RETHINKING CIRCUIT DISCOVERY WITH LEARNABLE CORRUPTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Patching-based circuit discovery has a hidden degree of freedom: the corruption that defines the behavioral contrast. Existing methods fix this corruption and optimize only the sparse circuit, even though a patching score is not a property of a circuit alone, but of a circuit under a chosen behavioral contrast. We make this dependence explicit by recasting circuit discovery as a joint optimization over sparse circuits and reference-anchored admissible corruptions. In our reformulated problem, the benchmark reference is retained as an anchor for constructing semantically admissible and behaviorally disruptive counterfactual families, with fixed-corruption discovery recovered as a boundary case. We instantiate this formulation with SEED (**S**tructure **E**dit and **E**dge **D**iscovery), a tractable alternating procedure that updates the corruption coordinate under the current circuit and updates a shared sparse circuit under the resulting contrasts. Across eight circuit-discovery tasks, under both denoising (sufficiency) and noising (necessity) discovery objectives and with both Faithfulness and Completeness measured on held-out data, learning corruptions improves over optimizing the circuit alone, although the gains are task-dependent. Yet the corruptions selected by SEED do not transfer uniformly to non-SEED methods, indicating that they behave as circuit-coupled best responses rather than universally better corruptions. These results suggest that circuit discovery should be understood as a coupled problem over circuits and behavioral contrasts.

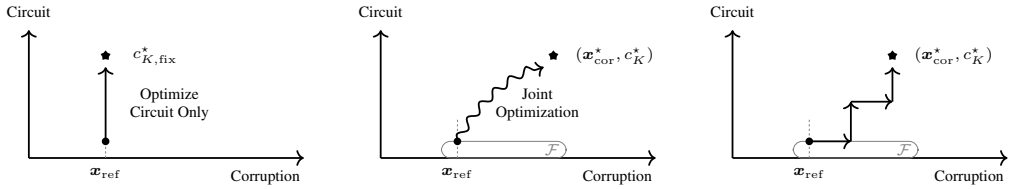
1 INTRODUCTION

Patching-based circuit discovery (Wang et al., 2022; Hanna et al., 2023; Heimersheim & Janiak, 2023; Conmy et al., 2023; Syed et al., 2023; Hanna et al., 2024) is a central tool in mechanistic interpretability (Olah et al., 2020; Olsson et al., 2022; Meng et al., 2022; Nanda et al., 2023). Given a clean input and a corrupted counterpart, it restores activations of a candidate subnetwork from the clean run into the corrupted run and measures how much clean task behavior is recovered (Goldowsky-Dill et al., 2023; Zhang & Nanda, 2023; Heimersheim & Nanda, 2024), a common operational proxy for circuit faithfulness (Conmy et al., 2023; Syed et al., 2023; Hanna et al., 2024).

Existing methods largely follow a fixed-corruption paradigm (Conmy et al., 2023; Syed et al., 2023; Hanna et al., 2024; Bhaskar et al., 2024; Kram'ar et al., 2024): the benchmark specifies the corruption, and the discovery algorithm optimizes only the sparse circuit. It is well recognized that patching results depend on the chosen corruption (Zhang & Nanda, 2023; Heimersheim & Nanda, 2024), yet this dependence is usually treated as a design decision made before discovery. The patching score depends on two coupled choices: the circuit, which controls where clean information enters the corrupted computation, and the corruption, which controls which behavioral distinction is tested; when several admissible corruptions exist, changing the corruption can change recovery profiles, circuit rankings, and even the circuit selected under a fixed edge budget. Fixed-corruption discovery therefore solves a one-coordinate slice of a coupled problem.

We make this coupling explicit by formulating circuit discovery as a joint problem over a sparse circuit and a behavioral contrast (Figure 1). We use the benchmark corruption as an anchor for a family of reference-anchored admissible corruptions, which must remain behaviorally disruptive relative to the clean input to rule out degenerate choices, and seek a shared sparse circuit together with

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



(a) Circuit Discovery with Fixed Corruptions **(b) Corruption-aware Circuit Discovery** **(c) SEED: Alternating Optimization**

Figure 1: **From fixed-corruption to corruption-aware circuit discovery.** (a) Fixed-corruption discovery optimizes only the circuit under a benchmark reference corruption. (b) Corruption-aware discovery jointly searches over reference-anchored admissible corruptions and sparse circuits. (c) SEED approximates this coupled search by alternating Structure Edit and Edge Discovery.

per-example admissible corruptions that expose informative contrasts; the standard fixed-corruption protocol is the boundary case in which the family contains only the benchmark corruption.

We instantiate this formulation with SEED (**S**tructure **E**dit and **E**dge **D**iscovery), a tractable alternating procedure on frozen models: *Structure Edit* reweights reference-anchored candidate corruptions under the current circuit, and *Edge Discovery* updates shared edge logits whose hard top-K decode gives the sparse circuit. SEED is one instantiation of the formulation built from standard components; its default objective is denoising (sufficiency; Section 4), and we also instantiate a noising (necessity) objective. The optimizer-matched control SEED-EDGE optimizes only the circuit, with the benchmark corruption fixed.

We evaluate on eight circuit-discovery tasks with frozen `gpt2-small` under a matched protocol (shared graph, data splits, edge budgets, and held-out benchmark corruptions), scoring every circuit by Faithfulness (sufficiency) and Completeness (necessity). Under both objectives, learning corruptions improves over optimizing the circuit alone: with the denoising objective, Faithfulness rises from 0.555 to 0.593 and Completeness from 0.921 to 0.951; with the noising objective, Completeness rises from 0.966 to 0.975. The gains are task-dependent and largest when the admissible family contains several informative contrasts, as in IOI. SEED with the denoising objective also attains the highest average Faithfulness among all methods and exceeds the learned-mask baseline Edge Pruning (Bhaskar et al., 2024) in both metrics. Finally, corruptions selected by SEED do not transfer uniformly to other discovery methods, consistent with learned contrasts being coupled to the circuit. Our contributions are as follows.

- **Formalization.** We formalize patching-based circuit discovery as joint optimization over a sparse circuit and a reference-anchored admissible corruption, with fixed-corruption discovery as a boundary case.
- **Algorithm.** We propose SEED, a tractable alternating algorithm that jointly optimizes the circuit and learns the corruption, for both denoising and noising objectives.
- **Empirical finding.** Under both objectives, and on held-out data with both Faithfulness and Completeness, learning corruptions is overall better than optimizing the circuit only.

2 Related Work

Mechanistic interpretability studies how model behavior arises from internal *representations* (Meng et al., 2022; Belrose et al., 2023; Newman et al., 2021; Vig et al., 2020; Chintam et al., 2023) or sparse *circuits* (Elhage et al., 2021; Olah et al., 2020; Olsson et al., 2022); we follow the intervention-based view, where causal involvement is tested by patching activations or paths (Goldowsky-Dill et al., 2023; Zhang & Nanda, 2023; Heimersheim & Nanda, 2024). Circuits are found either by top-down analyses of tasks such as indirect object identification (Wang et al., 2022), docstring completion (Heimersheim & Janiak, 2023), and greater-than reasoning (Hanna et al., 2023), or by automated search, including greedy patching-based pruning (ACDC; Conmy et al., 2023), attribution-based edge scores (EAP, EAP-IG, EAP-GP; Nanda, 2023; Hanna et al., 2024; Zhang et al., 2025), learned masks and edge pruning, and feature-level causal graphs (Bhaskar et al., 2024; Kram'ar et al., 2024; Marks et al., 2024). Closest to our work (Appendix C), optimal ablation (Li & Janson, 2024) optimizes the replacement activation of an ablated component, whereas we keep interventions at the input level and optimize over admissible counterfactual inputs anchored to the benchmark corruption; sheaf

discovery (Yu et al., 2025) jointly prunes computation-graph edges and weights with a combined faithfulness and completeness objective, changing the circuit objective rather than the contrast, which is complementary. Circuit completeness and uniqueness are also debated: single-direction patching can miss AND- or OR-structured components (Heimersheim & Nanda, 2024; Chen et al., 2025), faithfulness scores are sensitive to the ablation protocol (Miller et al., 2024), and multiple circuits can explain the same behavior (Méloux et al., 2025); so we evaluate circuits in both directions and treat the output of SEED as a sufficient but not unique explanation under an admissible contrast.

3 Rethinking Circuit Discovery

In this section, we revisit circuit discovery from a unified perspective that makes the role of corruptions explicit. Existing approaches to circuit discovery fix the corruption and optimize only over circuits. However, this design choice implicitly ties circuit quality to a particular corruption, rather than the underlying task behavior. To address this, we first formulate this dependence in terms of corruptions, recovery, and circuit structure. We then extend the discovery problem to allow reference-anchored corruption choices, yielding a corruption-aware formulation that jointly reasons about the behavioral contrast and the sparse circuit recovered under it.

3.1 Problem Setup

Let $\mathcal{M}$ be a frozen language model, whose forward computation on input $\boldsymbol{x}$ is represented as a directed graph $G = (V, E)$. A circuit is a subset of edges $c \subseteq E$, and we consider the budget-constrained class $\mathcal{C}_K \triangleq \{c \subseteq E : |c| \leq K\}$. We draw clean–reference pairs $(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}) \sim P$, where $\boldsymbol{X}_{\text{clean}}$ is the clean input and $\boldsymbol{X}_{\text{ref}}$ is the benchmark-provided reference corruption. We also consider a general counterfactual corruption random variable $\boldsymbol{X}_{\text{cor}}$, not necessarily sampled from P. Throughout, we use lowercase symbols $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}})$ to denote realizations of the corresponding random variables. Let $m(\cdot)$ denote the task score. Then for any clean input and corruption pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})$, we define the clean-corruption gap as $\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}) \triangleq m(\boldsymbol{x}_{\text{clean}}) - m(\boldsymbol{x}_{\text{cor}})$, which measures how strongly the corruption disrupts model behavior.

Given a circuit c, denoising patching evaluates $\boldsymbol{x}_{\text{cor}}$ while restoring the activations along edges in c from the clean run. This yields a patched score $s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c)$, formally defined in Appendix D, which quantifies how much of the task behavior can be recovered by the circuit.

We restrict attention to *valid* corruptions that induce a non-trivial performance gap and ensure that patching interpolates between corrupted and clean behavior.

Assumption 3.1 (Validity)**.** There exists a constant $\delta_\Delta > 0$ such that every clean–reference pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ and every valid counterfactual corruption $\boldsymbol{x}_{\text{cor}}$ associated with $\boldsymbol{x}_{\text{clean}}$ satisfy

$$\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \geq \delta_\Delta, \qquad \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}) \geq \delta_\Delta.$$

Moreover, for every circuit $c \subseteq E$ considered by the discovery objective, the patched score lies between the corrupted and clean full-model scores:

$$m(\boldsymbol{x}_{\text{ref}}) \leq s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}; c) \leq m(\boldsymbol{x}_{\text{clean}}), \quad m(\boldsymbol{x}_{\text{cor}}) \leq s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c) \leq m(\boldsymbol{x}_{\text{clean}}).$$

For valid corruptions, we measure circuit quality using normalized recovery

$$\tau(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c) \triangleq \frac{s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c) - m(\boldsymbol{x}_{\text{cor}})}{m(\boldsymbol{x}_{\text{clean}}) - m(\boldsymbol{x}_{\text{cor}})}. \tag{1}$$

Under Assumption 3.1, this score lies in $[0, 1]$ and reflects how well the circuit restores clean behavior relative to the corruption.

3.2 Circuit Discovery with Fixed Corruptions

Standard circuit discovery fixes the corruption to $\boldsymbol{X}_{\text{ref}}$ and optimizes only over circuits:

$$c^\star_{K,\text{fix}} = \arg\max_{c \in \mathcal{C}_K} \mathcal{J}_{\text{fix}}(c) \triangleq \mathbb{E}\left[\tau(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}; c)\right]. \tag{2}$$

However, this objective evaluates circuits only under a single corruption. Since recovery is defined relative to the corruption, the discovered circuit is inherently *corruption-dependent*. A poorly chosen corruption can obscure the contribution of an otherwise relevant circuit, suggesting that corruption choice should be treated as part of the discovery problem rather than a fixed input.

3.3 CORRUPTION-AWARE CIRCUIT DISCOVERY.

The fixed-corruption objective in Eqn. (2) evaluates circuits relative to the benchmark reference corruption. We further make this dependence explicit by allowing the corruption to vary within a reference-anchored admissible family, rather than over arbitrary counterfactuals. The discovery problem then becomes a joint optimization over the corruption used to define the behavioral contrast and the circuit recovered under that contrast.

For each clean–reference pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, we use the benchmark reference corruption as an anchor. A task-specific anchoring rule Γ specifies which part of the reference corruption may be edited and which replacements preserve the task structure. Formally, let $(\kappa, \mathcal{R}) \triangleq \Gamma(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, where κ is the editable task-relevant coordinate in $\boldsymbol{x}_{\text{ref}}$, and $\mathcal{R}$ is the set of semantically admissible replacements for that coordinate. For $\boldsymbol{x}_{\text{ref}} = (r_1, \ldots, r_L)$ and any replacement $r \in \mathcal{R}$, define the single-coordinate edit $\boldsymbol{\gamma}(\boldsymbol{x}_{\text{ref}}, \kappa, r) \triangleq (r_1, \ldots, r_{\kappa-1}, r, r_{\kappa+1}, \ldots, r_L)$. The anchoring rule is determined by the benchmark structure rather than learned, so the search space remains tied to the original reference corruption. Among these semantically admissible edits, we retain those that induce a sufficiently strong behavioral contrast. Specifically, for each clean–reference pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, we define the reference-anchored admissible corruption family as

$$\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \triangleq \{\boldsymbol{\gamma}(\boldsymbol{x}_{\text{ref}}, \kappa, r) : r \in \mathcal{R},\ \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{\gamma}(\boldsymbol{x}_{\text{ref}}, \kappa, r)) \geq \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})\}\,. \quad (3)$$

Thus, $\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ contains task-preserving corruptions that remain anchored to the benchmark reference and are at least as behaviorally disruptive as the original reference corruption. Thus, by Assumption 3.1, any $\boldsymbol{x} \in \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ has a non-trivial clean–corruption gap, and the corresponding normalized recovery in Eqn. (1) is well defined. A corruption selector maps each clean–reference pair to one admissible corruption. We define the selector class as

$$\Pi_{\mathcal{F}} \triangleq \{\pi : \pi(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \in \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \quad P\text{-a.s.}\}\,. \quad (4)$$

For a realized pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, the value $\pi(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ is the corruption used to evaluate a candidate circuit on that example.

Given the admissible family in Eqn. (3), circuit discovery is no longer only the problem of choosing a sparse circuit under a fixed benchmark corruption. Instead, we jointly choose a sparse circuit and an admissible corruption selector. At the population level, the corruption-aware discovery objective is

$$(c_K^\star, \pi^\star) = \arg \max_{c \in \mathcal{C}_K,\ \pi \in \Pi_{\mathcal{F}}} \mathcal{J}_{\text{cf}}(c, \pi) \triangleq \mathbb{E}_{(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}) \sim P} \left[\tau(\boldsymbol{X}_{\text{clean}}, \pi(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}); c)\right]. \quad (5)$$

The selected corruption for a realized pair is $\boldsymbol{x}_{\text{cor}}^\star \triangleq \pi^\star(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$.

Connection to Fixed-corruption Discovery. The corruption-aware formulation includes fixed-corruption discovery as a special case. If the admissible family collapses to the benchmark reference singleton $\{\boldsymbol{x}_{\text{ref}}\} = \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, then every admissible selector must satisfy $\boldsymbol{x}_{\text{ref}} = \pi(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ almost surely. In this case, the corruption coordinate becomes fixed, and Eqn. (5) reduces exactly to the fixed-corruption objective in Eqn. (2). Thus, our formulation generalizes the standard fixed-corruption protocol rather than replacing it.

More importantly, allowing the corruption to vary can also improve the best achievable recovery whenever the admissible family contains alternatives to the benchmark reference corruption.

Theorem 3.2 (Recovery improvement over fixed corruption)**.** *Suppose Assumption 3.1 holds and* $\boldsymbol{x}_{\text{ref}} \in \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ *for* P*-almost every* $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$*. Then,*

$$\max_{c \in \mathcal{C}_K,\ \pi \in \Pi_{\mathcal{F}}} \mathcal{J}_{\text{cf}}(c, \pi) \geq \max_{c \in \mathcal{C}_K} \mathcal{J}_{\text{fix}}(c). \quad (6)$$

Remark 3.3. Theorem 3.2 shows that making corruption choice explicit cannot reduce the population-level recovery optimum as long as the benchmark reference corruption is still admissible. The gain is zero when the best selector is the reference selector, and becomes positive only when some admissible corruption exposes a more informative behavioral contrast for recovering a sparse circuit. This motivates the learning problem studied in next section: how to efficiently search over admissible corruptions while simultaneously recovering a sparse circuit.

4 LEARNING CIRCUITS WITH LEARNABLE CORRUPTIONS

Eqn. (5) defines a joint optimization over a sparse circuit and an example-wise corruption selector. Direct optimization is difficult because both coordinates are discrete and coupled: the selected corruption depends on the current circuit, while the recovered circuit depends on the selected corruptions. To address these two challenges, we introduce **S**tructure **E**dit and **E**dge **D**iscovery (SEED), a tractable alternating approximation that parameterizes both coordinates and updates them in turn. We first describe the differentiable circuit and stochastic corruption parameterizations, and then present the two alternating updates. Pseudocode is provided in Algorithm 1 in Appendix E.2.

4.1 PARAMETERIZING CIRCUITS AND CORRUPTIONS

Parameterization of the circuit. Direct optimization over $\mathcal{C}_K$ is combinatorial, and the corruption update requires a differentiable signal that reflects the current circuit state. We therefore parameterize candidate edges by shared logits $\boldsymbol{\omega} \in \mathbb{R}^{|E|}$ and decode them into a hard budget-K circuit $c_K(\boldsymbol{\omega}) \triangleq \{e \in E : e \text{ is selected by } \mathrm{TopKMask}_K(\boldsymbol{\omega})\}$. The forward pass uses this exact K-edge circuit, while gradients are passed through a soft relaxation of the same edge scores. This hard-forward, soft-backward parameterization keeps the learned circuit as a single hard budget-K circuit shared across all examples, while still providing a differentiable signal for optimization.

This soft relaxation further induces a differentiable patched score $\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega})$, formally defined in Appendix E.1. Accordingly, we define the soft normalized recovery as

$$\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) \triangleq \frac{\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - m(\boldsymbol{x}_{\mathrm{cor}})}{\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})}. \tag{7}$$

This quantity is the differentiable counterpart of Eqn. (1) and is used by Structure Edit step to score candidate corruptions under the current circuit state.

Stochastic corruption selection. In Eqn. (5), the corruption coordinate is a selector π that maps each clean–reference pair to one admissible corruption. Directly optimizing this selector requires discrete, example-wise decisions whose usefulness depends on the current circuit. To make this coordinate trainable, we replace π during training with a circuit-conditioned stochastic selector. For current circuit parameters $\boldsymbol{\omega}$, the stochastic selector is a conditional distribution $q_{\boldsymbol{\omega}}(\cdot \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$ over corruptions for each clean–reference pair.

We define the stochastic selector class as

$$\mathcal{Q}_{\mathcal{F}} \triangleq \{q : \mathrm{supp}(q(\cdot \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})) \subseteq \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \quad P\text{-a.s.}\}. \tag{8}$$

Thus, for almost every clean–reference pair, the stochastic selector assigns probability only to admissible corruptions in the corresponding admissible family.

4.2 STRUCTURE EDIT

With the circuit parameters $\boldsymbol{\omega}$ fixed, the corruption-side update learns a stochastic selector over reference-anchored candidate corruptions. The goal is not to select corruptions that are merely easy to recover, but to identify corruptions that improve the recovery contrast relative to the benchmark reference corruption. We therefore score each candidate by the reference-anchored soft advantage

$$\widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) \triangleq \widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; \boldsymbol{\omega}). \tag{9}$$

Centering by the benchmark reference converts the corruption score from an absolute recovery measure into a reference-relative improvement signal. Thus, positive values indicate that the candidate corruption provides additional soft recovery evidence beyond the original benchmark corruption under the current circuit state. This soft advantage remains a stable proxy for the corresponding hard-circuit advantage; see Proposition B.1 in Appendix B.

