGURE 9(C): $\bar{U}/U$ VS. R

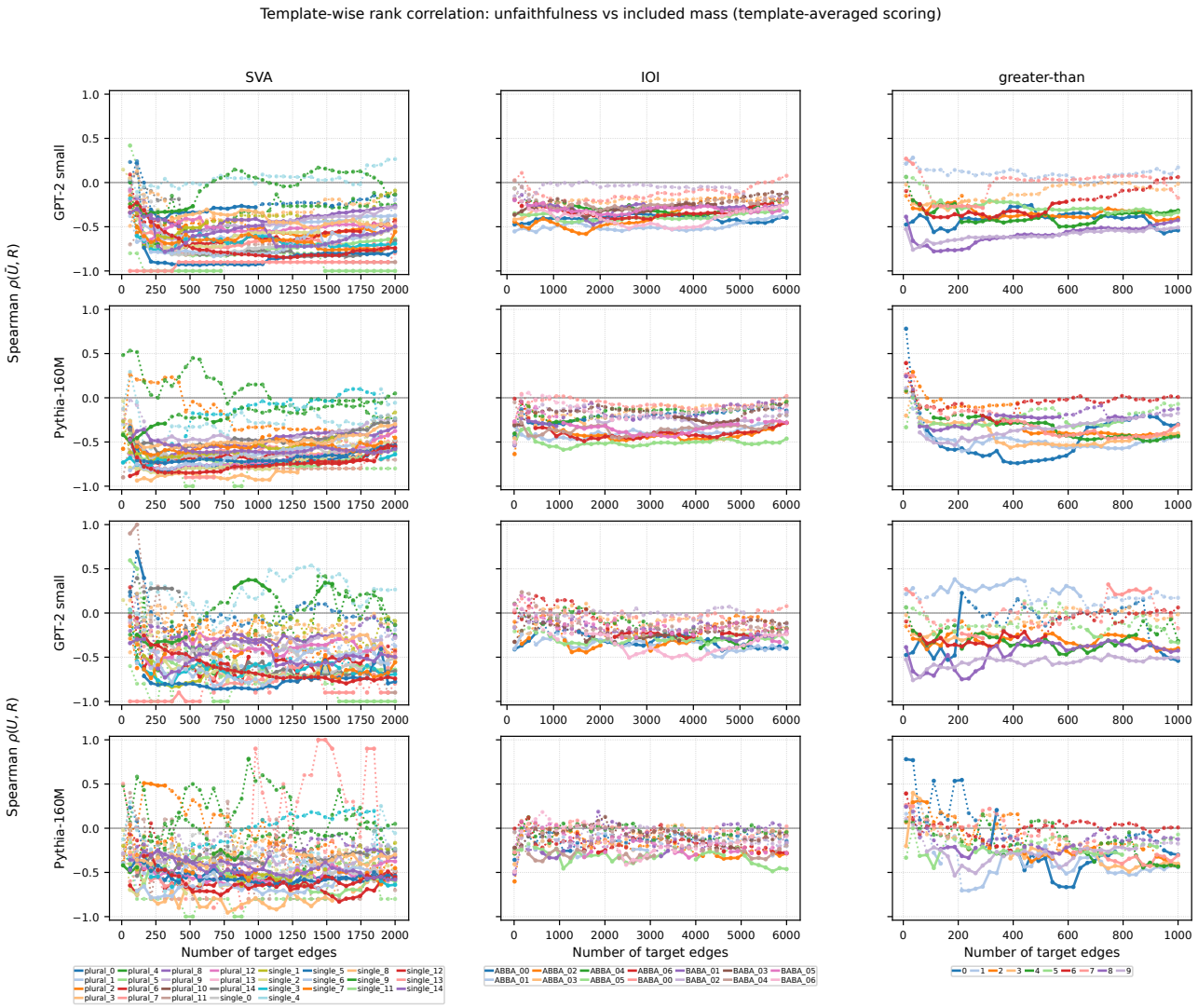


Figure 56. Template-averaged scoring version of the $\bar{U}/U$ vs. R diagnostic.

