CAN CIRCUIT DISCOVERY BE LEARNED? A GRAPH NEURAL NETWORK APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Circuit discovery aims to find the sparse subnetworks responsible for specific behaviors in language models. Existing automated methods either approximate the effect of interventions to first order, as in attribution patching, or optimize a new set of mask parameters for every task, as in Edge Pruning. In both cases, nothing reusable is learned about how to find circuits. We take a different approach and train a graph neural network to predict circuits directly from activations, extending recent work that trains models to interpret model internals. The network operates on the transformer's computation graph, takes clean and counterfactual activations as input, and assigns a score to each edge. Keeping the highest-scoring edges gives a circuit at any target sparsity in a single forward pass, without attribution-derived features or test-time optimization. Scorers trained on individual tasks outperform the previous state-of-the-art methods EAP and EAP-IG on integrated circuit performance (CPR) on the MIB Benchmark. We then train a scorer jointly across multiple tasks and this scorer successfully infers the task and its circuit from the activations alone, remaining competitive against task specific GNN. We see this as the first step toward general-purpose learned circuit discovery.

1 INTRODUCTION

Recent progress in the development of language models has brought attention to interpretability research that aims to demystify the internal workings of a model from seemingly random numbers to meaningful human-understandable structures. Many approaches have been explored, from parameter decomposition (Geiger et al., 2021) to causal variable localization (Bricken et al., 2023; Cunningham et al., 2024). Circuit discovery, identifying subnetworks within the model which are responsible for solving specific tasks, has been shown to be one such promising approach (Olah et al., 2020; Elhage et al., 2021; Wang et al., 2023), though identifying these circuits spans a range of automation, from manual causal interventions (Wang et al., 2023; Hanna et al., 2023) to fully automated circuit discovery algorithms (Conmy et al., 2023; Syed et al., 2024; Bhaskar et al., 2024). However, the automated methods largely rely on heuristics, first-order approximations, and task-dependent learning to understand complex circuits.

In this work, we formulate the task of discovering a circuit as a learning problem and propose to use graph neural networks (GNNs) to recover high-quality circuits from the activations of the residual streams of a language model. The input to the GNN is the clean and counterfactual activations at every node of the model's computation graph, and its output is a mask over all the edges of the graph, each of which is a potential component of the circuit; keeping the highest-scoring edges gives a circuit at any target sparsity in a single forward pass. We use the faithfulness metric defined in the literature (Hanna et al., 2024; Mueller et al., 2025) as the loss for optimizing the GNN, through a differentiable relaxation of the mask. On the MIB circuit-localization suite, spanning GPT-2, Qwen2.5-0.5B, Gemma-2-2B, and Llama-3.1-8B, our method finds more faithful circuits than EAP and EAP-IG on every model–task pair, with 1.9 to 6.2 times the integrated circuit performance of EAP-IG and the largest gains at the sparsest budgets, and the circuits it finds are almost fully connected. We further show that a single GNN trained jointly on seven tasks lands between 28% below and 25% above task-specific GNNs and above EAP-IG on all seven, with under a third of their combined training.

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.

These results suggest that circuit discovery need not be specific for every task and model: the knowledge of how to find a circuit can live in reusable weights, and a scorer can infer the task and its circuit from activations alone. This points toward an amortized, general-purpose circuit discoverer that returns a circuit for an arbitrary prompt in a single forward pass.

2 BACKGROUND

2.1 CIRCUIT DISCOVERY ON LANGUAGE MODELS

We focus our attention on discovering circuits for transformers, the paradigmatic architecture for language models. While neuron-level circuit discovery is possible, the sheer scale and organized structure of Large Language Models have motivated circuit discovery at higher abstraction levels. We adopt the transformer circuit formulation developed by Elhage et al. (2021) where components read and write information through a shared residual stream.

Let M be a frozen transformer and $G = (V, E)$ its directed computation graph at the level of attention heads and MLP blocks (Elhage et al., 2021; Wang et al., 2023). The nodes are the token embeddings, attention heads, MLPs, and output logits, and an edge (u, v) says that component v reads the output of component u from the residual stream. Given a dataset for a task, the goal is to pick out a sparse set of edges, called a *circuit*, such that the model still performs the task when everything outside the circuit is ablated. A candidate circuit would be a binary edge mask $m \in \{0, 1\}^{|E|}$ where $m_e = 1$ keeps the edge e and $m_e = 0$ ablates it.

Evaluating a candidate circuit requires specifying how contributions from edges outside the circuit are replaced, of which there are many options and is still a topic of debate. *Zero ablation* sets these contributions to zero, which may disrupt downstream computations that may expect activation values of a certain range(Wang et al., 2023). *Mean ablation* replaces each contribution with its average over a chosen dataset. *Counterfactual ablation* replaces each contribution with the corresponding activation from another input of the same structure and task while changing task-relevant information to elicit a different answer. These choices can substantially affect which circuits are found, making the ablation protocol an essential part of specifying and evaluating a circuit (Zhang & Nanda, 2024; Miller et al., 2024).