With Eqn. (9) as a stable corruption-side signal, we define the Structure Edit utility of a stochastic selector q as

$$\mathcal{L}_{\mathrm{se}}(\boldsymbol{\omega}; q) \triangleq \mathbb{E}_{\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}}\left[\mathbb{E}_{\boldsymbol{X}_{\mathrm{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}})}\left[\widetilde{A}(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}, \boldsymbol{X}_{\mathrm{cor}}; \boldsymbol{\omega})\right]\right]. \tag{10}$$

A purely advantage-driven Structure Edit step would maximize $\mathcal{L}_{\mathrm{se}}(\boldsymbol{\omega}; q)$ over stochastic selector class $\mathcal{Q}_{\mathcal{F}}$ defined in Eqn. (8). This prior-free update is not sufficient for two reasons. First, the selector should explicitly enforce the admissibility principle in Eqn. (3): the selected corruptions should be at least as behaviorally disruptive as the benchmark reference corruption. Second, because $\mathcal{L}_{\mathrm{se}}$ is linear in q, an unconstrained maximization would collapse onto the highest-advantage candidate and become a cherry-picking step rather than a controlled corruption update.

We therefore regularize the selector toward a fixed semantic prior $p_{\mathrm{MLM}}(\cdot \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, defined over the reference-anchored admissible family and instantiated by a frozen Masked Language Model (MLM) in our implementation (Shin et al., 2020). This prior represents contextually plausible corruptions before observing the current circuit state, while the soft advantage provides circuit-specific evidence for reweighting candidates. The final Structure Edit update is the constrained reweighting problem

$$\max_{q} \quad \mathcal{L}_{\mathrm{se}}(\boldsymbol{\omega}; q) \tag{11}$$

$$\text{s.t.} \quad \mathbb{E}_{\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}} \left[\mathbb{E}_{\boldsymbol{X}_{\mathrm{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}})} \left[\Delta(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{cor}}) \right] - \Delta(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}) \right] \geq 0, \tag{12}$$

$$\mathbb{E}_{\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}} \left[\mathrm{KL}(q(\cdot \mid \boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}) \,\|\, p_{\mathrm{MLM}}(\cdot \mid \boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}})) \right] \leq \epsilon_{\mathrm{KL}}. \tag{13}$$

The constrained objective can be interpreted as a prior-constrained reweighting of candidate corruptions. The soft advantage provides circuit-specific evidence, while the gap-floor and KL constraints control admissibility and deviation from the frozen MLM prior. Solving the corresponding Lagrangian yields the following exponential-tilting form.

Theorem 4.1 (Gibbs form of the Structure Edit update)**.** *Fix the circuit parameters $\boldsymbol{\omega}$. Suppose Assumption 3.1 holds. For P-almost every clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, assume the admissible corruption family $\mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$ is finite and $p_{\mathrm{MLM}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) > 0$ for every $\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. Assume further that an optimal solution of Eqn.* (11) *satisfies the KKT conditions with multipliers $\lambda^\star \geq 0$ and $\eta^\star > 0$ associated with the gap-floor constraint in Eqn.* (12) *and the KL trust-region constraint in Eqn.* (13)*, respectively. Then, for P-almost every clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, the optimal stochastic selector $q^\star_{\boldsymbol{\omega}}$ satisfies*

$$q^\star_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \propto p_{\mathrm{MLM}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \cdot \exp\left(\frac{\widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})}{\eta^\star} \right), \tag{14}$$

where $\lambda^\star$ enforces the gap-floor constraint, and $\eta^\star$ controls the strength of the KL trust region.

Thus, Structure Edit can be viewed as a controlled posterior-style update: the frozen MLM prior supplies the semantic base distribution, while the current circuit reweights candidate corruptions through the soft advantage and the gap-floor term. A large $\eta^\star$ keeps the selector close to the prior, whereas a small $\eta^\star$ allows sharper concentration on high-evidence corruptions.

At iteration t, given the current circuit parameters $\boldsymbol{\omega}_t$, we instantiate the Structure Edit distribution as $q^\star_t(\cdot \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \triangleq q^\star_{\boldsymbol{\omega}_t}(\cdot \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. We then decode $q^\star_t$ into a deterministic selector used by Edge Discovery: for P-almost every clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$,

$$\pi^\star_t : (\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \mapsto \arg \max_{\boldsymbol{x} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})} q^\star_t(\boldsymbol{x} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}). \tag{15}$$

This decoded selector defines the corruptions used in the next circuit-side update.

4.3 Edge Discovery

After Structure Edit selects corruptions through the decoded selector $\pi^\star_t$, the circuit-side update fixes this selector and updates the shared edge logits $\boldsymbol{\omega}$. This step asks a complementary question to Structure Edit: given the current learned corruptions, which budget-K circuit best recovers the clean behavior? Thus, Edge Discovery optimizes the circuit coordinate while keeping the corruption coordinate fixed.

Under Assumption 3.1, the normalized recovery τ lies in $[0, 1]$ for any valid corruption and hard circuit. Edge Discovery therefore uses normalized recovery as the circuit-side learning target under the selected corruptions. We define the population circuit-side objective as

$$\mathcal{L}_{\mathrm{ed}}(\boldsymbol{\omega}; \pi^\star_t) \triangleq \mathbb{E}_{\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}} \left[\Phi_{\mathrm{ed}}(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}, \pi^\star_t(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}); c_K(\boldsymbol{\omega})) \right], \tag{16}$$

where $\Phi_{\mathrm{ed}}(\cdot)$ is a recovery-aligned shaping utility whose optimum corresponds to matching the clean full-model behavior under the selected corruptions. The circuit update is

$$\boldsymbol{\omega}_{t+1}^{\star} = \arg\max_{\boldsymbol{\omega}} \mathcal{L}_{\mathrm{ed}}\left(\boldsymbol{\omega}; \pi_t^{\star}\right). \tag{17}$$

Remark 4.2. The shaping utility $\Phi_{\mathrm{ed}}(\cdot)$ is used only in Edge Discovery, where the shared edge logits $\boldsymbol{\omega}$ are optimized by gradient-based updates. Structure Edit instead uses the unshaped reference-anchored advantage, keeping the corruption update linear in q and yielding the exponential-tilting form in Theorem 4.1.

During optimization, the forward pass uses the decoded hard circuit $c_K(\boldsymbol{\omega})$, while gradients are routed through the soft edge-score relaxation introduced above. Thus, Edge Discovery learns one shared sparse circuit under the corruptions selected by Structure Edit, while preserving the exact hard budget used for evaluation. In the finite-sample implementation, we instantiate Φ_{ed} by the negative patch-to-clean alignment loss, which controls both under-recovery and overshoot; see Appendix E.2.

5 Experiments

We address three questions. **RQ1: Corruption conditioning.** Does the corruption used during discovery affect the recovered circuit? **RQ2: Does learning corruptions help circuit discovery?** Under both denoising and noising objectives, does SEED improve over SEED-Edge, which optimizes the circuit only? **RQ3: Corruption-circuit coupling.** Are the corruptions selected by SEED universally useful training corruptions, or are they coupled to the circuit state that selected them?

5.1 Experimental Setup

Protocol. All experiments use `gpt2-small` as the frozen target model and evaluate hard top-K circuits under matched edge budgets $K \in \{30, 50, 100, 300, 500, 1000, 2000, 10000\}$. We compare SEED with EAP (Nanda, 2023), EAP-IG (Hanna et al., 2024), the learned-mask baseline Edge Pruning (Bhaskar et al., 2024), and SEED-Edge. All methods share tasks, splits, seeds, graph, and budgets. We run SEED and SEED-Edge under both discovery objectives: denoising (D), our primary objective, targets sufficiency and minimizes the patch-to-clean loss in Eqn. (27), and noising (N) targets necessity and minimizes the patch-to-corrupt loss in Eqn. (28) (Appendix E.2). The key control is SEED-Edge, which minimizes the same loss under the benchmark corruption.

Tasks and corruptions. We evaluate on eight tasks: IOI (Wang et al., 2022), Capital-Country, Docstring (Heimersheim & Janiak, 2023), Subject-Verb Agreement (Newman et al., 2021), Greater-Than (Hanna et al., 2023), Negation, Gender Bias (Vig et al., 2020; Chintam et al., 2023), and Hypernymy (Overschelde et al., 2004). For each task, we construct 1,000 clean-reference pairs and use an 8:2 train/test split. During training, SEED selects corruptions from a reference-anchored admissible family $\mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, instantiated with a frozen `roberta-base` masked language model except for Greater-Than, which uses a uniform prior over year tokens; each pool has at most 50 candidates.

Metrics. We report $\mathrm{Faithfulness} = \mathrm{clip}(\tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; c), 0, 1)$ (sufficiency; Eqn. (1)) and $\mathrm{Completeness} = \mathrm{clip}\left(\frac{m(\boldsymbol{x}_{\mathrm{clean}}) - s_N(c)}{m(\boldsymbol{x}_{\mathrm{clean}}) - m(\boldsymbol{x}_{\mathrm{ref}})}, 0, 1\right)$ (necessity), where the noising score $s_N(c)$ runs $\boldsymbol{x}_{\mathrm{clean}}$ with the activations on the edges in c taken from $\boldsymbol{x}_{\mathrm{ref}}$. Both are computed per held-out example under the benchmark reference corruption, which is used for all methods, and then averaged. Because learned corruptions are used only during discovery, neither metric rewards choosing easier evaluation corruptions. The RQ1 and RQ3 diagnostics use held-out normalized recovery, i.e., τ in Eqn. (1) averaged over held-out examples without clipping. Further details are in Appendix F.

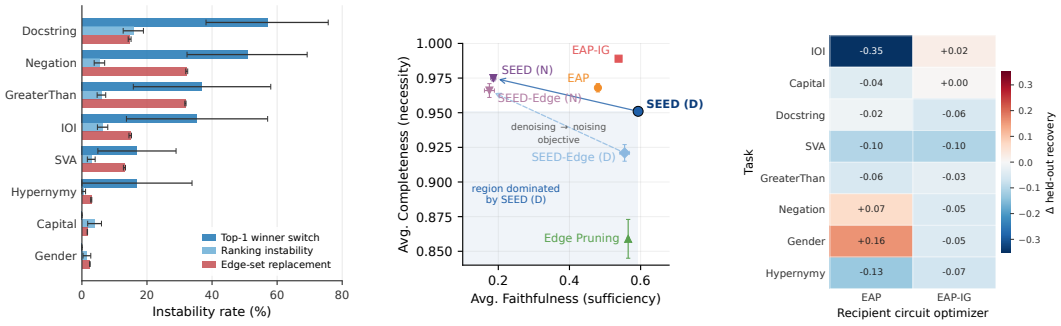
5.2 Main Results

We answer the three questions in turn; complete diagnostics, budget sweeps, and transfer results are in Appendix G.

RQ1: Circuit discovery is strongly conditioned on the corruption used during discovery. Consistent with prior observations that patching results depend on the corruption (Zhang & Nanda, 2023;

Table 1: Held-out Faithfulness per task, with task-averaged Faithfulness (F) and Completeness (C). Each cell averages the eight edge budgets and reports mean ± standard deviation over seeds. (D)/(N): denoising (sufficiency) / noising (necessity) discovery objective; every circuit is evaluated in both directions under the held-out benchmark corruption. Δ rows compare SEED with SEED-EDGE under the same objective. Bold: best per column. Per-task Completeness is in Table 10.

Method	IOI	Capital	Docstring	SVA	Greater	Negation	Gender	Hypernymy	Avg. F	Avg. C
EAP	0.357 ± 0.021	0.600 ± 0.006	0.260 ± 0.009	0.494 ± 0.004	0.784 ± 0.007	0.242 ± 0.018	0.716 ± 0.012	0.392 ± 0.002	0.480 ± 0.006	0.968 ± 0.003
EAP-IG	0.414 ± 0.030	0.667 ± 0.012	0.327 ± 0.014	0.554 ± 0.012	**0.791** ± 0.002	0.337 ± 0.016	0.740 ± 0.012	0.477 ± 0.007	0.538 ± 0.008	**0.989** ± 0.001
Edge Pruning	0.571 ± 0.012	0.542 ± 0.024	**0.468** ± 0.002	**0.613** ± 0.003	0.707 ± 0.015	**0.524** ± 0.007	0.609 ± 0.006	0.492 ± 0.022	0.565 ± 0.003	0.859 ± 0.014
SEED-EDGE (D)	0.374 ± 0.089	**0.689** ± 0.015	0.392 ± 0.002	0.471 ± 0.100	0.787 ± 0.018	0.480 ± 0.063	0.664 ± 0.021	**0.586** ± 0.013	0.555 ± 0.013	0.921 ± 0.006
SEED (D)	**0.612** ± 0.053	0.663 ± 0.015	0.379 ± 0.010	0.582 ± 0.018	0.784 ± 0.003	0.444 ± 0.018	**0.745** ± 0.009	0.533 ± 0.027	**0.593** ± 0.007	0.951 ± 0.003
SEED-EDGE (N)	0.059 ± 0.054	0.237 ± 0.076	0.053 ± 0.009	0.134 ± 0.007	0.564 ± 0.075	0.046 ± 0.033	0.119 ± 0.082	0.189 ± 0.034	0.175 ± 0.014	0.966 ± 0.005
SEED (N)	0.057 ± 0.030	0.108 ± 0.058	0.045 ± 0.031	0.119 ± 0.016	0.730 ± 0.032	0.087 ± 0.041	0.152 ± 0.051	0.194 ± 0.032	0.186 ± 0.007	0.975 ± 0.002
Δ (D): SEED − SEED-EDGE	+0.237	-0.026	-0.013	+0.111	-0.003	-0.036	+0.081	-0.054	+0.038	+0.030
Δ (N): SEED − SEED-EDGE	-0.003	-0.129	-0.008	-0.015	+0.166	+0.040	+0.033	+0.005	+0.011	+0.009



(a) Corruption conditioning (b) Faithfulness vs. Completeness (c) Corruption transfer

Figure 2: Main results. (a) Corruption conditioning of EAP circuits (RQ1): top-1 winner switch, ranking instability, and top-300 edge-set replacement across admissible discovery corruptions. (b) Task-averaged held-out Faithfulness and Completeness for all methods in Table 1 (error bars: ± s.d. over seeds); the shaded region is dominated by SEED (D) in both metrics, and arrows connect the denoising (D) and noising (N) objectives of SEED and SEED-EDGE. (c) Corruption transfer to EAP and EAP-IG, measured as the change in held-out normalized recovery (RQ3).

Heimersheim & Nanda, 2024), changing only the admissible discovery corruption replaces on average 14.2% of the selected top-300 EAP edges. Moreover, when EAP circuits discovered under different admissible corruptions are compared under two admissible evaluation corruptions, the two evaluations disagree on which discovery corruption gives the best-recovering circuit in 26.7% of cases (top-1 winner switch; Figure 2a). The effect is largest on Greater-Than and Negation, whose admissible families offer more diverse contrasts, and smallest on Capital-Country and Hypernymy, which have few valid alternatives. A fixed reference corruption is therefore not a neutral implementation choice; full diagnostics are in Appendix G.1.

RQ2: Learning corruptions improves circuit discovery under both objectives, with task-dependent gains. Table 1 and Figure 2b compare all methods on held-out data under both metrics. Under the denoising objective, which targets faithful circuits, SEED (D) improves over SEED-EDGE (D) in both Faithfulness ($0.555 \rightarrow 0.593$) and Completeness ($0.921 \rightarrow 0.951$). Under the noising objective, which targets necessary circuits, SEED (N) also improves over SEED-EDGE (N) on average in both Completeness ($0.966 \rightarrow 0.975$) and Faithfulness ($0.175 \rightarrow 0.186$). The gains are task-dependent. Under the denoising objective they are largest on IOI ($+0.237$ Faithfulness), whose admissible family contains several semantically plausible corruptions, and positive on SVA and Gender Bias, whereas on more constrained families such as Capital-Country, Negation, and Hypernymy SEED-EDGE is on par or slightly better (Figure 3); Completeness improves on IOI and SVA and is essentially unchanged on five tasks, decreasing only on Docstring. Across methods, SEED (D) attains the highest task-averaged Faithfulness and exceeds the learned-mask baseline Edge Pruning in both budget-averaged metrics ($0.593/0.951$ vs. $0.565/0.859$), and it is the most faithful method at every budget $K \leq 100$ (Appendix G.2), while EAP-IG attains the highest Completeness (0.989). The discovery objective determines which property is emphasized: switching from denoising to noising lowers Faithfulness and raises Completeness for both SEED and SEED-EDGE (Figure 2b), so this

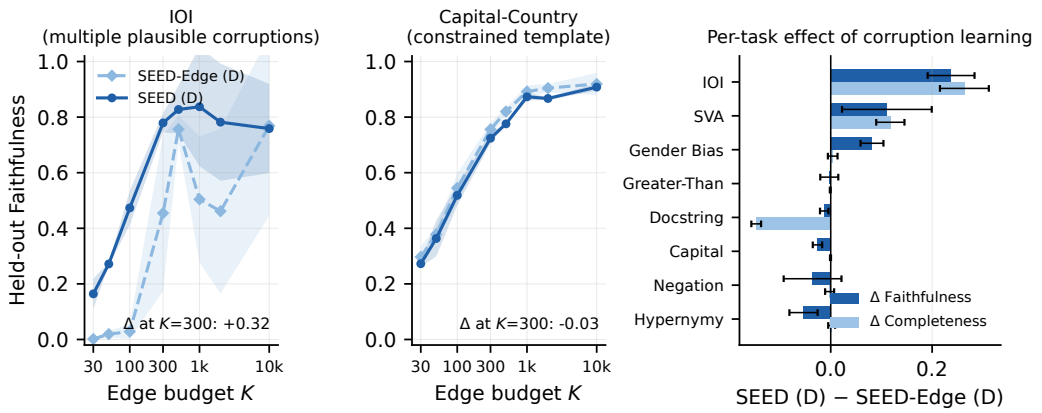


Figure 3: Effect of corruption learning under the optimizer-matched control. Left: held-out Faithfulness versus edge budget for SEED (D) and SEED-EDGE (D) on IOI, whose admissible family contains several plausible corruptions ($+0.32$ at $K = 300$), and on Capital-Country, whose constrained template leaves little room for corruption learning. Right: per-task difference SEED (D) − SEED-EDGE (D) in Faithfulness and Completeness, averaged over the eight budgets ($\pm$ s.d. over seeds).

trade-off stems from the objective rather than from corruption learning, and no method dominates both metrics. Per-task Completeness, budget curves, and normalized-recovery results are in Appendix G.2.

RQ3: Corruption selection must be coupled with circuit optimization. Figure 2c asks whether the corruptions found by SEED can be reused as a universally better training set for other discovery methods. They cannot: at $K = 300$, transferring SEED-optimized corruptions to EAP improves Negation and Gender Bias but hurts the other six tasks, and EAP-IG improves slightly on IOI and Capital-Country but drops on the remaining six. Corruption quality is therefore not absolute but depends on the circuit state and optimizer that use it, which is consistent with our view of circuit discovery as **a coupled problem over circuits and corruptions** and with optimizing the two jointly rather than selecting corruptions in a preprocessing stage. Full results are in Appendix G.3.

5.3 Ablations and Additional Analysis

Ablations. Appendix H ablates the soft-patching surrogate, Edge Discovery loss, Structure Edit, candidate family, warmup, budget regularization, and gate granularity: all main components are needed, and SEED is robust to temperature and optimization choices. Completeness is stable, Faithfulness varies moderately with the number of candidates, and a disruptiveness floor is needed (Appendix H.4).

Additional analysis. Appendix I shows that SEED again improves Faithfulness over SEED-EDGE on four tasks with Qwen3-0.6B (0.822 vs. 0.769); that the two methods share about half of their edges; that Structure Edit adds $1.46\times$ wall time and remains cheaper than Edge Pruning; and that it reorders candidates according to the current circuit.

6 Conclusion

We formulated patching-based circuit discovery as joint optimization over a sparse circuit and a behavioral contrast, with fixed-corruption discovery as a boundary case, and instantiated it with SEED. On eight tasks, learning corruptions improves over optimizing the circuit alone under both objectives and both held-out metrics, with task-dependent gains, and learned corruptions are circuit-coupled rather than universally transferable.