M.2.7. FIGURE 9(D): $\bar{U}/U$ VS. μ

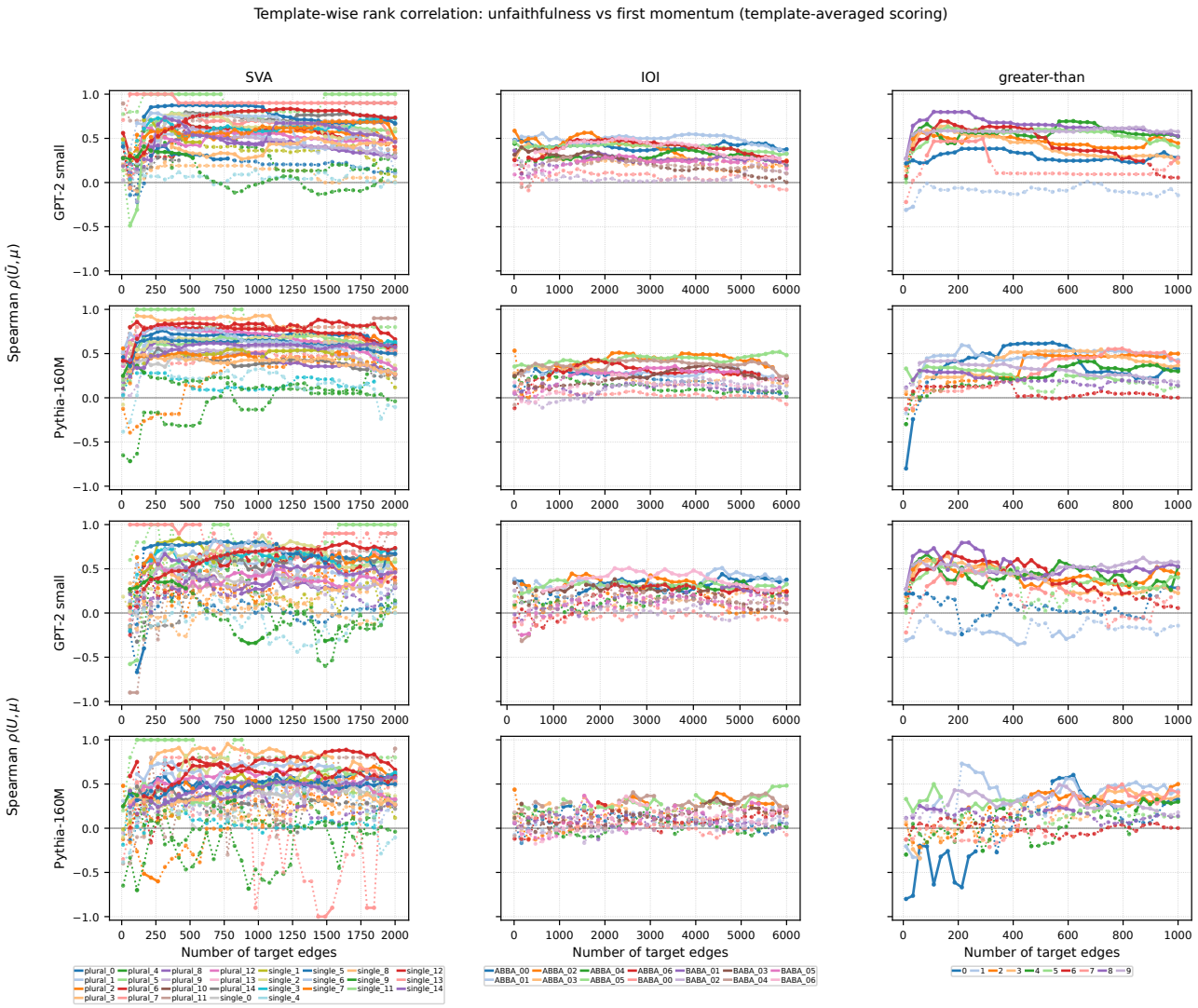


Figure 57. Template-averaged scoring version of the $\bar{U}/U$ vs. μ diagnostic.

N. Substantiate the Mental Image of Figure 8 with Two Real Samples

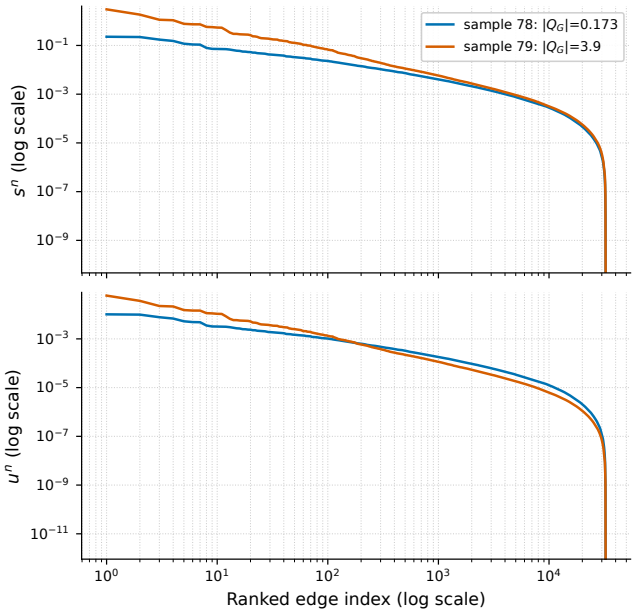


Figure 58. Redraw Figure 8 with two real samples of the same template. Sample 78 is the one with the poorest U reading in Figure 6a, and sample 79 is a sample of the same template but much higher $|Q_G|$. The curves are shown on logarithmic axes to make both the high-score region and long tail visible.