We use counterfactual ablation: each clean prompt x is paired with a counterfactual prompt x' that changes the task-relevant information, and ablating an edge swaps the clean information it carries for counterfactual information. Let $a_u(x)$ and $a_u(x')$ denote the activations of node u when the model runs on the clean and the counterfactual prompt, respectively. In the intervened forward pass on the clean prompt, node v receives from node u, along the edge $e = (u, v)$, the contribution

$$\tilde{a}_e \;=\; m_e\, a_u(x) \;+\; (1 - m_e)\, a_u(x'), \tag{1}$$

where $a_u(x')$ is cached from an unmodified forward pass on x', while $a_u(x)$ is the activation of u within the intervened pass itself. Each node sums its incoming contributions, $z_v = \sum_{u:(u,v)\in E} \tilde{a}_{(u,v)}$, and applies its usual computation. If edges into u are ablated, its output changes, and every kept edge out of u carries that changed output, so ablations compound through the network instead of each edge being patched in isolation. We denote the intervened model M_m and we use one mask across all token positions so each task gets a single circuit.

2.2 AUTOMATED CIRCUIT DISCOVERY

Recently, there's been a new line of work of automating the process of circuit discovery. Attribution-based methods approximate the effect of replacing clean activations with counterfactual activations. Edge Attribution Patching (EAP; Syed et al., 2024) estimates the change in a task metric F from ablating edge $e = (u, v)$ as $\Delta F_e \approx \langle a_u(x') - a_u(x),\, \partial F/\partial z_v \rangle$, where a_u is the source activation and z_v is the destination's input, with the gradient evaluated on the clean forward pass. EAP-IG (Hanna et al., 2024) averages gradients along an interpolation between counterfactual and clean input embeddings to improve this approximation.

Other approaches use search or learning to identify circuits. Automatic Circuit DisCovery (ACDC; Conmy et al., 2023) directly tests edge interventions, greedily removing edges when the resulting change in model behavior falls below a threshold. Learning-based methods instead optimize masks

or train a model to make circuit-selection decisions. The ACDC paper adapts Subnetwork Probing (Cao et al., 2021), which learns masks over model components using gradient descent and a sparsity penalty. Davies et al. (2023) likewise learn a continuous mask over attention heads and MLPs, but specify the target computation through *desiderata*: sets of original, alternate, and target outputs that state how patching the sought-after components should change the model's answer, which lets them isolate a subtask shared across tasks, such as variable binding, rather than a full input–output circuit. Edge Pruning (Bhaskar et al., 2024) learns stochastic masks over connections between components through a continuous relaxation, minimizing the KL divergence between the full model and the intervened model while enforcing a target sparsity. MechRL (Khadka, 2026) trains a reinforcement-learning policy to sequentially remove edges, using rewards based on circuit faithfulness and size, and reports transfer from training behaviors to held-out behaviors. In concurrent work, Graph Circuit Learning (Tan et al., 2026) trains a GNN across model–task pairs to predict which edges belong to a circuit; it is supervised with ground-truth masks, which exist only for semi-synthetic Tracr and InterpBench transformers, and is evaluated by edge AUROC rather than by faithfulness.

Each of these methods has its own flaws and inductive biases. ACDC (Conmy et al., 2023) makes locally greedy pruning decisions, which need not produce the best circuit overall. Works like Cao et al. (2021); Davies et al. (2023); Bhaskar et al. (2024) learn separate mask parameters for each task, rather than a shared, activation-dependent rule for assigning edge importance.

EAP and EAP-IG score edges individually, whereas circuit performance depends on their joint intervention. A Taylor expansion around the full model gives

$$F(M_m(x)) = F(M(x)) - \underbrace{\sum_{m_e=0} \frac{\partial F}{\partial m_e}}_{\text{edge attributions}} + \underbrace{\frac{1}{2} \sum_{m_e=m_f=0} \frac{\partial^2 F}{\partial m_e \, \partial m_f}}_{\text{curvature and interactions}} + \cdots,$$

with m treated as continuous through Equation 1 and derivatives evaluated at $m = \mathbf{1}$. EAP omits higher-order terms, which can be important when selecting combinations of edges. EAP-IG accounts for nonlinear effects through path-averaged gradients but does not explicitly estimate interactions between edges.

2.3 Evaluating Circuits

We evaluate a circuit by how much task performance it recovers under the intervention protocol above; the choice of performance metric matters (Edge Pruning, for instance, uses the KL divergence between the full model's and the circuit's output distributions; Bhaskar et al., 2024). Following Mueller et al. (2025), we evaluate the task logit difference and normalize it relative to the full model and the empty circuit. Let $S(M_m)$ denote the mean task logit difference of the intervened model on the evaluation set. Circuit faithfulness is defined as

$$f(m) = \frac{S(M_m) - S(M_{\mathbf{0}})}{S(M) - S(M_{\mathbf{0}})}, \tag{2}$$

where $M_{\mathbf{0}}$ is the empty circuit, with all edge contributions replaced by their counterfactual activations. Higher faithfulness indicates a larger average logit margin in favor of the target answer over the contrast answer.