AI USE STATEMENT

We used generative AI tools (large language models) only to polish the manuscript, i.e., to improve grammar, wording, and readability and to assist with LaTeX formatting. We did not use generative AI tools for any of the tasks that require disclosure under the ICLR 2027 AI Policy for Authors, including developing research ideas, hypotheses, or theoretical results, writing proofs, designing experiments, implementing methods, generating or processing data, and analyzing or interpreting results. All AI-assisted edits were reviewed by the authors, and we take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, crowdsourcing, or personal data. All experiments use publicly available pretrained models (`gpt2-small` and `roberta-base`) and template-based prompts for established circuit-discovery tasks from prior work; the Gender Bias task is used only as an existing interpretability benchmark. We are not aware of any specific ethical concerns raised by this work.

REPRODUCIBILITY STATEMENT

All details needed to reproduce our results are provided in the appendix. Appendix A summarizes the notation, and Appendix B gives complete proofs of the theoretical results together with their assumptions. Appendix E describes the soft-patching surrogate and the full SEED training procedure (Algorithm 1). Appendix F specifies the task definitions and dataset construction, task scores, candidate corruption pools, admissibility filtering, model, graph, and edge budgets, baseline implementations, and all hyperparameters and compute (Table 5). Appendices G and H report the full per-task results and ablations.

REFERENCES

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/methods.html.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor V. Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *ArXiv*, abs/2303.08112, 2023. URL https://api.semanticscholar.org/CorpusID:257504984.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. *ArXiv*, abs/2406.16778, 2024. URL https://api.semanticscholar.org/CorpusID:270702321.

Hang Chen, Jiaying Zhu, Xinyu Yang, and Wenya Wang. Rethinking circuit completeness in language models: And, or, and adder gates, 2025. URL https://arxiv.org/abs/2505.10039.

Abhijith Chintam, Rahel Beloch, Willem H. Zuidema, Michael Hanna, and Oskar van der Wal. Identifying and adapting transformer-components responsible for gender bias in an english language model. In *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2023. URL https://api.semanticscholar.org/CorpusID:264306211.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *ArXiv*, abs/2304.14997, 2023. URL https://api.semanticscholar.org/CorpusID:258418244.

Hoagy Cunningham, Aidan Ewart, Logan Riggs Smith, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *ArXiv*, abs/2309.08600, 2023. URL https://api.semanticscholar.org/CorpusID:261934663.

J. Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2017. URL https://api.semanticscholar.org/CorpusID:21698802.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Leo Gao, Tom Dupr'e la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *ArXiv*, abs/2406.04093, 2024. URL https://api.semanticscholar.org/CorpusID:270286001.

Matt Gardner, Yoav Artzi, Jonathan Berant, Ben Bogin, Sihao Chen, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings*, 2020. URL https://api.semanticscholar.org/CorpusID:222124366.

Atticus Geiger, Hanson Lu, Thomas F. Icard, and Christopher Potts. Causal abstractions of neural networks. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235358214.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah D. Goodman, Christopher Potts, and Thomas F. Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *J. Mach. Learn. Res.*, 26:83:1–83:64, 2023. URL https://api.semanticscholar.org/CorpusID:255749463.

Nicholas Goldowsky-Dill, Christopher Macleod, Lucas Jun Koba Sato, and Aryaman Arora. Localizing model behavior with path patching. *ArXiv*, abs/2304.05969, 2023. URL https://api.semanticscholar.org/CorpusID:258079237.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *ArXiv*, abs/2305.00586, 2023. URL https://api.semanticscholar.org/CorpusID:258426987.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *ArXiv*, abs/2403.17806, 2024. URL https://api.semanticscholar.org/CorpusID:268691935.

Stefan Heimersheim and Jett Janiak. A circuit for python docstrings in a 4-layer attention-only transformer. LessWrong, 2023. URL https://www.lesswrong.com/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching. *ArXiv*, abs/2404.15255, 2024. URL https://api.semanticscholar.org/CorpusID:269302704.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *ArXiv*, abs/1909.12434, 2019. URL https://api.semanticscholar.org/CorpusID:203591519.

J'anos Kram'ar, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components. *ArXiv*, abs/2403.00745, 2024. URL https://api.semanticscholar.org/CorpusID:268230363.

Maximilian Li and Lucas Janson. Optimal ablation for interpretability, 2024. URL https://arxiv.org/abs/2409.09951.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, J'anos Kram'ar, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *ArXiv*, abs/2408.05147, 2024. URL https://api.semanticscholar.org/CorpusID:271843380.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://api.semanticscholar.org/CorpusID:198953378.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *ArXiv*, abs/2403.19647, 2024. URL https://api.semanticscholar.org/CorpusID:268732732.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems 35*, 2022. URL https://api.semanticscholar.org/CorpusID:255825985.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit faithfulness metrics are not robust, 2024. URL https://arxiv.org/abs/2407.08734.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. Everything, everywhere, all at once: Is mechanistic interpretability identifiable?, 2025. URL https://arxiv.org/abs/2502.20914.

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

Neel Nanda, Lawrence Chan, Tom Lieberum, Jessie J. Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *ArXiv*, abs/2301.05217, 2023. URL https://api.semanticscholar.org/CorpusID:255749430.

Benjamin Newman, Kai-Siang Ang, Julia Gong, and John Hewitt. Refining targeted syntactic evaluation of language models. *ArXiv*, abs/2104.09635, 2021. URL https://api.semanticscholar.org/CorpusID:233307036.

Christopher Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. 2020. URL https://api.semanticscholar.org/CorpusID:215930358.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova Dassarma, Thomas Henighan, Benjamin Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, John Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom B. Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *ArXiv*, abs/2209.11895, 2022. URL https://api.semanticscholar.org/CorpusID:252532078.

James P. Van Overschelde, Katherine A. Rawson, and John Dunlosky. Category norms: An updated and expanded version of the battig and montague (1969) norms. *Journal of Memory and Language*, 50:289–335, 2004. URL https://api.semanticscholar.org/CorpusID:144158836.

Judea Pearl. Direct and indirect effects. *Probabilistic and Causal Inference*, 2001. URL https://api.semanticscholar.org/CorpusID:5947965.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020. URL https://arxiv.org/abs/2010.15980.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *ArXiv*, abs/2310.10348, 2023. URL https://api.semanticscholar.org/CorpusID:264147090.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. Investigating gender bias in language models using causal mediation analysis. In *Neural Information Processing Systems*, 2020. URL https://api.semanticscholar.org/CorpusID:227275068.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://api.semanticscholar.org/CorpusID:201698258.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *ArXiv*, abs/2211.00593, 2022. URL https://api.semanticscholar.org/CorpusID:253244237.

Lei Yu, Jingcheng Niu, Zining Zhu, Xi Chen, and Gerald Penn. Sheaf discovery with joint computation graph pruning and flexible granularity, 2025. URL https://arxiv.org/abs/2407.03779.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *ArXiv*, abs/2309.16042, 2023. URL https://api.semanticscholar.org/CorpusID:263131114.

Lin Zhang, Wenshuo Dong, Zhuoran Zhang, Shu Yang, Lijie Hu, Ninghao Liu, Pan Zhou, and Di Wang. Eap-gp: Mitigating saturation effect in gradient-based automated circuit identification. *ArXiv*, abs/2502.06852, 2025. URL https://api.semanticscholar.org/CorpusID:276258614.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043, 2023. URL https://api.semanticscholar.org/CorpusID:260202961.

Appendix Contents

A Notation Table **14**

B Proof **14**

C Additional Discussion of Related Work **18**

D Denoising Activation Patching and Patched Score **20**

E Additional Methodological Details **20**

E.1 Soft-Patching Surrogate 21

E.2 SEED Training Algorithm 22

F Experimental Details **25**
F.1 Task Definitions and Dataset Construction 25
F.2 Task Scores 25
F.3 Construction of Reference-Anchored Candidate Pools 26
F.4 Admissibility Filtering 26
F.5 Model, Graph, and Edge Budgets 27
F.6 Baseline Implementations 27
F.7 SEED Experimental Instantiation 27
F.8 Hyperparameters and Compute 27

G Full Results for Main Experiments **28**
G.1 Full Results for RQ1: Corruption Conditioning 28
G.2 Full Results for RQ2: Main Held-Out Performance 31
G.3 Full Results for RQ3: Cross-Evaluation of SEED-Optimized Corruptions 32

H Ablations **34**
H.1 Surrogate Fidelity: Matching Training to Discrete Evaluation 34
H.2 Mechanism Ablations: When Does Structure Edit Help? 37
H.3 Optimization Robustness and Parameterization Choices 38
H.4 Sensitivity to the Admissible Candidate Pool 40

I Additional Analyses **40**
I.1 Second Target Model: Qwen3-0.6B 41
I.2 Computational Cost 41
I.3 Structural Overlap between SEED and SEED-EDGE 41
I.4 Example of Structure Edit Selection 41

J Qualitative Circuit Visualizations **42**

A NOTATION TABLE

B PROOF

Theorem 3.2 (Recovery improvement over fixed corruption)**.** *Suppose Assumption 3.1 holds and* $\boldsymbol{x}_{\mathrm{ref}} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$ *for* P*-almost every* $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$*. Then,*

$$\max_{c \in \mathcal{C}_K,\ \pi \in \Pi_{\mathcal{F}}} \mathcal{J}_{\mathrm{cf}}(c, \pi) \geq \max_{c \in \mathcal{C}_K} \mathcal{J}_{\mathrm{fix}}(c). \tag{6}$$

Proof. For each clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, define the reference selector $\pi_{\mathrm{ref}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, where $\boldsymbol{x}_{\mathrm{ref}} = \pi_{\mathrm{ref}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. By assumption, $\boldsymbol{x}_{\mathrm{ref}} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$ for P-almost every $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. Hence, π_{ref} is feasible, i.e., $\pi_{\mathrm{ref}} \in \Pi_{\mathcal{F}}$.

For any fixed circuit $c \in \mathcal{C}_K$, evaluating the corruption-aware objective at this feasible selector gives

$$\mathcal{J}_{\mathrm{cf}}(c, \pi_{\mathrm{ref}}) = \mathbb{E}\left[\tau\left(\boldsymbol{X}_{\mathrm{clean}}, \pi_{\mathrm{ref}}(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}); c\right)\right].$$

Table 2: Main notations used in the section3 and section 4. Click here to return to the main paper.

Notation	Meaning
$\mathcal{M}$	Frozen language model
$G = (V, E)$	Directed graph representing the forward computation of $\mathcal{M}$, with node set V and edge set E
$c \subseteq E$	Circuit represented as a subset of graph edges
$\mathcal{C}_K$	Budget-K circuit class, $\{c \subseteq E : \lvert c\rvert \leq K\}$
P	Distribution over clean–reference pairs
$\boldsymbol{X}_{\text{clean}}$ / $\boldsymbol{x}_{\text{clean}}$	Clean input and its realization
$\boldsymbol{X}_{\text{ref}}$ / $\boldsymbol{x}_{\text{ref}}$	Benchmark-provided reference corruption and its realization
$\boldsymbol{X}_{\text{cor}}$ / $\boldsymbol{x}_{\text{cor}}$	General counterfactual corruption and its realization
$m(\cdot)$	Task score of the frozen model
$\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})$	Clean–corruption task-score gap
$s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c)$	Score after patching clean activations on circuit c
$\tau(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c)$	Normalized recovery induced by circuit c
Γ	Task-specific anchoring rule for constructing admissible corruptions
κ	Editable task-relevant coordinate in the reference corruption
$\mathcal{R}$	Set of semantically admissible replacements for coordinate κ
$\boldsymbol{\gamma}(\boldsymbol{x}_{\text{ref}}, \kappa, r)$	Single-coordinate edit of $\boldsymbol{x}_{\text{ref}}$ replacing coordinate κ by r
$\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$	Reference-anchored admissible corruption family
π	Deterministic corruption selector
$\Pi_{\mathcal{F}}$	Class of selectors supported on the admissible corruption family
$\mathcal{J}_{\text{fix}}(c)$	Population fixed-corruption recovery objective
$\mathcal{J}_{\text{cf}}(c, \pi)$	Population corruption-aware circuit discovery objective
$\boldsymbol{\omega}$	Shared edge logits used to parameterize the circuit
$c_K(\boldsymbol{\omega})$	Hard budget-K circuit decoded from edge logits $\boldsymbol{\omega}$
$\widetilde{s}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega})$	Differentiable soft-patching score induced by edge logits
$\widetilde{\tau}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega})$	Soft normalized recovery used during Structure Edit
$q_{\boldsymbol{\omega}}(\cdot \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$	Circuit-conditioned stochastic corruption selector
$\mathcal{Q}_{\mathcal{F}}$	Class of stochastic selectors supported on admissible corruptions
$\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega})$	Reference-anchored soft advantage of a candidate corruption
$p_{\text{MLM}}(\cdot \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$	Frozen MLM prior over reference-anchored candidate corruptions
$\lambda^\star$	KKT multiplier for the gap-floor constraint
$\eta^\star$	KKT multiplier controlling the KL trust region
$\Phi_{\text{ed}}(\cdot)$	Recovery-aligned shaping utility used in Edge Discovery
$\mathcal{L}_{\text{se}}(\boldsymbol{\omega}; q)$	Structure Edit utility for stochastic selector q
$\mathcal{L}_{\text{ed}}(\boldsymbol{\omega}; \pi)$	Edge Discovery objective under decoded selector π

By the definition of π_{ref}, this becomes

$$\mathcal{J}_{\mathrm{cf}}(c, \pi_{\mathrm{ref}}) = \mathbb{E}_P \left[\tau \left(\boldsymbol{X}_{\mathrm{clean}}, \boldsymbol{X}_{\mathrm{ref}}; c\right)\right] = \mathcal{J}_{\mathrm{fix}}(c).$$

Therefore, for every $c \in \mathcal{C}_K$,

$$\max_{\pi \in \Pi_{\mathcal{F}}} \mathcal{J}_{\mathrm{cf}}(c, \pi) \geq \mathcal{J}_{\mathrm{cf}}(c, \pi_{\mathrm{ref}}) = \mathcal{J}_{\mathrm{fix}}(c).$$

Taking the maximum over $c \in \mathcal{C}_K$ on both sides yields

$$\max_{c \in \mathcal{C}_K,\, \pi \in \Pi_{\mathcal{F}}} \mathcal{J}_{\mathrm{cf}}(c, \pi) \geq \max_{c \in \mathcal{C}_K} \mathcal{J}_{\mathrm{fix}}(c),$$

completing the proof. □

Structure Edit scores candidate corruptions using the soft-patching surrogate, whereas the final circuit is decoded as a hard budget-K circuit. The next proposition shows that this surrogate does not introduce an uncontrolled corruption-selection signal: if the soft patched score uniformly approximates the hard patched score, then the induced soft normalized recovery and reference-anchored advantage remain close to their hard-circuit counterparts.

Proposition B.1 (Stability of the soft advantage)**.** *Suppose Assumption 3.1 holds. Fix an edge-logit vector $\boldsymbol{\omega}$ and assume that the soft-patching approximation error satisfies*

$$|\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}; \boldsymbol{\omega}) - s(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}; c_K(\boldsymbol{\omega}))| \leq \epsilon_{\mathrm{soft}}$$

for every $\boldsymbol{x} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. Then, for every $\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$,

$$|\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))| \leq \frac{\epsilon_{\mathrm{soft}}}{\delta_\Delta},$$

and

$$\left|\widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - A(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))\right| \leq \frac{2\epsilon_{\mathrm{soft}}}{\delta_\Delta}.$$

Proof. Fix a clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$ and a candidate corruption $\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{F}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$. By the definition of normalized recovery,

$$\tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega})) = \frac{s(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega})) - m(\boldsymbol{x}_{\mathrm{cor}})}{\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})},$$

and by the definition of soft normalized recovery,

$$\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) = \frac{\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - m(\boldsymbol{x}_{\mathrm{cor}})}{\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})}.$$

Therefore,

$$|\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))| = \frac{|\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - s(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))|}{\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})}.$$

By Assumption 3.1, $\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}) \geq \delta_\Delta > 0$. Together with the assumed soft-patching approximation bound, this gives

$$|\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))| \leq \frac{\epsilon_{\mathrm{soft}}}{\delta_\Delta}. \tag{18}$$

For the advantage bound, recall that

$$\widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) = \widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; \boldsymbol{\omega}),$$

and

$$A(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega})) = \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega})) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; c_K(\boldsymbol{\omega})).$$

Hence,

$$\begin{aligned}
&\left|\widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - A(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))\right| \\
&\quad \leq |\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; c_K(\boldsymbol{\omega}))| + |\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; \boldsymbol{\omega}) - \tau(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}; c_K(\boldsymbol{\omega}))|.
\end{aligned}$$

The first term is bounded by $\epsilon_{\text{soft}}/\delta_\Delta$ by Eqn. (18). The same argument applies to the benchmark reference corruption $\boldsymbol{x}_{\text{ref}}$, since Assumption 3.1 also gives $\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \geq \delta_\Delta$, and the soft-patching approximation bound is assumed to hold for $\boldsymbol{x}_{\text{ref}}$ as well. Therefore,

$$\left|\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) - A(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; c_K(\boldsymbol{\omega}))\right| \leq \frac{2\epsilon_{\text{soft}}}{\delta_\Delta},$$

completing the proof. □

Theorem 4.1 (Gibbs form of the Structure Edit update)**.** *Fix the circuit parameters $\boldsymbol{\omega}$. Suppose Assumption 3.1 holds. For P-almost every clean–reference pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, assume the admissible corruption family $\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ is finite and $p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) > 0$ for every $\boldsymbol{x}_{\text{cor}} \in \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$. Assume further that an optimal solution of Eqn.* (11) *satisfies the KKT conditions with multipliers $\lambda^\star \geq 0$ and $\eta^\star > 0$ associated with the gap-floor constraint in Eqn.* (12) *and the KL trust-region constraint in Eqn.* (13)*, respectively. Then, for P-almost every clean–reference pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, the optimal stochastic selector $q^\star_{\boldsymbol{\omega}}$ satisfies*

$$q^\star_{\boldsymbol{\omega}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \propto p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \cdot \exp\left(\frac{\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})}{\eta^\star}\right), \tag{14}$$

where $\lambda^\star$ enforces the gap-floor constraint, and $\eta^\star$ controls the strength of the KL trust region.

Proof. Fix the circuit parameters $\boldsymbol{\omega}$. Recall that Structure Edit update maximizes

$$\mathcal{L}_{\text{se}}(\boldsymbol{\omega}; q) = \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\mathbb{E}_{\boldsymbol{X}_{\text{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})}\left[\widetilde{A}(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}, \boldsymbol{X}_{\text{cor}}; \boldsymbol{\omega})\right]\right],$$

subject to the gap-floor constraint

$$\mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\mathbb{E}_{\boldsymbol{X}_{\text{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})}\left[\Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{cor}})\right]\right] \geq \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right],$$

and the KL trust-region constraint

$$\mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\text{KL}\left(q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}) \,\|\, p_{\text{MLM}}(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right)\right] \leq \epsilon_{\text{KL}}.$$

Therefore, for multipliers $\lambda \geq 0$ and $\eta \geq 0$, the Lagrangian is

$$\begin{aligned}
\mathfrak{L}(q, \lambda, \eta) =& \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\mathbb{E}_{\boldsymbol{X}_{\text{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})}\left[\widetilde{A}(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}, \boldsymbol{X}_{\text{cor}}; \boldsymbol{\omega})\right]\right] \\
&+ \lambda\left\{\mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\mathbb{E}_{\boldsymbol{X}_{\text{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})}\left[\Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{cor}})\right]\right]\right. \\
&\qquad \left. - \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right]\right\} \\
&+ \eta\left\{\epsilon_{\text{KL}} - \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\text{KL}\left(q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}) \,\|\, p_{\text{MLM}}(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right)\right]\right\}.
\end{aligned}$$

By assumption, the optimal KKT multipliers $\lambda^\star \geq 0$ and $\eta^\star > 0$ exist. For fixed $(\lambda^\star, \eta^\star)$, the terms $-\lambda^\star \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\left[\Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right] + \eta^\star \epsilon_{\text{KL}}$ do not depend on q. Thus, maximizing the Lagrangian $\mathfrak{L}(q, \lambda^\star, \eta^\star)$ over q is equivalent to maximizing

$$\begin{aligned}
\tilde{\mathfrak{L}}(q, \lambda^\star, \eta^\star) = \mathbb{E}_{\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}}\Bigg[&\mathbb{E}_{\boldsymbol{X}_{\text{cor}} \sim q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})}\Big[\widetilde{A}(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}, \boldsymbol{X}_{\text{cor}}; \boldsymbol{\omega}) \\
&+ \lambda^\star \Delta(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{cor}})\Big] \\
&- \eta^\star \text{KL}\left(q(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}}) \,\|\, p_{\text{MLM}}(\cdot \mid \boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})\right)\Bigg].
\end{aligned}$$

Since the expectation over $(\boldsymbol{X}_{\text{clean}}, \boldsymbol{X}_{\text{ref}})$ is linear and the conditional distribution $q(\cdot \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ is normalized separately for each clean–reference pair, the optimization decomposes pointwise

over clean–reference pairs. Therefore, for P-almost every fixed pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, the conditional distribution $q(\cdot \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ solves

$$\begin{aligned}\max_{q(\cdot|\boldsymbol{x}_{\text{clean}},\boldsymbol{x}_{\text{ref}})} \quad & \sum_{\boldsymbol{x}_{\text{cor}}\in\mathcal{F}(\boldsymbol{x}_{\text{clean}},\boldsymbol{x}_{\text{ref}})} q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})\Big[\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})\Big] \\ & - \eta^\star \sum_{\boldsymbol{x}_{\text{cor}}\in\mathcal{F}(\boldsymbol{x}_{\text{clean}},\boldsymbol{x}_{\text{ref}})} q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \log \frac{q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})}{p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})} \\ \text{s.t.} \quad & \sum_{\boldsymbol{x}_{\text{cor}}\in\mathcal{F}(\boldsymbol{x}_{\text{clean}},\boldsymbol{x}_{\text{ref}})} q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) = 1, \\ & q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \geq 0.\end{aligned}$$

Consider a normalization multiplier $\alpha(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ for the equality constraint. Then, the first-order stationarity condition for every corruption with positive probability is

$$\begin{aligned}&\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}) \\ &\quad - \eta^\star \left(\log \frac{q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})}{p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})} + 1\right) + \alpha(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) = 0.\end{aligned}$$

Rearranging gives

$$\begin{aligned}\log \frac{q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})}{p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})} =& \frac{\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})}{\eta^\star} \\ &+ \frac{\alpha(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})}{\eta^\star} - 1.\end{aligned}$$

Exponentiating both sides yields

$$\begin{aligned}q(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) =& p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \\ &\cdot \exp\left(\frac{\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})}{\eta^\star}\right) \\ &\cdot \exp\left(\frac{\alpha(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})}{\eta^\star} - 1\right).\end{aligned}$$

Note that the last exponential factor depends only on $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ and not on $\boldsymbol{x}_{\text{cor}}$. It is therefore absorbed into the normalization constant. Hence, for P-almost every clean–reference pair,

$$\begin{aligned}q^\star_{\boldsymbol{\omega}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \propto& \, p_{\text{MLM}}(\boldsymbol{x}_{\text{cor}} \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \\ &\cdot \exp\left(\frac{\widetilde{A}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}) + \lambda^\star \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}})}{\eta^\star}\right),\end{aligned}$$

completing the proof. □

C Additional Discussion of Related Work

Mechanistic interpretability. Mechanistic interpretability seeks to explain model behavior by identifying internal structures that are not only correlated with outputs, but also causally involved in producing them. A central approach is to describe behavior in terms of circuits: sparse sets of components or paths that implement recognizable computations, such as induction heads, indirect object identification, greater-than reasoning, and docstring completion (Elhage et al., 2021; Olah et al., 2020; Olsson et al., 2022; Wang et al., 2022; Hanna et al., 2023; Heimersheim & Janiak, 2023). This circuit-level view is closely related to causal intervention methods, as a proposed circuit is typically assessed by intervening on activations or paths and measuring whether the intervention restores or disrupts the intended behavior (Goldowsky-Dill et al., 2023; Zhang & Nanda, 2023; Heimersheim & Nanda, 2024). It is also connected to causal mediation analysis, which studies how effects are transmitted through intermediate variables (Pearl, 2001). Related causal extraction

work provides a more formal account of when neural computations can be aligned with higher-level causal variables (Geiger et al., 2021; 2023). Other strands of mechanistic interpretability focus on complementary units of explanation: representation-level studies ask where syntactic, semantic, factual, or social variables are encoded, localized, mediated, decoded, or edited (Belrose et al., 2023; Meng et al., 2022; Newman et al., 2021; Vig et al., 2020; Chintam et al., 2023), while feature-based approaches use dictionary learning, sparse autoencoders, or attribution graphs to decompose dense activations into interpretable features and causal feature graphs (Cunningham et al., 2023; Gao et al., 2024; Lieberum et al., 2024; Marks et al., 2024; Ameisen et al., 2025; Lindsey et al., 2025). Our work builds on the circuit-level and intervention-based branch of mechanistic interpretability, but focuses on a usually fixed part of the intervention protocol: the corrupted counterfactual that defines the behavioral contrast used for patching-based faithfulness.

Circuit discovery. Circuit discovery implements a circuit-level perspective on mechanistic interpretability by identifying sparse subnetworks sufficient for task-relevant behavior. Early and task-specific studies constructed circuits for behaviors such as indirect object identification, docstring completion, greater-than reasoning, and bias-related predictions, showing that complex language-model behaviors can sometimes be decomposed into interacting attention heads, MLPs, features, or paths (Wang et al., 2022; Heimersheim & Janiak, 2023; Hanna et al., 2023; Vig et al., 2020; Chintam et al., 2023). Because manual circuit construction is labor-intensive and difficult to scale, recent work has developed automated discovery methods that search over candidate edges or components. ACDC performs greedy pruning using patching-based tests (Conmy et al., 2023), while EAP and EAP-IG estimate edge importance with attribution-based approximations to intervention scores (Syed et al., 2023; Hanna et al., 2024). Other approaches improve scalability or faithfulness through learned edge masks, edge pruning, component localization, or feature-level causal graphs (Bhaskar et al., 2024; Kram'ar et al., 2024; Marks et al., 2024; Ameisen et al., 2025). Despite these differences, existing methods share a common structure: they optimize the sparse circuit under a clean-corrupted contrast specified before discovery. Thus, prior work mainly asks which circuit best explains a fixed behavioral contrast; our work asks how circuit discovery changes when the corruption defining that contrast is also treated as a discovery coordinate.

Corruptions, counterfactuals, and benchmark protocol choices. Patching-based circuit discovery depends on a clean input and a corrupted counterpart. The corruption is not merely a nuisance perturbation: it defines the behavioral contrast that a candidate circuit is asked to repair, and therefore determines the meaning of the resulting recovery or faithfulness score. Prior work on activation and path patching has emphasized that intervention sites, metrics, and patching protocols affect how causal effects should be interpreted (Goldowsky-Dill et al., 2023; Zhang & Nanda, 2023; Heimersheim & Nanda, 2024). In circuit benchmarks, the corrupted input is typically fixed as part of the task protocol. For example, IOI, Greater-Than, syntactic agreement, and bias-related evaluations use task-specific counterfactual constructions to elicit a behavioral difference before circuit search begins (Wang et al., 2022; Hanna et al., 2023; Newman et al., 2021; Vig et al., 2020; Chintam et al., 2023). This standardization is valuable because it enables reproducible and comparable evaluation across discovery methods. However, fixing the corruption also hides a degree of freedom. When multiple semantically admissible corruptions preserve the task structure and induce a nontrivial clean-corruption gap, choosing one of them determines which behavioral distinction the circuit is optimized to recover. This role of admissible corruptions is related in spirit to contrast sets and counterfactually augmented data, where minimally edited examples are used to probe local decision boundaries or expose behavior under controlled semantic changes (Gardner et al., 2020; Kaushik et al., 2019). However, our use of counterfactuals is different: we do not construct them to evaluate input-output robustness, augment training data, or improve generalization. Instead, corruptions are selected inside the patching protocol, where they define the behavioral contrast under which a sparse circuit is discovered. As a result, different admissible corruptions can lead to different recovery profiles, edge rankings, and sparse circuits under the same budget. Our work makes this dependence explicit by treating corruption choice as part of the discovery problem. We keep the benchmark corruption as a reference anchor and retain it for held-out evaluation, but allow discovery to use admissible, gap-constrained counterfactual corruptions derived from that reference.

This differs from adversarial input search, prompt optimization, or universal trigger methods, which seek inputs that change, elicit, or attack model behavior (Shin et al., 2020; Ebrahimi et al., 2017; Wallace et al., 2019; Zou et al., 2023). Our target model is frozen, and the goal is not to find

universally stronger corruptions. Instead, the selected corruptions serve as circuit-coupled contrasts: they are useful because they expose information relevant to the current sparse circuit being optimized, not because they form a generally better corruption dataset.

D Denoising Activation Patching and Patched Score

This section formalizes the hard denoising-patching protocol that defines the patched score s used throughout the paper.

Let the frozen model computation be represented as a directed edge-level graph $G = (V, E)$. Each directed edge $e = (u \to v) \in E$ carries a message from a source node u to a destination node v. For a clean input $\boldsymbol{x}_{\text{clean}}$ and a corrupted input $\boldsymbol{x}_{\text{cor}}$, let $\mu_e^{\text{cor}} = \mu_e(\boldsymbol{x}_{\text{cor}})$ denote the message carried by edge e in the corrupted full-model run. For a hard circuit $c \subseteq E$, the patched computation is defined recursively in causal order. Figure 4 illustrates this procedure: a corrupted forward pass first caches all corrupted edge messages, and the patched computation then allows only edges in c to carry messages generated from the clean-side patched states, while clamping all other edges to their corrupted values. Once the patched source state $h_u^{(c)}$ has been computed, the message sent along edge $e = (u \to v)$ is

$$\mu_e^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right) = \begin{cases} \psi_{u\to v}\left(h_u^{(c)}\right), & e \in c, \\ \mu_e^{\text{cor}}, & e \notin c, \end{cases}$$

where $\psi_{u\to v}$ denotes the local message map from source node u to destination node v. The input arriving at node v is then

$$z_v^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right) = \sum_{e=(u\to v)} \mu_e^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right),$$

followed by the usual frozen node update

$$h_v^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right) = \phi_v\left(z_v^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right)\right).$$

All local maps $\psi_{u\to v}$ and ϕ_v are those of the frozen model $\mathcal{M}$; only the edge messages are intervened on according to the circuit c.

Equivalently, the implementation first runs the corrupted input $\boldsymbol{x}_{\text{cor}}$ to record the corrupted edge messages $\{\mu_e^{\text{cor}}\}_{e\in E}$, and then performs a patched forward computation in which edges outside c are clamped to their corrupted messages, while edges inside c transmit the messages generated by the current patched source states. This causal ordering ensures that downstream source states reflect all upstream patching decisions.

Let $\mathcal{M}^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right)$ denote the model computation produced by this recursive denoising-patching procedure. The hard patched score is defined as the task score of this patched computation:

$$s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c) \triangleq m\Big(\mathcal{M}^{(c)}\left(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}\right)\Big).$$

Under this convention, $c = \emptyset$ gives the corrupted full-model score, since every edge carries its corrupted message:

$$s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; \emptyset) = m(\boldsymbol{x}_{\text{cor}}).$$

Similarly, $c = E$ gives the clean full-model score, since no corrupted messages are injected:

$$s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; E) = m(\boldsymbol{x}_{\text{clean}}).$$

For intermediate circuits $c \subset E$, the patched score measures how much of the clean task behavior is restored when only the messages along the selected edges are allowed to follow the patched clean computation, while all unselected edges remain clamped to their corrupted messages.

E Additional Methodological Details

In this section, we provide the methodological details deferred from Section 4. We first describe the differentiable soft-patching surrogate used to compute the soft patched score $\widetilde{s}$ and the soft normalized recovery $\widetilde{\tau}$. We then give the alternating training procedure that implements Structure Edit and Edge Discovery in SEED.

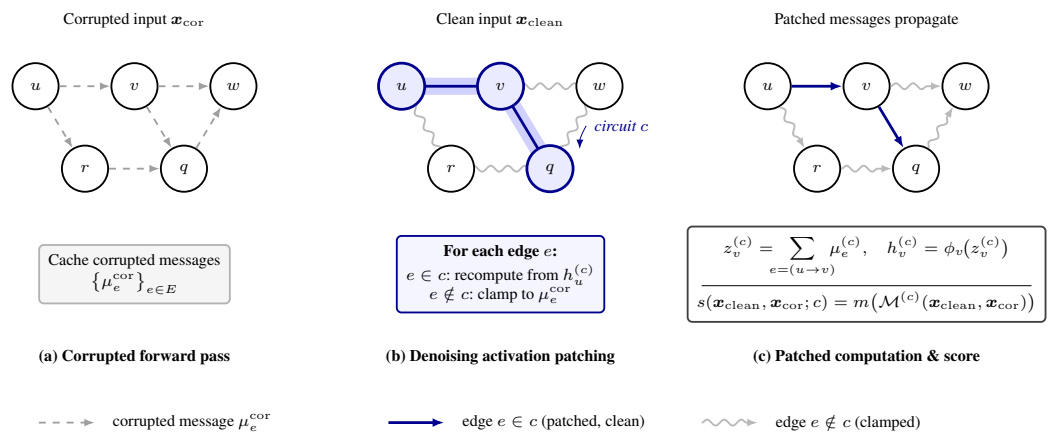


Figure 4: **Denoising activation patching.** (a) The corrupted forward pass on $\boldsymbol{x}_{\text{cor}}$ caches a corrupted message μ_e^{cor} on every edge. (b) On the clean input $\boldsymbol{x}_{\text{clean}}$, edges inside the circuit c are recomputed from upstream patched activations, while edges outside c are clamped to their cached corrupted messages. (c) Patched messages propagate in causal order, yielding patched node states and the circuit score $s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c)$.

E.1 Soft-Patching Surrogate

Structure Edit requires a differentiable circuit-conditioned signal that responds smoothly to changes in the edge logits $\boldsymbol{\omega}$, while preserving the edge-level semantics of denoising patching. The hard patched score $s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; c)$ in Appendix D is defined by a recursive activation patching procedure: selected edges transmit messages generated by the current patched source states, while unselected edges are clamped to their corrupted messages. We introduce the soft-patching surrogate by relaxing this hard edge intervention inside the same recursive patched computation. This yields the soft patched score $\widetilde{s}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega})$.

Corrupted edge messages. Recall that the forward computation of the frozen model is represented as a directed graph $G = (V, E)$. Each directed edge $e = (u \to v) \in E$ carries a message from a source node u to a destination node v. For a corruption $\boldsymbol{x}_{\text{cor}}$, run the frozen model on $\boldsymbol{x}_{\text{cor}}$ and let $\mu_e^{\text{cor}} = \mu_e(\boldsymbol{x}_{\text{cor}})$ denote the corrupted full-model message carried by edge e. These corrupted messages serve as the clamped values for edges that are not patched.

Edge-level relaxation. Given edge logits $\boldsymbol{\omega} \in \mathbb{R}^{|E|}$, the hard budget-$K$ circuit is decoded by selecting the K largest valid edge logits. Let $m_e^{\text{hard}}(\boldsymbol{\omega}) \in \{0, 1\}$ denote this hard top-K mask, so that $c_K(\boldsymbol{\omega}) = \{e \in E : m_e^{\text{hard}}(\boldsymbol{\omega}) = 1\}$. During warmup, we use the sigmoid gate $m_e^{\text{warm}}(\boldsymbol{\omega}) = \sigma(\omega_e/\tau)$, and add a budget penalty encouraging $\sum_{e\in E} m_e^{\text{warm}}(\boldsymbol{\omega})$ to approach K. After warmup, we use a differentiable top-K relaxation. Let $\theta_K(\boldsymbol{\omega})$ denote the K-th largest logit among valid edges. We define

$$\widetilde{m}_e(\boldsymbol{\omega}) = \sigma\left(\frac{\omega_e - \theta_K(\boldsymbol{\omega})}{\tau_c}\right), \tag{19}$$

where $\tau_c > 0$ controls the sharpness of the relaxation. The gate used in the patched computation is the straight-through gate

$$\alpha_e(\boldsymbol{\omega}) = m_e^{\text{hard}}(\boldsymbol{\omega}) + \widetilde{m}_e(\boldsymbol{\omega}) - \text{sg}[\widetilde{m}_e(\boldsymbol{\omega})], \tag{20}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator. Therefore, $\alpha_e(\boldsymbol{\omega})$ has the hard top-K value in the forward pass and the soft top-K gradient in the backward pass.

Soft recursive patched computation. The patched computation is defined recursively in causal order, analogously to the hard denoising-patching procedure in Appendix D. Once the patched source state $\widetilde{h}_u(\boldsymbol{\omega})$ has been computed, the message sent along edge $e = (u \to v)$ is

$$\widetilde{\mu}_e(\boldsymbol{\omega}) = \alpha_e(\boldsymbol{\omega})\psi_{u\to v}\Big(\widetilde{h}_u(\boldsymbol{\omega})\Big) + \big(1 - \alpha_e(\boldsymbol{\omega})\big)\mu_e^{\text{cor}}, \tag{21}$$

where $\psi_{u\to v}$ is the frozen message map from source node u to destination node v. Thus, an edge with gate value 1 transmits the message generated from the patched source state, while an edge with gate value 0 transmits its cached corrupted message.

The input to node v is the sum of incoming patched messages,

$$\widetilde{z}_v(\boldsymbol{\omega}) = \sum_{e=(u\to v)} \widetilde{\mu}_e(\boldsymbol{\omega}),$$

and the node state is updated by the frozen node map

$$\widetilde{h}_v(\boldsymbol{\omega}) = \phi_v(\widetilde{z}_v(\boldsymbol{\omega}))\,.$$

Let $\widetilde{\mathcal{M}}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})$ denote the model computation obtained by applying this recursive patched forward pass. The surrogate patched score is defined by applying the same task-score functional $m(\cdot)$ to the output of this patched computation:

$$\widetilde{s}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}) \triangleq m\Big(\widetilde{\mathcal{M}}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})\Big)\,. \tag{22}$$

The tilde indicates that this score is optimized through the straight-through surrogate. After warmup, the forward pass still follows the decoded hard circuit $c_K(\boldsymbol{\omega})$, while gradients are supplied by the relaxed top-K gate. This score further induces the soft normalized recovery $\widetilde{\tau}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega})$ in Eqn. (7) and the soft advantage $\widetilde{A}$ in Eqn. (9).

Consistency properties. The soft recursive patching procedure in Eqn. (21) satisfies two useful consistency properties.

Reduction to hard denoising patching. When the gates collapse to binary values $\alpha_e(\boldsymbol{\omega}) \in \{0, 1\}$ and encode the mask of some hard circuit $c \subseteq E$, Eqn. (21) becomes

$$\widetilde{\mu}_e(\boldsymbol{\omega}) = \begin{cases} \psi_{u\to v}\Big(\widetilde{h}_u(\boldsymbol{\omega})\Big)\,, & e \in c, \\ \mu_e^{\mathrm{cor}}, & e \notin c. \end{cases}$$

This is exactly the recursive hard denoising-patching computation in Appendix D, with $\widetilde{h}_u(\boldsymbol{\omega}) = h_u^{(c)}$. Hence, the soft-patched score is a continuous relaxation of the hard patched score, not a separate evaluation protocol.

Edge-local causal semantics. Clean-side information can propagate through node v only through incoming edges with nonzero gate. If $\alpha_e(\boldsymbol{\omega}) = 0$, edge e transmits exactly its corrupted full-model message and contributes no patched source-state information downstream. This edge-local property would generally fail under a node-level relaxation: once a clean or patched node state is mixed into a relaxed node representation, it can propagate through all outgoing edges, including edges that are not selected by the circuit. The edge-level relaxation therefore matches the causal semantics of an edge circuit.

Hard-forward, soft-backward training. The straight-through gate in Eqn. (20) ensures that every post-warmup forward pass evaluates exactly the decoded budget-K circuit $c_K(\boldsymbol{\omega})$ under the recursive denoising-patching protocol. Gradients, however, are routed through the differentiable soft top-K gate in Eqn. (19). Thus, the relaxation is used only for optimization, whereas the circuit used for evaluation and reporting is always the hard budget-K circuit.

E.2 SEED Training Algorithm

Algorithm 1 summarizes the finite-sample training procedure for SEED. The algorithm maintains shared edge logits $\boldsymbol{\omega}$, which decode to one budget-K circuit shared across all examples. Each iteration follows the same alternating structure as Section 4: Structure Edit first selects reference-anchored corruptions under the current circuit state, and Edge Discovery then updates the shared edge logits under the selected corruptions.

Structure Edit update. The population Structure Edit update in Theorem 4.1 defines a conditional stochastic selector over admissible corruptions, with KKT multipliers $\lambda^\star$ and $\eta^\star$ for the expected gap-floor constraint and the KL trust-region constraint, respectively. In finite-sample training, we approximate this dual structure at the level of individual pairs: the selector is normalized over the candidate pool $\mathcal{S}_i = \operatorname{supp} p_{\mathrm{MLM}}(\cdot \mid \boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})$, which contains $\mathcal{F}(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})$ and is not pre-filtered by the gap floor; the expected gap floor is enforced for each pair with its own multiplier $\widehat{\lambda}_i(\boldsymbol{\omega})$; and the KL multiplier is implemented as a fixed temperature η. The empirical gap-floor multiplier $\widehat{\lambda}_i(\boldsymbol{\omega})$ is not a separately learned parameter; it is recomputed at every step for the current edge logits.

For the current edge logits $\boldsymbol{\omega}$, and for any clean–reference pair $(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}})$, define

$$\widetilde{A}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}) = \widetilde{A}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}).$$

Given a candidate multiplier $\lambda \geq 0$, we instantiate the finite-sample selector using the Gibbs form from Theorem 4.1:

$$q^{\lambda}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \propto p_{\mathrm{MLM}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}) \cdot \exp\left(\frac{\widetilde{A}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{ref}}, \boldsymbol{x}_{\mathrm{cor}}) + \lambda \Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})}{\eta}\right), \tag{23}$$

where $\boldsymbol{x}_{\mathrm{cor}}$ ranges over the candidate pool of the pair, over which the normalizing constant is taken. The temperature η is the finite-sample counterpart of the KL multiplier $\eta^\star$: smaller values produce sharper posterior-style reweighting of the MLM prior, while larger values keep the selector closer to the prior.

For notational compactness, write $\mathcal{F}_i = \mathcal{F}(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})$ and, for any candidate multiplier $\lambda \geq 0$, $q^{\lambda}_{\mathcal{B},i}(\boldsymbol{x}_{\mathrm{cor}}) \triangleq q^{\lambda}_{\boldsymbol{\omega}}(\boldsymbol{x}_{\mathrm{cor}} \mid \boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})$ for $\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{S}_i$. For each pair $i \in \mathcal{B}$, we choose $\widehat{\lambda}_i(\boldsymbol{\omega})$ as the smallest nonnegative value satisfying the pair-level gap-floor constraint

$$\sum_{\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{S}_i} q^{\widehat{\lambda}_i(\boldsymbol{\omega})}_{\mathcal{B},i}(\boldsymbol{x}_{\mathrm{cor}}) \Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{cor}}) \geq \Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i}). \tag{24}$$

Requiring the expected gap floor for every pair is a stricter, pair-level version of the population constraint in Eqn. (12). Each search over $\widehat{\lambda}_i(\boldsymbol{\omega})$ is one-dimensional. Moreover, the left-hand side of Eqn. (24) is monotone in λ. Indeed, for any $\lambda \geq 0$,

$$\frac{\partial}{\partial \lambda} \sum_{\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{S}_i} q^{\lambda}_{\mathcal{B},i}(\boldsymbol{x}_{\mathrm{cor}}) \Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{cor}}) = \frac{1}{\eta} \mathrm{Var}_{\boldsymbol{X}_{\mathrm{cor}} \sim q^{\lambda}_{\mathcal{B},i}} \left[\Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{X}_{\mathrm{cor}})\right] \geq 0.$$

We therefore solve for each $\widehat{\lambda}_i(\boldsymbol{\omega})$ by bisection, vectorized across the minibatch. If Eqn. (24) already holds at $\lambda = 0$, we set $\widehat{\lambda}_i(\boldsymbol{\omega}) = 0$; otherwise, we bisect on $[0, \lambda_{\max}]$ and use $\lambda_{\max}$ if the constraint cannot be met. Feasibility is ensured by including the benchmark reference corruption in each candidate family and falling back to it when no non-reference candidate is gap-feasible.

Using the multiplier $\widehat{\lambda}_i(\boldsymbol{\omega})$, the Structure Edit distribution instantiated for example i is

$$q^{\star}_{\mathcal{B},i} = q^{\widehat{\lambda}_i(\boldsymbol{\omega})}_{\boldsymbol{\omega}}(\cdot \mid \boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i}). \tag{25}$$

The deterministic corruption used by Edge Discovery is decoded as

$$\widehat{\boldsymbol{x}}_{\mathrm{cor},i} = \arg\max_{\boldsymbol{x}_{\mathrm{cor}} \in \mathcal{F}_i} q^{\star}_{\mathcal{B},i}(\boldsymbol{x}_{\mathrm{cor}}). \tag{26}$$

Thus, within each minibatch update, Structure Edit evaluates a conditional stochastic selector at each clean–reference pair, rather than introducing independent per-example selector parameters. The multiplier $\widehat{\lambda}_i(\boldsymbol{\omega})$ is recomputed from the current circuit at every step and plays the role of the population multiplier in Theorem 4.1.

Edge Discovery update. After Structure Edit decodes the deterministic corruption, Edge Discovery fixes the selected corruptions and updates the shared edge logits $\boldsymbol{\omega}$. This is the finite-sample implementation of the population circuit-side update in Eqn. (17). The population circuit-side

objective is Eqn. (16) in Section 4, with update Eqn. (17). Eqn. (16) uses a recovery-aligned shaping utility Φ_{ed}, whose optimum corresponds to matching the clean full-model behavior under the selected corruptions. In implementation, we realize this optimum condition by minimizing a patch-to-clean alignment loss.

For a minibatch $\mathcal{B}$, we use the following finite-sample Edge Discovery loss:

$$\mathcal{L}_{\mathrm{p2c}}(\boldsymbol{\omega}; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \varphi \left(\frac{s(\boldsymbol{x}_{\mathrm{clean},i}, \widehat{\boldsymbol{x}}_{\mathrm{cor},i}; c_K(\boldsymbol{\omega})) - m(\boldsymbol{x}_{\mathrm{clean},i})}{\Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})} \right), \tag{27}$$

where $\widehat{\boldsymbol{x}}_{\mathrm{cor},i}$ is defined in Eqn. (26) and $\varphi(u) = \sqrt{1+u^2} - 1$. Algorithm 1 then updates $\boldsymbol{\omega}$ by minimizing Eqn. (27). This minimization step implements the same optimum condition as the recovery-aligned objective in Eqn. (16): both favor circuits whose patched computations reproduce the clean full-model behavior under the corruptions fixed by Structure Edit.

Each summand is minimized when the patched score under the selected corruption matches the clean full-model score,

$$s(\boldsymbol{x}_{\mathrm{clean},i}, \widehat{\boldsymbol{x}}_{\mathrm{cor},i}; c_K(\boldsymbol{\omega})) = m(\boldsymbol{x}_{\mathrm{clean},i}).$$

Thus, the implementation enforces the optimum condition described by Φ_{ed}, without requiring a specific closed-form choice of Φ_{ed} in the main text. Unlike directly maximizing the patched score, this patch-to-clean loss anchors the optimum at the clean full-model score and therefore penalizes both under-recovery and overshoot.

The denominator uses the benchmark reference gap $\Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})$, rather than the selected-corruption gap. The selected corruption determines the patched computation in the numerator, whereas the benchmark reference corruption provides a fixed normalization anchor tied to the held-out reference-corruption evaluation protocol. This keeps the loss scale tied to the same reference gap used by the held-out evaluation protocol, while still allowing Structure Edit to provide different training-time contrasts.

Noising objective. The noising variants, SEED (N) and SEED-EDGE (N), target necessity rather than sufficiency and therefore reverse the intervention direction. Instead of restoring clean activations inside the circuit on the corrupted run, noising patching runs the model on the clean input and replaces the activations along the circuit edges with those of the selected corruption, so it tests how much the task behavior degrades when only the circuit is corrupted. With the noising score s_N of Section 3, the minibatch Edge Discovery loss becomes

$$\mathcal{L}_{\mathrm{noise}}(\boldsymbol{\omega}; \mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \varphi \left(\frac{s_N(\boldsymbol{x}_{\mathrm{clean},i}, \widehat{\boldsymbol{x}}_{\mathrm{cor},i}; c_K(\boldsymbol{\omega})) - m(\widehat{\boldsymbol{x}}_{\mathrm{cor},i})}{\Delta(\boldsymbol{x}_{\mathrm{clean},i}, \boldsymbol{x}_{\mathrm{ref},i})} \right), \tag{28}$$

which is minimized when corrupting the circuit alone reproduces the corrupted behavior, i.e., when the circuit is necessary for the clean behavior. Structure Edit uses the same construction with the noising score: the soft recovery in Eqn. (7) is replaced by $(m(\boldsymbol{x}_{\mathrm{clean}}) - \widetilde{s}_N(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}}; \boldsymbol{\omega}))/\Delta(\boldsymbol{x}_{\mathrm{clean}}, \boldsymbol{x}_{\mathrm{cor}})$. All other components, including the gate parameterization, warmup, budget enforcement, and the normalization by the benchmark reference gap, are shared with the denoising objective, and SEED-EDGE (N) minimizes the same loss with the corruption fixed to $\boldsymbol{x}_{\mathrm{ref},i}$. Every circuit, regardless of its discovery objective, is evaluated with both Faithfulness and Completeness on the held-out benchmark corruptions.

During this Edge Discovery step, the forward pass uses the decoded hard circuit $c_K(\boldsymbol{\omega})$, while gradients are routed through the hard-forward, soft-backward relaxation described in Appendix E.1. Therefore, the optimization updates one shared edge-logit vector $\boldsymbol{\omega}$, and the final reported circuit remains the hard budget-K circuit $c_K(\boldsymbol{\omega})$.

Warmup and budget enforcement. For the first T_{warm} steps, SEED uses sigmoid gates without top-K thresholding and adds a budget penalty to encourage the total gate mass to approach K. After warmup, SEED switches to the hard-forward, soft-backward top-K gate, so every forward pass evaluates exactly K active edges while gradients are still routed through the differentiable relaxation. The final reported circuit and all held-out evaluations use the hard circuit $c_K(\boldsymbol{\omega})$, not the soft surrogate.

Table 3: Task examples and editable coordinates. Underlined spans indicate the editable coordinate κ in the benchmark reference corruption.

Task	Clean input $\boldsymbol{x}_{\text{clean}}$	Reference $\boldsymbol{x}_{\text{ref}}$	Example Slot κ
IOI (Wang et al., 2022)	`Then, Mary and John went to the store. John gave a book to`	`Then, Mary and John went to the store. Alice gave a book to`	S2 name token
Capital–Country	`"Tirana, the capital of"`	`"London, the capital of"`	capital token
Docstring (Heimersheim & Janiak, 2023)	`def f(load, size, files): ... :param load: ... :param size: ... :param`	`def f(load, path, files): ... :param load: ... :param path: ... :param`	argument name
SVA (Newman et al., 2021)	`The keys that the chef likes`	`The key that the chef likes`	subject number
Greater–Than (Hanna et al., 2023)	`The war lasted from the year 1742 to 17`	`The war lasted from the year 1701 to 17`	2-digit year suffix
Negation	`The apple is hot, not cold. It is`	`The apple is hot, quite cold. It is`	negation token
Gender Bias (Vig et al., 2020; Chintam et al., 2023)	`The nurse said that`	`The man said that`	profession noun
Hypernymy (Overschelde et al., 2004)	`, diamonds, rubies, emeralds and other`	`, apples, oranges, bananas and other`	instance phrase

F Experimental Details

This appendix specifies the experimental protocol underlying Section 5. The algorithmic details of SEED, including the soft-patching surrogate and the alternating Structure Edit / Edge Discovery updates, are given in Appendix E. Here we focus on the experimental instantiation: task construction, task scores, candidate corruption pools, admissibility filtering, baselines, model configuration, edge budgets, and hyperparameters. Across all methods, the target model, graph construction, edge-budget grid, train/test splits, and held-out evaluation protocol are shared. Methods differ only in how they select the circuit $c \in \mathcal{C}_K$ and, for SEED, in how they select training-time corruptions from the reference-anchored admissible family $\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ in Eqn. (3).

F.1 Task Definitions and Dataset Construction

We evaluate on eight circuit-discovery tasks: Indirect Object Identification (IOI), Capital–Country, Docstring, Subject-Verb Agreement (SVA), Greater-Than, Negation, Gender Bias, and Hypernymy. For each task, we generate 1000 clean–reference pairs $\{(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i})\}_{i=1}^{1000}$ using a task-specific template and the standard benchmark perturbation rule for constructing the reference corruption $\boldsymbol{x}_{\text{ref},i}$. The two prompts in each pair share all tokens except the task-relevant coordinate κ specified by the anchoring rule Γ in Section 3. Representative examples are listed in Table 3. Pairs whose reference gap $\Delta(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i})$ falls below a task-specific minimum are rejected during dataset construction; see Section F.4. Each task is split 8:2 into training and test sets, shared across all methods and edge budgets.

F.2 Task Scores

Each task specifies a scalar task score $m(\cdot)$, where larger values correspond to stronger task-consistent behavior. Representative clean-reference pairs and editable coordinates are shown in Table 3. The corresponding task scores are listed in Table 4.

Held-out evaluation follows the unified reference-corruption protocol. Although SEED may use learned admissible corruptions during training, every final circuit is evaluated on the test set using the benchmark reference corruption. For a learned circuit $c \in \mathcal{C}_K$, we report

$$\widehat{\mathcal{J}}_{\text{fix}}^{\text{test}}(c) \triangleq \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) \in \mathcal{D}_{\text{test}}} \tau(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}; c). \tag{29}$$

Table 4: Task score definitions. Larger values indicate stronger task-consistent behavior.

Task	Score $m(\cdot)$
IOI (Wang et al., 2022)	$\mathrm{logit}(\text{“ John”}) - \mathrm{logit}(\text{“ Mary”})$
Capital–Country	$\mathrm{logit}(\text{“ Albania”}) - \mathrm{logit}(\text{“ United Kingdom”})$
Docstring (Heimersheim & Janiak, 2023)	$\mathrm{logit}(\text{answer}) - \mathrm{logit}(\text{distractor})$
SVA (Newman et al., 2021)	Mean over verb pairs of $\mathrm{logit}(v_{\text{agree}}) - \mathrm{logit}(v_{\text{disagree}})$
Greater–Than (Hanna et al., 2023)	$\log \sum_{y \geq 42} p(y) - \log \sum_{y < 42} p(y)$ over the 100 two-digit year tokens
Negation	$\mathrm{logit}(\text{“ hot”}) - \mathrm{logit}(\text{“ cold”})$
Gender Bias (Vig et al., 2020; Chintam et al., 2023)	$\mathrm{logit}(\text{“ she”}) - \mathrm{logit}(\text{“ he”})$
Hypernymy (Overschelde et al., 2004)	$\sum_{a \in A_{\text{clean}}} \mathrm{logit}(a) - \sum_{a \in A_{\text{ref}}} \mathrm{logit}(a)$

Thus, evaluation measures the quality of the discovered circuit under a shared fixed intervention protocol rather than the ability to choose easier evaluation corruptions.

F.3 Construction of Reference-Anchored Candidate Pools

For each clean-reference pair, the candidate pool is formed by editing the reference corruption at the coordinate κ specified by the anchoring rule Γ, while keeping all other coordinates fixed. After applying the task specific replacement set $\mathcal{R}$ and the admissibility filter in Eqn. (3), the remaining candidates define $\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$.

For all tasks except Greater-Than, replacements are generated with a frozen `roberta-base` masked language model. We mask the editable coordinate in $\boldsymbol{x}_{\text{ref}}$, take the top-100 MLM predictions, retain candidates whose GPT-2 tokenization length matches the reference token, and keep at most 50 candidates per example. The MLM probabilities restricted to this pool define $p_{\text{MLM}}(\cdot \mid \boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$.

For Greater-Than, κ is a two-digit year suffix, and the candidate pool is the set of two-digit year tokens with a uniform prior. In all tasks, the benchmark reference corruption is included in the pool, so $\boldsymbol{x}_{\text{ref}} \in \mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$ by construction.

F.4 Admissibility Filtering

Admissibility is applied at the dataset and candidate levels. During dataset construction, we discard clean-reference pairs whose reference gap is below a task-specific threshold δ_Δ, as required by Assumption 3.1. We use $\delta_\Delta = 0.5$ for IOI, $\delta_\Delta = 0.3$ for Capital–Country, Docstring, Greater-Than, and Gender Bias, and $\delta_\Delta = 0.05$ for SVA, Negation, and Hypernymy.

At the candidate level, admissible corruptions are those satisfying the gap condition in Eqn. (3): $\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}) \geq \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$. Thus, every admissible corruption is at least as behaviorally disruptive as the benchmark reference corruption. The candidate pool is not pre-filtered: during Structure Edit, the selector is normalized over the whole pool and the gap condition is enforced in expectation through the per-pair dual approximation described in Appendix E.2. When decoding the deterministic corruption for Edge Discovery, we select the posterior mode among gap-feasible candidates. If no non-reference candidate is gap-feasible, we use the reference corruption, which is included in the candidate pool by construction.

F.5 Model, Graph, and Edge Budgets

All experiments use `gpt2-small` as the frozen target model $\mathcal{M}$. We represent its forward computation as the directed graph $G = (V, E)$ defined in Section 3. Held-out evaluation uses hard denoising patching with a discrete circuit $c \subseteq E$. For each test pair $(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, edges in c carry clean messages and all other edges carry reference corrupted messages. The patched score is $s(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}; c)$, and recovery is computed using Eqn. (29). We evaluate all methods at $K \in \{30, 50, 100, 300, 500, 1000, 2000, 10000\}$.

F.6 Baseline Implementations

In every baseline, the corruption selector is fixed to the benchmark reference corruption: $\pi_{\text{ref}}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) = \boldsymbol{x}_{\text{ref}}$. Thus, baselines differ only in how they select the circuit $c \in \mathcal{C}_K$.

Random. Random samples K structurally valid edges uniformly at random.

EAP. (Nanda, 2023) EAP scores edges using a single-step gradient-times-activation-difference attribution under the reference corruption and decodes the top-K edges by absolute score.

EAP-IG. (Hanna et al., 2024) EAP-IG replaces the single-step gradient estimate with integrated gradients along the clean–corruption interpolation path. We use five integration steps.

Edge Pruning. (Bhaskar et al., 2024) Edge Pruning learns continuous masks over the edges and nodes of the computation graph by gradient descent.

SEED-Edge. SEED-Edge shares the circuit parameterization, Edge Discovery update, and optimizer with SEED, but disables Structure Edit by fixing the corruption to the benchmark reference throughout training. It corresponds to the singleton-family special case:$\mathcal{F}(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) = \{\boldsymbol{x}_{\text{ref}}\}$. This isolates the effect of optimizing the corruption coordinate.

Baselines not included. We omit ACDC(Conmy et al., 2023) and EAP-GP(Zhang et al., 2025) for reproducibility and computational reasons. ACDC performs iterative edge pruning based on an attribution threshold, which does not directly control the final sparsity and therefore does not naturally match our fixed-K evaluation setting. Moreover, the iterative pruning procedure is prohibitively slow for our full benchmark. EAP-GP is also excluded because **no official open-source implementation** is available, preventing a controlled comparison under the same protocol.

F.7 SEED Experimental Instantiation

We use the training loop in Algorithm 1. SEED and SEED-Edge use the same circuit parameterization, optimizer, training schedule, edge budget, and held-out evaluation protocol. The only difference is that SEED refreshes the corruption selector using Structure Edit, whereas SEED-Edge fixes $\pi(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}}) = \boldsymbol{x}_{\text{ref}}$ throughout training. For SEED, the stochastic selector is computed using the Gibbs form in Theorem 4.1, with the population multiplier replaced by the per-pair multipliers described in Appendix E.2. The KL multiplier is treated as a fixed temperature hyperparameter η, whose value is reported in Table 5. The decoded corruption is the posterior mode, restricted to gap-feasible candidates when possible. All optimization hyperparameters, including learning rate, number of training steps, warmup steps, batch size, gate temperature, and temperature η, are listed in Table 5.

F.8 Hyperparameters and Compute

Table 5 lists the hyperparameters shared by SEED and SEED-Edge. Baselines use the same data, graph, edge-budget grid, and held-out evaluation protocol. Method- specific hyperparameters use standard reference values; for example, EAP-IG uses five integration steps.

The original experiments were run on a single NVIDIA GeForce RTX 5090 GPU, where a single run for one task and one edge budget takes approximately five minutes. The additional baselines and

Table 5: Hyperparameters used in the main experiments. Values are shared between SEED and SEED-EDGE; baselines use the same data, graph, and held-out evaluation, with standard reference values for any method-specific hyperparameter, e.g., five EAP-IG integration steps (Hanna et al., 2024).

Category	Parameter	Value
Model	Backbone	`gpt2-small` (Radford et al., 2019) (frozen)
Dataset	Pairs / split / seeds	1000 per task / $8:2$ / $\{42, 123, 456, 789, 101112\}$
Corruption	Prior	MLM (`roberta-base` (Liu et al., 2019)); uniform on Greater-Than
Corruption	MLM top-n / pool size	100 / ≤ 50
Corruption	Dual-solve tolerance / max iters	10^{-3} / 20
Circuit	Edge budgets K	$\{30, 50, 100, 300, 500, 1000, 2000, 10000\}$
Circuit	Gate temperature τ_c	0.1
Circuit	Warmup steps / penalty weight	200 / 0.5
Training	Steps / batch size	1000 / 4
Training	Optimizer / learning rate	Adam (no scheduler) / 0.05
Training	Outer objective	patch-to-clean alignment loss
Training	Structure Edit temperature η	1.0

noising variants in Section 5 and the Qwen3-0.6B study (Appendix I.1) were run on AMD MI210 GPUs, one GPU per run; wall-time comparisons across methods on a common GPU are reported in Appendix I.2.

G FULL RESULTS FOR MAIN EXPERIMENTS

This section provides the complete results for the main experiments discussed in Section 5.2. While the main text reports condensed results to highlight the main findings, we include the full RQ1 diagnostics, RQ2 budget sweeps, and RQ3 transfer results here. Overall, these extended results are consistent with the main-text conclusions: circuit discovery is conditioned on the discovery corruption, SEED improves held-out recovery under a unified reference-corruption evaluation protocol, and SEED-optimized corruptions are coupled to the circuit optimizer rather than universally transferable. We use the same setup as in Section 5.1; additional implementation details are provided in Appendix F.

G.1 FULL RESULTS FOR RQ1: CORRUPTION CONDITIONING

This section reports the complete diagnostic results for RQ1. We fix the discovery method to EAP and vary only the admissible corruption selector used during discovery, while keeping the model, split, and edge budget fixed. Table 6 summarizes the cross-discovery/evaluation protocol.

Diagnostic protocol. For each task, we run EAP with multiple admissible discovery selectors and evaluate the resulting hard top-K circuits under a shared set of admissible evaluation selectors. Table 6 summarizes the protocol.

The `reference` selector uses the benchmark reference corruption. The `rank1`–`rank4` selectors choose the top admissible non-reference alternatives under the frozen MLM prior, and `prior_sample_01`–`prior_sample_08` sample from the same admissible family. Unavailable alternatives fall back to the benchmark reference corruption.

Metrics. Let $c_\pi^{(K)}$ be the hard top-K circuit discovered under discovery selector π, and let $R_{\pi,\sigma}$ be its held-out normalized recovery under evaluation selector σ. We measure structural sensitivity using

$$\mathrm{EdgeReplacement@}K(\pi, \pi') = 1 - \frac{\left|c_\pi^{(K)} \cap c_{\pi'}^{(K)}\right|}{K}.$$

Table 7 reports EdgeReplacement@300 averaged over `reference`-versus-`rank1`–`rank4` pairs:

$$\frac{1}{4}\sum_{j=1}^{4} \mathrm{EdgeReplacement@300}(\texttt{reference}, \texttt{rank}j).$$

Table 6: Protocol for the RQ1 corruption-conditioning diagnostic. All factors are fixed except the admissible corruption selector used during discovery.

Item	Setting
Discovery method	EAP (Nanda, 2023)
Target model	`gpt2-small`
Edge budget	$K = 300$
Tasks	IOI (Wang et al., 2022), Capital-Country, Docstring (Heimersheim & Janiak, 2023), SVA (Newman et al., 2021), Greater-Than (Hanna et al., 2023), Negation, Gender Bias (Vig et al., 2020; Chintam et al., 2023), Hypernymy (Overschelde et al., 2004)
Seeds	$42, 123, 456, 789, 101112$
Discovery selectors	{`reference`, `rank1`, `rank2`, `rank3`, `rank4`}
Evaluation selectors	{`reference`, `rank1`, ..., `rank4`, `prior_sample_01`, ..., `prior_sample_08`}

Table 7: Task-level corruption-conditioning diagnostics for the single-method EAP diagnostic at K=300. Means ± standard errors over five random seeds; standard errors are typeset smaller for readability. EdgeRepl.@300 is the primary structural metric: it measures the fraction of selected top-300 edges replaced when only the admissible discovery corruption is changed. MMG and RTR are functional-profile diagnostics computed from the cross-discovery/evaluation recovery matrix. The valid-alternative and fallback rates quantify how much room each task has for reference-anchored corruption variation.

Task	MMG	RTR	EdgeRepl.300	Valid-alt	Fallback
IOI	0.0022 ± 0.0058	-0.1538 ± 0.0421	0.1485 ± 0.0039	0.976	0.036
Capital–Country	-0.0001 ± 0.0001	-0.0006 ± 0.0063	0.0168 ± 0.0006	0.097	0.915
Docstring	0.0151 ± 0.0052	0.0222 ± 0.0032	0.1467 ± 0.0044	0.978	0.031
SVA	-0.0059 ± 0.0031	-0.0884 ± 0.0218	0.1308 ± 0.0032	0.798	0.236
Greater-Than	0.0152 ± 0.0022	0.0127 ± 0.0017	0.3182 ± 0.0019	0.985	0.021
Negation	-0.0137 ± 0.0052	-0.0366 ± 0.0148	0.3222 ± 0.0030	0.988	0.021
Gender Bias	-0.0011 ± 0.0006	0.0121 ± 0.0094	0.0245 ± 0.0009	0.417	0.670
Hypernymy	0.0001 ± 0.0001	-0.0024 ± 0.0047	0.0287 ± 0.0017	0.174	0.848

We also report two recovery-profile diagnostics. With $\pi_0 =$ `reference` and $\pi_j =$ `rank`j, the matched–mismatched gap is

$$\mathrm{MMG} = \frac{1}{4}\sum_{j=1}^{4}\left[\frac{R_{\pi_0,\pi_0} + R_{\pi_j,\pi_j}}{2} - \frac{R_{\pi_0,\pi_j} + R_{\pi_j,\pi_0}}{2}\right],$$

and the reference-trained regret is

$$\mathrm{RTR} = \frac{1}{4}\sum_{j=1}^{4}\left(R_{\pi_j,\pi_j} - R_{\pi_0,\pi_j}\right).$$

MMG and RTR are diagnostic statistics only; they are not optimization objectives. We also report valid_alt_rate and fallback_rate to indicate how often a task admits non-reference corruptions.

Task-level results. Table 7 confirms the trend reported in Section 5.2. Across all eight tasks, changing only the discovery corruption replaces 14.2% of the selected top-300 edges on average, with the largest effects on Negation and Greater-Than. The lower replacement rates on Capital-Country and Hypernymy coincide with lower valid-alternative rates and higher fallback rates, suggesting that these tasks provide less room for corruption-conditioned variation.

The lower replacement rates on Capital-Country and Hypernymy are consistent with their limited admissible-alternative support. These tasks have low valid-alternative rates and high fallback rates, so the selector often cannot deviate from the benchmark reference corruption. Thus, their lower instability reflects a boundary condition of the admissible family rather than evidence that the discovery procedure is generally corruption-invariant.

Table 8: Numeric values for the representative Greater-Than recovery matrix in Figure 5. Entries are held-out normalized recoveries for circuits discovered under the row selector and evaluated under the column selector at K=300.

Discovery selector	`reference`	`rank1`	`rank2`	`rank3`	`rank4`
`reference`	0.892	0.851	0.866	0.884	0.886
`rank1`	0.825	0.828	0.828	0.851	0.842
`rank2`	0.832	0.811	0.852	0.873	0.851
`rank3`	0.922	0.884	0.898	0.927	0.917
`rank4`	0.917	0.889	0.894	0.929	0.942

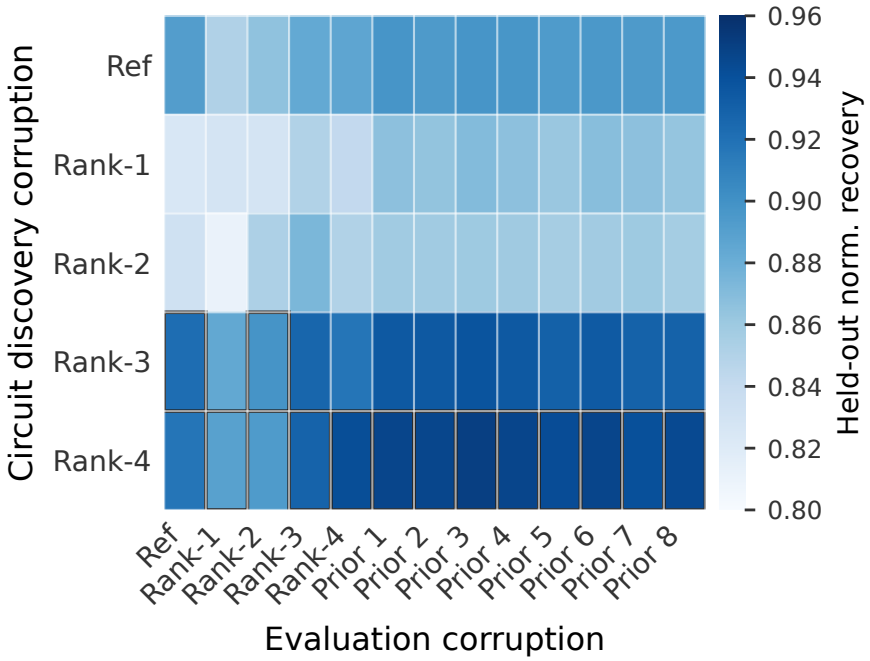


Figure 5: Representative cross-discovery/evaluation recovery matrix for the corruption-conditioning diagnostic on Greater-Than. Rows are EAP circuits discovered under different admissible discovery selectors, and columns are admissible evaluation selectors. Entries report held-out normalized recovery. Different row profiles indicate that the discovery corruption changes the recovered circuit.

Representative recovery matrix. Figure 5 and Table 8 provide a representative cross-discovery/evaluation recovery matrix on Greater-Than. Rows correspond to discovery selectors, columns correspond to evaluation selectors, and entries report held-out normalized recovery.

The recovery matrix shows that circuits discovered under different admissible corruptions do not have identical evaluation profiles. In this example, the reference-discovered circuit is not always the best circuit under alternative evaluations. This supports the claim that the benchmark reference corruption is one slice of a broader circuit-corruption landscape.

Aggregate instability. For each evaluation selector σ, let $w(\sigma) = \arg\max_{\pi \in \Pi_{\text{disc}}} R_{\pi,\sigma}$ be the discovery selector whose circuit obtains the highest recovery under σ. The winner-change rate is

$$\text{WinnerChange} = \frac{2}{|\Pi_{\text{eval}}|(|\Pi_{\text{eval}}|-1)} \sum_{\sigma<\sigma'} \mathbf{1}[w(\sigma) \neq w(\sigma')] \,.$$

We also compute rank instability using the normalized pairwise ranking disagreement

$$D_{\text{rank}}(\sigma,\sigma') = \frac{2}{|\Pi_{\text{disc}}|(|\Pi_{\text{disc}}|-1)} \sum_{\{\pi,\pi'\}\subset\Pi_{\text{disc}}} \mathbf{1}\left[(R_{\pi,\sigma} - R_{\pi',\sigma})\,(R_{\pi,\sigma'} - R_{\pi',\sigma'}) < 0\right],$$

Table 9: Aggregate corruption-conditioning statistics across the eight tasks. Rank instability is the average pairwise ranking disagreement between evaluation selectors; the winner-change rate is the fraction of selector pairs whose top circuit differs; $\mathrm{EdgeReplacement@300}$ is the average fraction of selected top-300 edges replaced when the discovery corruption changes.

Statistic	Value
Average rank instability	0.053
Winner-change rate	0.267
Average $\mathrm{EdgeReplacement@300}$	0.142

Table 10: Held-out Completeness per task for all methods in Table 1 (same circuits and budgets). Each cell averages the eight edge budgets and reports mean ± standard deviation over seeds.

Method	IOI	Capital	Docstring	SVA	Greater	Negation	Gender	Hypernymy	Avg.
EAP	0.951 ± 0.002	0.992 ± 0.000	0.888 ± 0.019	0.991 ± 0.001	0.996 ± 0.000	0.941 ± 0.007	**0.996** ± 0.001	0.992 ± 0.001	0.968 ± 0.003
EAP-IG	**0.987** ± 0.001	**0.994** ± 0.001	**0.977** ± 0.001	**0.997** ± 0.000	0.998 ± 0.000	0.969 ± 0.005	0.995 ± 0.001	**0.995** ± 0.000	**0.989** ± 0.001
Edge Pruning	0.940 ± 0.021	0.781 ± 0.027	0.852 ± 0.037	0.898 ± 0.017	0.959 ± 0.013	0.810 ± 0.024	0.885 ± 0.034	0.750 ± 0.021	0.859 ± 0.014
SEED-EDGE (D)	0.716 ± 0.056	0.987 ± 0.003	0.949 ± 0.001	0.838 ± 0.018	0.997 ± 0.000	0.951 ± 0.009	0.942 ± 0.015	0.988 ± 0.005	0.921 ± 0.006
SEED (D)	0.980 ± 0.008	0.987 ± 0.004	0.803 ± 0.009	0.956 ± 0.043	0.996 ± 0.001	0.949 ± 0.002	0.946 ± 0.006	0.990 ± 0.002	0.951 ± 0.003
SEED-EDGE (N)	0.950 ± 0.019	0.979 ± 0.011	0.862 ± 0.020	0.985 ± 0.003	**0.998** ± 0.000	**0.974** ± 0.002	0.988 ± 0.003	0.993 ± 0.001	0.966 ± 0.005
SEED (N)	0.958 ± 0.012	0.978 ± 0.011	0.969 ± 0.007	0.986 ± 0.006	0.998 ± 0.000	0.928 ± 0.009	0.993 ± 0.001	0.992 ± 0.003	0.975 ± 0.002
Δ (D): SEED − SEED-EDGE	+0.264	-0.000	-0.147	+0.118	-0.001	-0.002	+0.004	+0.002	+0.030
Δ (N): SEED − SEED-EDGE	+0.009	-0.001	+0.107	+0.001	-0.000	-0.046	+0.005	-0.000	+0.009

averaged over pairs of evaluation selectors.

Table 9 reports the aggregate instability statistics corresponding to Figure 2a. The winner circuit changes in 26.7% of evaluation comparisons, confirming that corruption choice can affect both the selected edge set and the ranking of recovered circuits. Together with the edge-replacement results, these complete diagnostics support the RQ1 conclusion that patching-based circuit discovery is materially conditioned on the corruption used during discovery.

G.2 Full Results for RQ2: Main Held-Out Performance

This section complements Table 1 with per-task Completeness, budget curves for both metrics, and normalized-recovery results under the original protocol.

Bidirectional protocol and sources. Every hard top-K circuit is frozen after discovery and evaluated on the 200 held-out examples of each task under the benchmark reference corruption, in both the denoising (Faithfulness) and noising (Completeness) directions. All methods use the same graph, splits, task scores, patching protocol, and exact budgets $K \in \{30, 50, 100, 300, 500, 1000, 2000, 10000\}$, with seeds 42, 123, and 456. EAP, EAP-IG, and Edge Pruning are run once per seed; SEED (D) and SEED-EDGE (D) use 1,000 optimization steps, as in the original five-seed protocol; the noising variants use 1,500 steps. Per-task values average the eight budgets within each seed; task averages average the eight tasks within each seed; we report mean ± sample standard deviation across seeds.

Budget curves. Figure 6 reports task-averaged Faithfulness and Completeness as functions of the edge budget for all seven methods, and Figures 7 and 8 report the per-task curves. SEED (D) has the highest average Faithfulness for $K \leq 100$ and is on par with Edge Pruning at $K = 300$; Edge Pruning is strongest for $K \geq 500$ but recovers almost nothing for $K \leq 100$, and at the largest budget the fixed-corruption attribution baselines also exceed both SEED variants. Completeness is close to saturation for most methods at $K \geq 300$; the differences in task-averaged Completeness in Table 1 come mainly from small budgets. Noising-aligned variants attain high Completeness at small budgets but low Faithfulness at all budgets below $K = 10{,}000$.

Original five-seed normalized recovery. For reference, Figure 9 and Table 11 report held-out normalized recovery under the original denoising-only protocol with five seeds and a Random baseline.

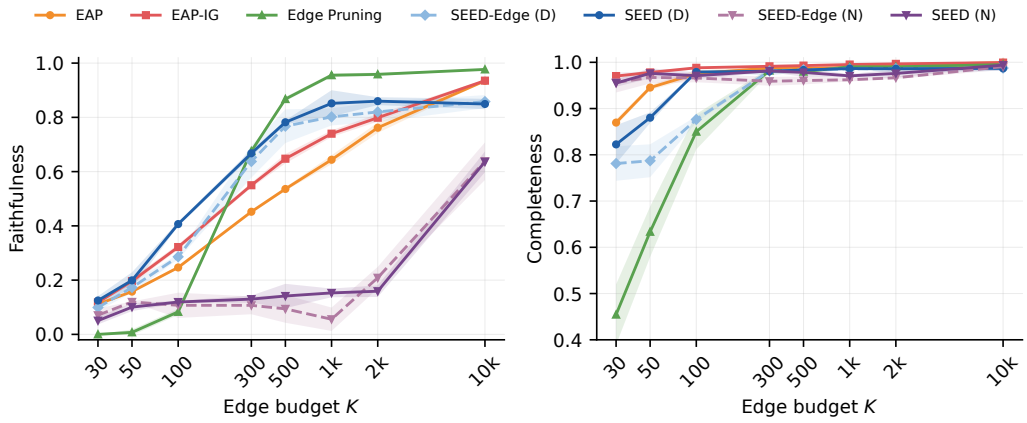


Figure 6: Task-averaged held-out Faithfulness (left) and Completeness (right) versus edge budget for all methods in Table 1. Shaded regions: ± one standard deviation across seeds.

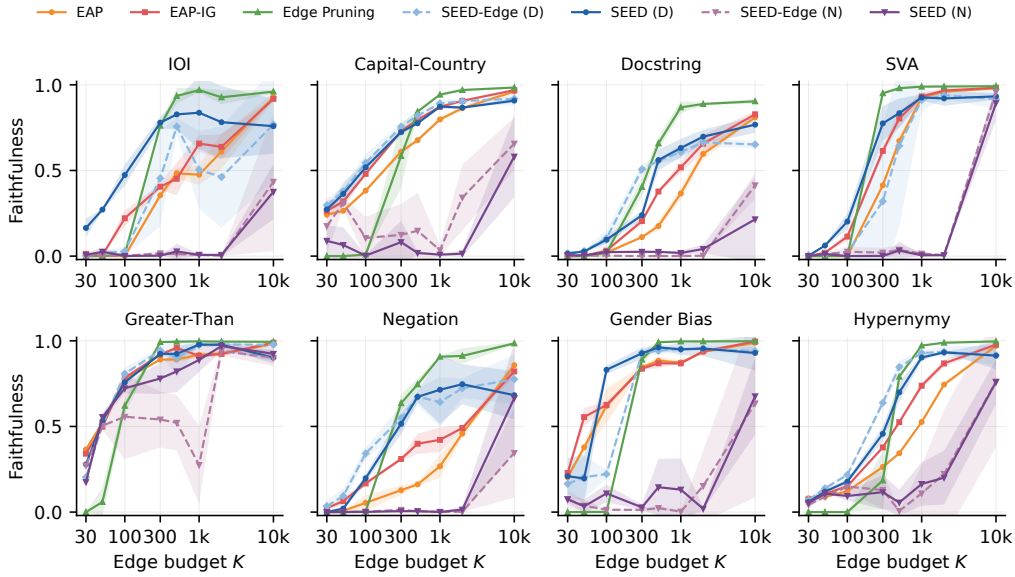


Figure 7: Per-task held-out Faithfulness versus edge budget. Shaded regions: ± one standard deviation across seeds.

Under this protocol, SEED achieves the strongest aggregate held-out recovery, and its advantage over SEED-EDGE, which shares the same edge optimizer and hard top-K decoding, varies across tasks and budgets, consistent with the bidirectional results.

G.3 FULL RESULTS FOR RQ3: CROSS-EVALUATION OF SEED-OPTIMIZED CORRUPTIONS

Table 12 and Table 13 provide the full transfer results for RQ3 in Section 5.2. The experiment tests whether the corruptions selected by SEED can be used as generally better training corruptions for non-SEED methods. The results show mixed and often negative transfer, supporting the conclusion that these corruptions are coupled to the circuit optimizer that selected them.

Transfer protocol. We use the corruptions selected by SEED at source budget $K = 300$. For each task and seed, these corruptions define an optimized training set $\{(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}^{\star}_{\text{cor},i})\}_{i=1}^{n_{\text{train}}}$, where $\boldsymbol{x}^{\star}_{\text{cor},i}$ is the decoded corruption selected by Structure Edit. We then train EAP and EAP-IG under

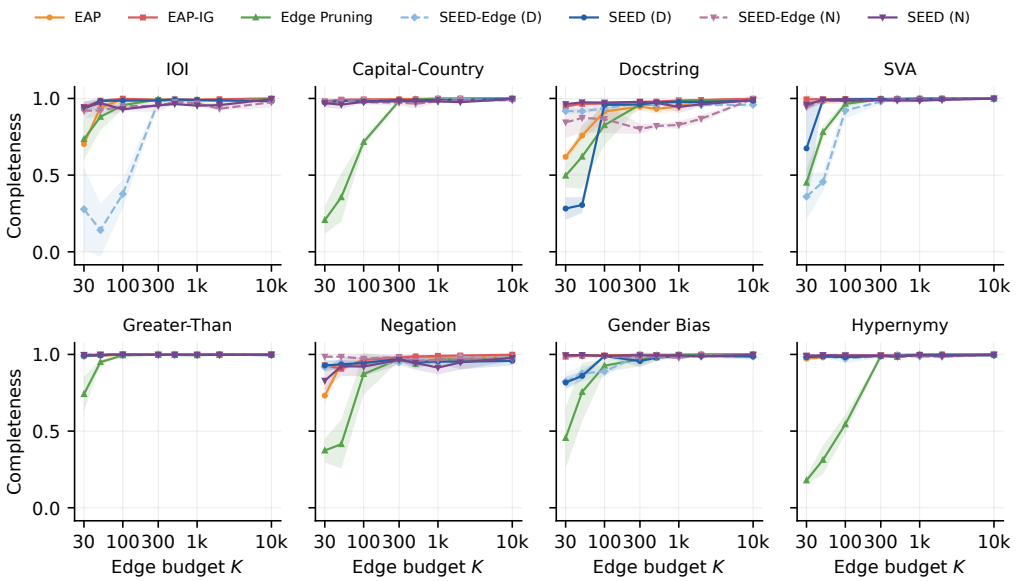


Figure 8: Per-task held-out Completeness versus edge budget. Shaded regions: ± one standard deviation across seeds.

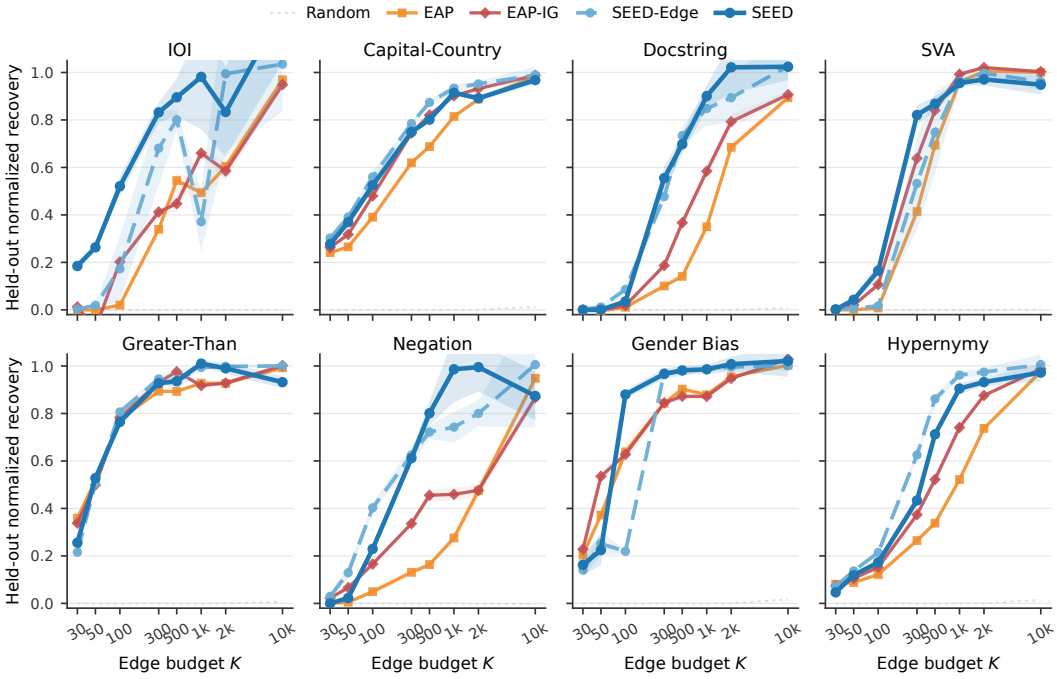


Figure 9: Per-task held-out normalized recovery versus edge budget under the original five-seed protocol. Curves show means over five random seeds, and shaded regions denote standard errors. All methods are evaluated under the same held-out benchmark reference corruptions.

two training-corruption conditions:

$$\text{REFERENCE-TRAIN} : (\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i}), \qquad \text{OPTIMIZED-TRAIN} : (\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}^{\star}_{\text{cor},i}).$$

Both resulting circuits are evaluated on the same held-out benchmark reference corruptions. Thus, the optimized-train condition changes only the training corruption, not the evaluation distribution.

Table 11: Aggregate held-out normalized recovery by edge budget, averaged across all eight tasks and five random seeds. The values aggregate the per-task curves in Figure 9. Best results in each column are shown in bold.

Method	K=30	K=50	K=100	K=300	K=500	K=1000	K=2000	K=10000
Random	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.007
EAP	0.111	0.156	0.252	0.451	0.546	0.653	0.784	0.971
EAP-IG	**0.117**	0.183	0.317	0.558	0.663	0.766	0.820	0.966
SEED-EDGE	0.096	0.182	0.310	0.705	0.832	0.850	0.951	**1.003**
SEED	0.116	**0.196**	**0.412**	**0.737**	**0.837**	**0.955**	**0.955**	1.001

Table 12: Protocol for the RQ3 transfer experiment. SEED-optimized corruptions are taken from the source SEED run at $K = 300$, while all resulting circuits are evaluated under the benchmark reference corruption.

Item	Setting
Transferred corruptions	SEED-optimized training corruptions
Source budget	$K = 300$
Transfer methods	EAP, EAP-IG
Tasks	IOI, Capital-Country, Docstring, SVA, Greater-Than, Negation, Gender Bias, Hypernymy
Seeds	$42, 123, 456, 789, 101112$
Evaluated budgets	$30, 50, 100, 300, 500, 1000, 2000, 10000$
Evaluation corruption	Fixed reference corruption
Training conditions	Reference-train vs. optimized-train

Results at the source budget. Table 13 reports the transfer comparison at $K = 300$, the budget from which the SEED-optimized corruptions were obtained.

The $K = 300$ results show that SEED-optimized corruptions are not universally useful for non-SEED methods. For EAP, optimized training improves Negation and Gender Bias but reduces recovery on the other six tasks, giving an average transfer delta of -0.0589. For EAP-IG, optimized training improves IOI and Capital-Country slightly but reduces recovery on the remaining six tasks, giving an average transfer delta of -0.0407.

These results are consistent with the RQ3 conclusion in Section 5.2. They suggest that the usefulness of a corruption depends on the circuit state and optimizer that use it. Thus, Structure Edit should be viewed as part of joint circuit–corruption optimization, not as a preprocessing step that finds a globally better corruption dataset.

H ABLATIONS

This appendix provides ablations for the main design choices in SEED. All ablations evaluate the decoded hard top-K circuit under the held-out reference corruptions. Appendices H.1–H.3 use held-out normalized recovery under the original protocol, and Appendix H.4 uses Faithfulness and Completeness.

H.1 SURROGATE FIDELITY: MATCHING TRAINING TO DISCRETE EVALUATION

This section reports ablations for the soft-patching surrogate described in Appendix E.1. The goal is to verify that the train time surrogate remains aligned with the decoded hard top-K circuit used at evaluation. Across the ablations, the strongest results are obtained when the gate is budget-aware, uses a hard top-K forward pass, and keeps a differentiable relaxation only for the backward pass.

Gate parameterization. Figure 10 compares five soft-gate parameterizations for mapping edge logits ω_e to train-time patching gates. All variants are evaluated using the decoded hard top-K circuit at $K = 300$, so the comparison tests whether each training surrogate produces a circuit that matches

Table 13: Transfer of SEED-optimized corruptions at K=300. "Ref" trains the method on benchmark reference corruptions; "Opt" trains the same method on SEED-optimized corruptions from the source K=300 run. Both conditions are evaluated under held-out benchmark reference corruptions; values average over five paired seeds. Negative deltas dominate, indicating that the optimized corruptions are not universally beneficial as training data for non-SEED discovery methods.

Task	EAP			EAP-IG		
	Ref	Opt	Δ	Ref	Opt	Δ
IOI	0.339	-0.007	-0.346	0.412	0.435	0.022
Capital-Country	0.620	0.577	-0.043	0.745	0.749	0.004
Docstring	0.101	0.085	-0.016	0.187	0.131	-0.055
SVA	0.414	0.311	-0.104	0.638	0.534	-0.105
Greater-Than	0.894	0.838	-0.056	0.931	0.900	-0.031
Negation	0.131	0.196	0.065	0.336	0.289	-0.047
Gender Bias	0.843	1.000	0.158	0.845	0.798	-0.047
Hypernymy	0.265	0.135	-0.130	0.374	0.306	-0.068
Average	–	–	-0.059	–	–	-0.041
Positive tasks	–	–	2/8	–	–	2/8

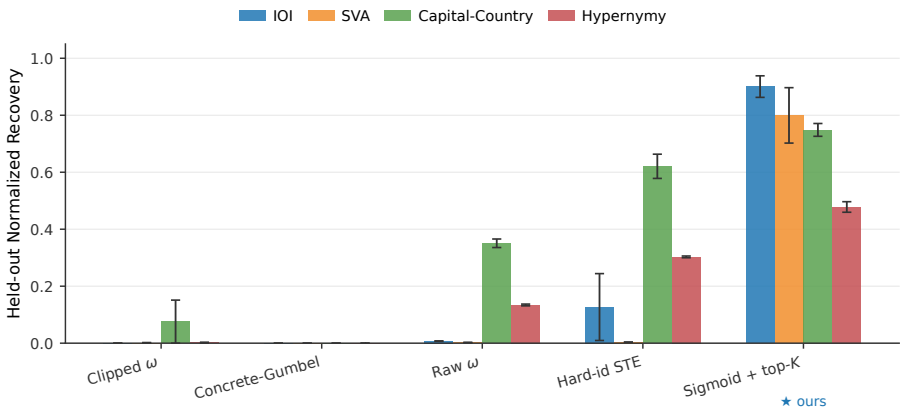


Figure 10: Held-out normalized recovery under five train-time gate parameterizations. All variants use the same Edge Discovery loss, edge budget $K = 300$, and evaluation on decoded hard top-K circuits. Only the mapping from edge logits ω_e to train-time patching gates changes.

the final evaluation rule. The compared gate mechanisms are:

$$\begin{aligned}
\text{Raw logits:} \quad & m_e = \omega_e, \\
\text{Clipped logits:} \quad & m_e = \mathrm{clip}(\omega_e, 0, 1), \\
\text{Concrete-Gumbel:} \quad & m = \mathrm{SoftTopK}_{K,\tau_c}(\boldsymbol{\omega} + \boldsymbol{g}), \qquad g_e \sim \mathrm{Gumbel}(0,1), \\
\text{Hard-identity STE:} \quad & m_e = \left[\mathrm{TopKMask}_K(\boldsymbol{\omega})\right]_e \quad \text{with identity backward}, \\
\text{Sigmoid top-}K\text{ projection:} \quad & \alpha_e = m_e^{\mathrm{hard}} + \widetilde{m}_e - \mathrm{sg}[\widetilde{m}_e], \quad \widetilde{m}_e = \sigma\left(\frac{\omega_e - \theta_K(\boldsymbol{\omega})}{\tau_c}\right).
\end{aligned}$$

The complete results show that the sigmoid-with-top-K projection is the most consistent parameterization across tasks. Other soft-gate mechanisms can work in some settings, but they are less stable when the train-time gates are not explicitly aligned with the hard top-K evaluation circuit. This supports the surrogate design in Appendix E.1, where the forward pass matches the decoded circuit and the soft relaxation is used only to provide gradients.

Sharpness of the top-K relaxation. Figure 11 reports the temperature sweep for the top-K relaxation in Eqn. (19). This ablation checks whether the surrogate depends on a finely tuned temperature.

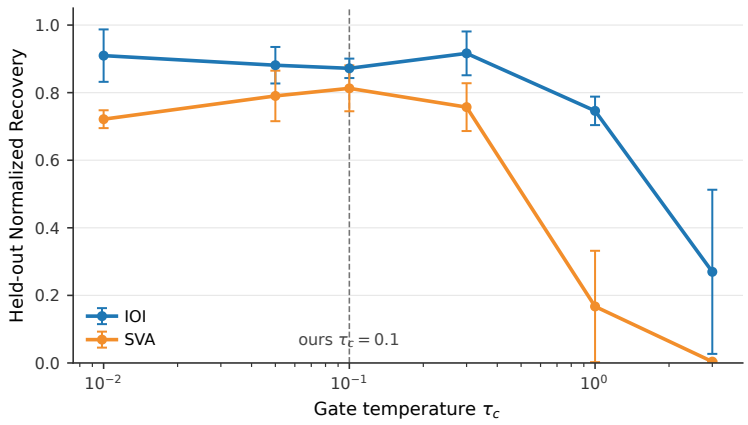


Figure 11: Held-out normalized recovery as a function of the top-K relaxation temperature τ_c on IOI and Subject-Verb Agreement at $K = 300$. All other settings use the default SEED configuration.

Table 14: Train/eval configurations used in the surrogate-fidelity ablation. Each variant changes whether patching uses the soft relaxation $\widetilde{m}$ or the decoded hard top-K mask during training and held-out evaluation.

Variant	Train forward	Train backward	Held-out eval
Soft eval no projection	$\widetilde{m}$	soft gate	soft gate
Soft-fwd + top-K	$\widetilde{m}$	soft gate	hard top-K
Hard-id STE	m^{hard}	identity STE	hard top-K
Hard-fwd / soft-bwd	m^{hard}	soft top-K	hard top-K

The results show a stable range for $\tau_c \in [10^{-2}, 3\times10^{-1}]$, suggesting that performance is not sensitive to the exact temperature within the near-binary regime. Recovery drops when $\tau_c \geq 1$, where the relaxation becomes too smooth to approximate the hard top-K mask. This identifies a clear boundary condition: the surrogate is robust to reasonable temperature choices, but it must remain sufficiently sharp to match the discrete evaluation circuit.

Train/eval forward separation. Figure 12 compares four ways of combining train-time patching gates with held-out evaluation. Let

$$\widetilde{m}_e = \sigma\left(\frac{\omega_e - \theta_K(\boldsymbol{\omega})}{\tau_c}\right)$$

be the soft top-K relaxation, and let $m^{\text{hard}} = \text{TopKMask}_K(\boldsymbol{\omega})$. Table 14 summarizes the variants.

The default setting is hard-forward / soft-backward:

$$m_e = m_e^{\text{hard}} + \widetilde{m}_e - \text{sg}[\widetilde{m}_e].$$

Thus, training uses the decoded budget-K circuit in the forward pass while routing gradients through the soft relaxation. All variants except soft eval no projection compute held-out recovery after decoding $\boldsymbol{\omega}$ into $c_K(\boldsymbol{\omega})$.

The results favor the hard-forward / soft-backward setting. Soft evaluation gives weaker recovery, confirming that the relaxed mask should not replace the decoded circuit at test time. This supports the surrogate design used in SEED: soft gates provide gradients during training, while reported recovery is measured on the discrete top-K circuit.

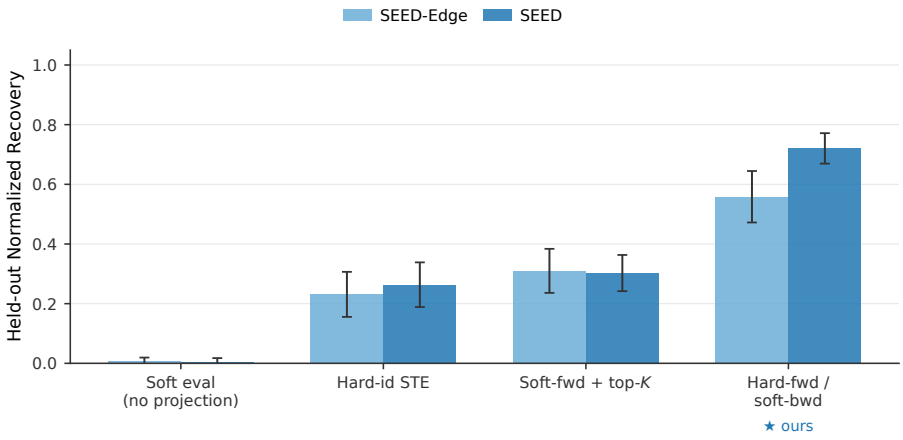


Figure 12: Held-out normalized recovery for different train/eval forward-pass configurations, evaluated for SEED and SEED-EDGE. All variants use the same Edge Discovery loss and edge budget $K = 300$.

H.2 MECHANISM ABLATIONS: WHEN DOES STRUCTURE EDIT HELP?

This section reports ablations for the main mechanism choices in SEED: the Edge Discovery loss, Structure Edit, and the construction of the admissible candidate family $\mathcal{F}$.

All variants are evaluated using held-out normalized recovery on the decoded hard top-K circuit under the benchmark reference corruptions. The results support a conditional interpretation of the method: Structure Edit is most useful when the admissible family provides informative alternatives, and the candidate family must remain both reference-anchored and semantically plausible.

Edge Discovery loss. Figure 13 compares the default patch-to-clean alignment loss in Eqn. (27) with a direct utility-maximization objective. All settings are held fixed except the Edge Discovery loss. For the utility-max variant, we replace Eqn. (27) with

$$\mathcal{L}_{\text{util}}(\boldsymbol{\omega}; \mathcal{B}) = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left[s(\boldsymbol{x}_{\text{clean},i}, \widehat{\boldsymbol{x}}_{\text{cor},i}; c_K(\boldsymbol{\omega})) - m(\widehat{\boldsymbol{x}}_{\text{cor},i}) \right]. \tag{30}$$

This objective directly rewards increasing the patched score relative to the corrupted score, but unlike the patch-to-clean loss, it does not anchor the optimum at the clean full-model score.

The patch-to-clean loss keeps recovery near the validity range across budgets, while direct utility maximization grows far beyond normalized recovery 1, reaching 22.51 at $K = 10000$. This confirms that maximizing patched score directly can produce severe overshoot rather than recovery of the clean target. The result supports the use of the patch-to-clean alignment loss, which penalizes both under-recovery and overshoot.

Structure Edit and admissible-family richness. Figure 14 compares SEED with SEED-EDGE on all eight tasks at $K = 300$. Both variants use the same edge graph, candidate family, Edge Discovery loss, and hard top-K decoding; only SEED updates the corruption selector through Structure Edit.

The eight-task comparison shows a task-dependent effect. The largest positive gains appear on Subject-Verb Agreement, IOI, and Docstring, where the admissible family contains useful alternative contrasts. On Capital-Country, Greater-Than, Negation, and Gender Bias, SEED tracks SEED-EDGE closely, consistent with more template-like or fixed corruption constructions that leave less room for the learned selector to improve the training contrast. Hypernymy is the main negative case, suggesting that Structure Edit can select valid alternatives that are less aligned with the held-out reference corruption when the admissible family is narrow or semantically heterogeneous. Overall, learned corruptions are useful as circuit-coupled training contrasts, but they should not be interpreted as a universally better corruption dataset.

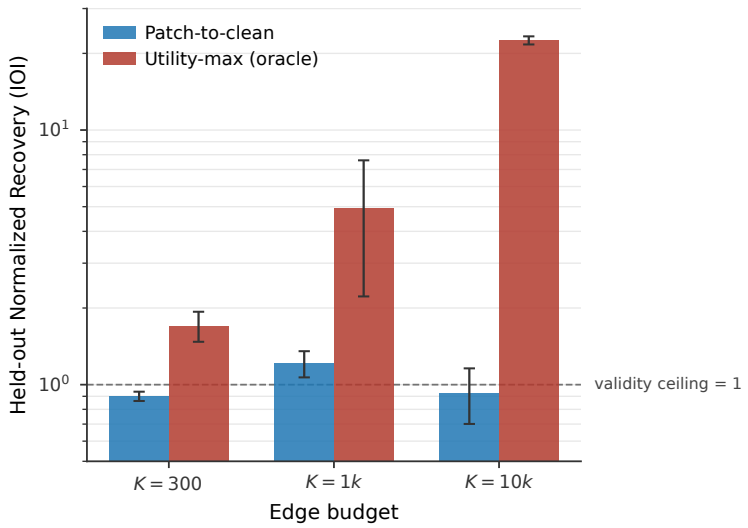


Figure 13: Held-out normalized recovery on IOI for two Edge Discovery losses across $K \in \{300, 1000, 10000\}$. Both variants use the same Structure Edit refresh, admissible family, and training schedule. The default variant uses the patch-to-clean loss in Eqn. (27); the ablation uses the utility-max loss in Eqn. (30).

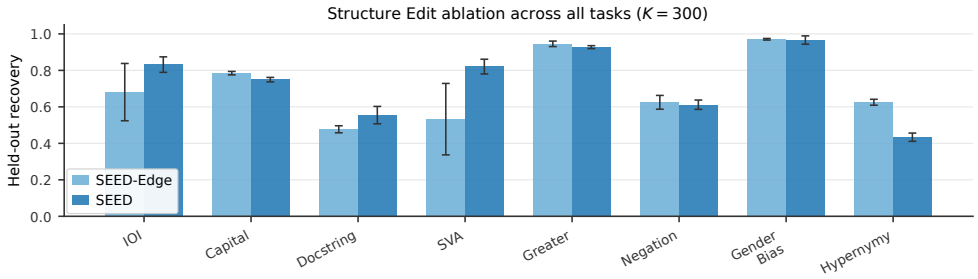


Figure 14: Full $K = 300$ Structure Edit ablation across all eight tasks. Bars show held-out normalized recovery for SEED-EDGE and SEED, evaluated under the same benchmark reference corruptions. Values are means over five random seeds; error bars show one standard error.

Candidate-family construction. Figure 15 compares five constructions of the admissible candidate family, averaged across IOI, Subject-Verb Agreement, Capital-Country, and Negation at $K = 300$.

All variants use the same Edge Discovery loss, Structure Edit refresh, and edge budget; only the candidate family changes.

The MLM-pruned reference-anchored family achieves the best average recovery (0.77), outperforming the frequency prior (0.57), reference-only family (0.55), full-vocabulary expansion (0.52), and random vocabulary pool (0.49). The result shows that simply enlarging the corruption space is not sufficient: full-vocabulary and random-vocabulary variants perform worse despite providing larger or similarly sized candidate sets. It also shows that using only the reference corruption removes much of the benefit of Structure Edit. Overall, the ablation supports the candidate family design used in SEED: the family should be anchored to the benchmark reference while using a contextual prior to retain semantically plausible alternatives.

H.3 Optimization Robustness and Parameterization Choices

This section reports robustness checks for two implementation choices that affect optimization stability and circuit expressivity: the warmup schedule and the gate granularity. The results show that SEED is stable once the warmup mechanism is enabled, and that edge-level gates are important for recovering sparse circuits with fine-grained routing structure.

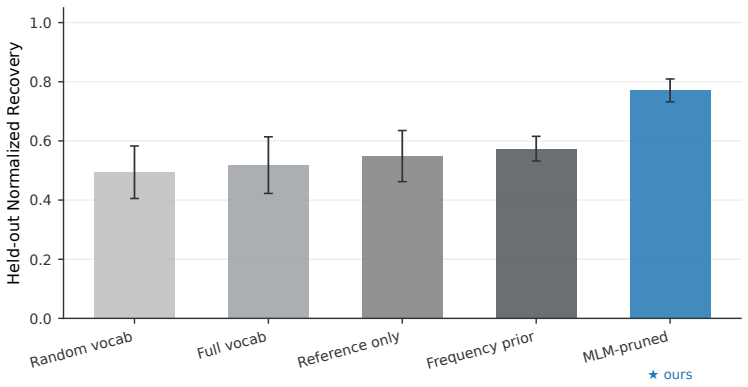


Figure 15: Held-out normalized recovery for five candidate-family constructions, averaged across IOI, Subject-Verb Agreement, Capital-Country, and Negation at $K = 300$. All variants use the same Edge Discovery loss, Structure Edit refresh, and edge budget. Only the construction of the admissible candidate family changes.



Figure 16: Held-out normalized recovery as a function of warmup length T_{warm} and budget penalty weight λ_{warm}. Values are averaged over IOI and Subject-Verb Agreement at $K = 300$. All other settings use the default SEED configuration; the default setting $(T_{\text{warm}}, \lambda_{\text{warm}}) = (200, 0.5)$ is outlined.

Warmup phase and budget penalty. Figure 16 varies the warmup length T_{warm} and the budget penalty weight λ_{warm}. The reported value is the mean held-out normalized recovery over IOI and Subject-Verb Agreement at $K = 300$.

The heatmap shows that both warmup and budget regularization are needed. When $T_{\text{warm}} = 0$, recovery remains low for all budget penalties. When $\lambda_{\text{warm}} = 0$, recovery also remains low even with warmup. Once both are enabled, recovery is stable across a range of settings: $T_{\text{warm}} \geq 50$ with $\lambda_{\text{warm}} = 0.5$ reaches 0.78–0.86, and the default setting achieves 0.84, within 0.02 of the best cell. A larger penalty, $\lambda_{\text{warm}} = 5$, is consistently weaker, suggesting that overly strong budget pressure may concentrate gate mass too early.

These results support the default warmup design. Warmup avoids degenerate early masks, while the budget penalty keeps the soft gate mass aligned with the target sparsity before hard top-K decoding. The method is not highly sensitive to the exact warmup length once both components are active.

Gate granularity. Figure 17 compares the default per-edge gates with a head-tied variant that shares one gate across the q, k, and v destination slots of each attention head. All other settings use the default SEED configuration.

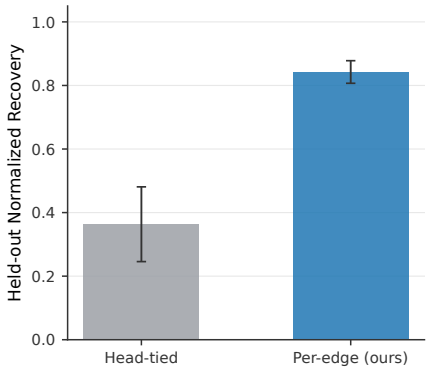


Figure 17: Held-out normalized recovery for per-edge gates and head-tied gates on IOI and Subject-Verb Agreement at $K = 300$. The per-edge setting assigns a separate gate to each directed edge. The head-tied setting shares one gate across the q, k, and v destination slots of each attention head.

Per edge gates substantially outperform head tied gates, with average recovery 0.84 versus 0.36. This confirms that the gate parameterization must preserve edge-level routing distinctions. Tying the q, k, and v destination slots into a single gate reduces the expressivity of the recovered circuit and weakens held-out recovery. Thus, the edge-level parameterization is important for matching the granularity of the patching evaluation.

H.4 Sensitivity to the Admissible Candidate Pool

We vary two choices that shape the admissible family: the number of prior candidates requested from the proposal model (top-n) and the disruptiveness threshold. Both sweeps use `gpt2-small`, SEED (D), $K = 300$, 1,500 steps, and three seeds on four tasks (IOI, Capital-Country, SVA, Negation); values are task averages with standard deviations across seeds. The default setting is $n = 50$ and $\rho = 1$.

Top-n. Table 15 shows that Completeness is stable across $n \in \{10, \ldots, 10{,}000\}$, while Faithfulness varies moderately and non-monotonically; the default $n = 50$ is not the best setting. Increasing n from 10 to 10,000 adds approximately 3.8% mean runtime.

Disruptiveness threshold. We replace the admissibility constraint by $\Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{cor}}) \geq \rho\, \Delta(\boldsymbol{x}_{\text{clean}}, \boldsymbol{x}_{\text{ref}})$, where smaller ρ admits less disruptive candidates. Table 16 shows that intermediate thresholds improve average Faithfulness over the strict default, whereas removing the floor ($\rho = 0$) reduces it substantially. The floor therefore prevents weak or degenerate contrasts, while the strict setting $\rho = 1$ is conservative rather than optimal. The effect varies across tasks: the gain at intermediate ρ is driven mainly by SVA, whereas IOI is best at $\rho = 1$.

Table 15: Top-n sweep.

n	Faithfulness	Completeness
10	0.660 ± 0.067	0.989 ± 0.004
50	0.625 ± 0.089	0.985 ± 0.004
100	0.648 ± 0.090	0.986 ± 0.007
1,000	$\mathbf{0.698} \pm 0.029$	0.984 ± 0.003
10,000	0.665 ± 0.084	0.988 ± 0.003

Table 16: Disruptiveness threshold ρ.

ρ	Faithfulness	Completeness
1.00	0.625 ± 0.089	0.985 ± 0.004
0.75	0.661 ± 0.027	0.989 ± 0.003
0.50	$\mathbf{0.684} \pm 0.024$	0.987 ± 0.006
0.25	0.659 ± 0.053	0.989 ± 0.003
0.00	0.523 ± 0.050	0.984 ± 0.007

I Additional Analyses

This section reports additional analyses of SEED. Unless stated otherwise, all circuits are frozen after discovery and evaluated on held-out examples under the benchmark reference corruption with the Faithfulness and Completeness metrics of Section 5.1.

I.1 SECOND TARGET MODEL: QWEN3-0.6B

We run SEED (D) and SEED-EDGE (D) on Qwen3-0.6B on four tasks: IOI, Greater-Than, Hypernymy, and Docstring. This experiment also changes the proposal prior: instead of `roberta-base`, candidates are proposed by the diffusion language model `dllm-hub/Qwen3-0.6B-diffusion-mdlm-v0.1`. Each condition uses 1,000 examples (800 train / 200 held out), three seeds, 1,500 optimization steps, and an exact budget of 1% of the valid edges ($K = 3{,}172$ of 317,171). Within the model, both methods share the graph, budget, data, task score, patching protocol, and evaluation data; the prior is used only during discovery. Table 17 shows that SEED improves average Faithfulness over SEED-EDGE on these four tasks (0.822 vs. 0.769), with non-negative per-task differences and comparable Completeness.

Table 17: Qwen3-0.6B with a diffusion-LM proposal prior. Mean $\pm$ standard deviation over three seeds on held-out examples.

	Faithfulness		Completeness	
Task	SEED (D)	SEED-EDGE (D)	SEED (D)	SEED-EDGE (D)
IOI	**0.651** $\pm$ 0.023	0.596 $\pm$ 0.118	0.980 $\pm$ 0.007	**0.983** $\pm$ 0.014
Greater-Than	**0.966** $\pm$ 0.007	0.819 $\pm$ 0.038	**0.997** $\pm$ 0.000	0.986 $\pm$ 0.007
Hypernymy	0.937 $\pm$ 0.003	0.937 $\pm$ 0.014	0.998 $\pm$ 0.001	0.998 $\pm$ 0.001
Docstring	**0.735** $\pm$ 0.057	0.726 $\pm$ 0.090	**0.983** $\pm$ 0.004	0.978 $\pm$ 0.003
Average	**0.822**	0.769	**0.989**	0.986

I.2 COMPUTATIONAL COST

Table 18 reports wall time and peak GPU memory for IOI discovery with `gpt2-small`, $K = 300$, and 800 training examples on one NVIDIA RTX 5090. SEED and SEED-EDGE use batch size 4, EAP-IG uses five integration steps, and Edge Pruning uses 3,000 optimization steps. Structure Edit increases wall time over SEED-EDGE by $1.46\times$; SEED is slower than EAP, comparable to five-step EAP-IG, and cheaper than Edge Pruning in both wall time and peak memory.

Table 18: Discovery cost on IOI (`gpt2-small`, $K = 300$, 800 examples, one RTX 5090).

Method	Wall time (s)	Peak memory (MiB)
EAP	80.90	2,679
EAP-IG	324.85	2,385
Edge Pruning	1,330.25	13,284
SEED-EDGE	223.71	6,157
SEED	327.32	4,293

I.3 STRUCTURAL OVERLAP BETWEEN SEED AND SEED-EDGE

To test whether the differences between SEED and SEED-EDGE are confined to a few unstable edges, we compare matched circuits across all eight tasks and budgets. For each task and K, we select the highest-Faithfulness circuit among the three seeds for each method and compute $\mathrm{Overlap@}K = |E_{\text{SEED}} \cap E_{\text{SEED-EDGE}}|/K$. Table 19 reports the results. Across the 64 task–budget pairs, the mean overlap is 48.5% (range 16.7%–93.3%) and the mean Jaccard similarity is 34.2%. The two methods therefore share a partial backbone while replacing many edges. Overlap is lowest on IOI, where learning the corruption helps most, and highest on constrained tasks such as Capital-Country and Hypernymy. Structural overlap alone does not establish that the replaced edges implement distinct mechanisms.

I.4 EXAMPLE OF STRUCTURE EDIT SELECTION

To illustrate what Structure Edit optimizes, consider an IOI example whose benchmark corruption changes the second occurrence of the name `Alicia` to `Thomas`; SEED edits only this position.

Table 19: Edge overlap between the best-Faithfulness SEED (D) and SEED-EDGE (D) circuits (% of K).

Task	$K=30$	50	100	300	500	1,000	2,000	10,000
IOI	16.7	22.0	39.0	29.3	29.8	25.2	26.8	43.9
Capital-Country	90.0	80.0	77.0	70.7	64.6	51.4	46.9	51.2
SVA	16.7	22.0	19.0	51.3	46.4	41.1	43.6	54.2
Greater-Than	90.0	88.0	75.0	58.0	54.0	44.4	39.3	55.9
Negation	53.3	42.0	37.0	38.7	38.2	34.4	28.1	38.3
Gender Bias	40.0	36.0	41.0	44.3	40.6	39.2	35.4	55.9
Hypernymy	93.3	74.0	76.0	69.7	66.6	59.4	53.0	52.6
Docstring	40.0	42.0	50.0	48.0	47.2	38.8	41.0	47.3

The frozen MLM prior strongly favors `She` (probability 0.5999) over `He` (0.0183). At step 600, the posterior is still spread over several plausible candidates, with `She` as the mode (0.4064). By step 1,000, the posterior concentrates on `He`, despite its roughly $33\times$ lower prior probability. Structure Edit therefore does not copy the language-model mode; it reorders admissible candidates by how clearly patching through the current circuit restores clean behavior. The selected edits are used only during discovery and never enter the reported metrics.

J QUALITATIVE CIRCUIT VISUALIZATIONS

This section provides qualitative node-link visualizations for representative decoded circuits from the main benchmark. We fix the edge budget to $K = 100$ and show one benchmark seed (seed $= 42$) for each task. The visualizations are meant as qualitative examples rather than aggregate evidence: the quantitative conclusions are based on the five-seed, multi-budget results in Table 1 and Appendix G.2.

Color coding. Edge colors encode the destination read slot: query (q), key (k), value (v), MLP/residual read, and final logit read. Within each circuit, edge width is proportional to the absolute edge logit. Node fill colors provide a lightweight layer cue, with special colors for the embedding and final residual/logit nodes.

Algorithm 1 SEED: Structure Edit and Edge Discovery

Require: Training pairs $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i})\}_{i=1}^{n}$; precomputed candidate priors $\{p_{\text{MLM}}(\cdot \mid \boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i})\}_{i=1}^{n}$; edge budget K; steps T; batch size B; Structure Edit temperature η; warmup steps T_{warm}.

Initialize shared edge logits $\boldsymbol{\omega}$ and optimizer.
for $t = 0, \ldots, T-1$ **do**
  Sample minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$.
  Structure Edit: update the corruption coordinate
  for each $i \in \mathcal{B}$ **do**
    Retrieve the precomputed candidate prior

$$p_i(\cdot) \leftarrow p_{\text{MLM}}(\cdot \mid \boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i}).$$

    Set the reference-anchored finite candidate family

$$\mathcal{S}_i \leftarrow \text{supp}(p_i).$$

    for each $\boldsymbol{x}_{\text{cor}} \in \mathcal{S}_i$ **do**
      Compute the soft advantage and gap:

$$\widetilde{A}_i(\boldsymbol{x}_{\text{cor}}) \leftarrow \widetilde{A}(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{cor}}; \boldsymbol{\omega}), \quad d_i(\boldsymbol{x}_{\text{cor}}) \leftarrow \Delta(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{cor}}).$$

    end for
    Set the reference gap

$$d_i^{\text{ref}} \leftarrow \Delta(\boldsymbol{x}_{\text{clean},i}, \boldsymbol{x}_{\text{ref},i}).$$

  end for
  for each $i \in \mathcal{B}$ **do**
    Define the tilted distribution for a per-pair candidate multiplier $\lambda \geq 0$:

$$q_{\mathcal{B},i}^{\lambda}(\boldsymbol{x}_{\text{cor}}) \leftarrow \text{Normalize}_{\mathcal{S}_i}\left[p_i(\boldsymbol{x}_{\text{cor}}) \exp\left(\frac{\widetilde{A}_i(\boldsymbol{x}_{\text{cor}}) + \lambda d_i(\boldsymbol{x}_{\text{cor}})}{\eta}\right)\right].$$

  end for
  For each $i \in \mathcal{B}$, find the smallest $\widehat{\lambda}_i(\boldsymbol{\omega}) \geq 0$ by bisection such that

$$\sum_{\boldsymbol{x}_{\text{cor}} \in \mathcal{S}_i} q_{\mathcal{B},i}^{\widehat{\lambda}_i(\boldsymbol{\omega})}(\boldsymbol{x}_{\text{cor}}) d_i(\boldsymbol{x}_{\text{cor}}) \geq d_i^{\text{ref}}.$$

  If the constraint already holds at $\lambda = 0$, set $\widehat{\lambda}_i(\boldsymbol{\omega}) \leftarrow 0$.
  for each $i \in \mathcal{B}$ **do**
    Set $q_{\mathcal{B},i}^{\star} \leftarrow q_{\mathcal{B},i}^{\widehat{\lambda}_i(\boldsymbol{\omega})}$.
    Decode the training corruption

$$\widehat{\boldsymbol{x}}_{\text{cor},i} \leftarrow \arg \max_{\boldsymbol{x}_{\text{cor}} \in \mathcal{S}_i:\, d_i(\boldsymbol{x}_{\text{cor}}) \geq d_i^{\text{ref}}} q_{\mathcal{B},i}^{\star}(\boldsymbol{x}_{\text{cor}}).$$

  end for
  Edge Discovery: update the circuit coordinate
  if $t < T_{\text{warm}}$ **then**
    Use warmup sigmoid gates and add the soft budget penalty.
  else
    Use the hard-forward / soft-backward top-K gate induced by $\boldsymbol{\omega}$.
  end if
  Compute the minibatch loss $\mathcal{L}_o(\boldsymbol{\omega})$ (denoising, Eqn. (27), or noising, Eqn. (28))
  Update $\boldsymbol{\omega}$ by one optimizer step minimizing $\mathcal{L}_o$ plus the warmup penalty if applicable.
end for
return Final hard circuit $c_K(\boldsymbol{\omega}) = \{e \in E : \text{TopKMask}_K(\boldsymbol{\omega})_e = 1\}$..



Figure 18: Circuit visualizations at $K = 100$ and $\text{seed} = 42$ across all eight tasks. Each cell shows the decoded hard top-K SEED circuit for one task, with no in-panel metric annotation.