To compare automated discovery methods, we evaluate their circuits across a range of retained-edge fractions. The integrated circuit performance ratio (CPR; Mueller et al., 2025) summarizes the area under the faithfulness curve, $\mathsf{CPR} = \int_0^1 f(m^{(\rho)})\, d\rho$, where $m^{(\rho)}$ denotes the circuit mask produced by a discovery method at retained-edge fraction ρ.

For this paper, we place greater emphasis on performance at small retained-edge fractions, where circuits are most sparse. We therefore use a logarithmically weighted variant of CPR:

$$\mathsf{CPR}_{\log} = \frac{1}{\log(1/\rho_{\min})} \int_{\rho_{\min}}^1 f(m^{(\rho)})\, \frac{d\rho}{\rho}, \tag{3}$$

where $\rho_{\min} > 0$ is the smallest evaluated retained-edge fraction. This weighting assigns equal importance to equal multiplicative ranges of circuit size, such as 0.1–1% and 1–10% of the edges. CPR rewards circuits whose logit difference is large, including larger than the full model's. MIB's

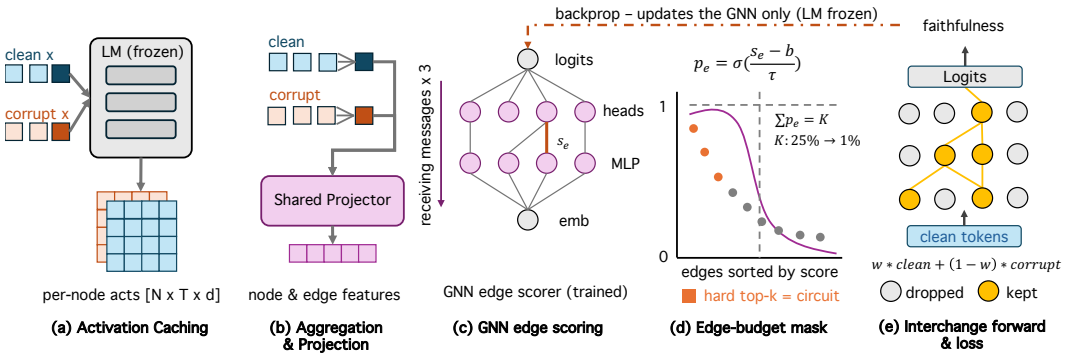


Figure 1: **Method overview.** (a) We run the frozen language model on paired clean and counterfactual (corrupt) prompts and cache the activations at every node. (b) A shared projector pools the activations of the original prompt and the counterfactual prompt into node and edge features. (c) A GNN propagates information across neighboring nodes and assigns a score to each edge. (d) The scores are turned into soft gates p_e that sum to an edge budget K; at evaluation, the top-K edges are the circuit. (e) During training, each edge mixes clean and counterfactual activations by its gate, and a faithfulness loss on the resulting logits trains the GNN while the language model stays frozen.

second metric, the circuit–model distance $\mathsf{CMD} = \int_0^1 |1 - f(m^{(\rho)})|\, d\rho$, instead penalizes deviations from the full model in either direction and is zero when the circuit reproduces it at every budget; we report its log-weighted analogue $\mathsf{CMD}_{\log}$, defined in Appendix D.

2.4 Interpreting Models with Models

A growing line of work hands interpretability analyses to a second model: prompted or fine-tuned language models that describe units or decode activations (Hernandez et al., 2022; Bills et al., 2023; Paulo et al., 2025; Choi et al., 2024; Ghandeharioun et al., 2024; Chen et al., 2024; Pan et al., 2026; Karvonen et al., 2025; Costarelli et al., 2024; Huang et al., 2025; Fraser-Taliente et al., 2026), whose text output can only be verified by judging text (Bauer et al., 2026), and meta-models that read another network's weights to recover its program, trojans, or properties (Langosco et al., 2024; Kolouri et al., 2020; Xu et al., 2021; Schürholt et al., 2021), which need populations of trained networks and have been limited to small models. Closest to our setup, HyperDAS trains a hypernetwork that amortizes distributed alignment search (Geiger et al., 2024; Sun et al., 2025), locating the positions and subspace that represent a concept from counterfactual data. Our scorer is likewise trained on activations, but it outputs a circuit, which the protocol of Section 2.3 can check directly, and it learns from the model's behavior under intervention rather than from labels; unlike HyperDAS, it recovers edges across the whole computation graph. The other learned circuit discoverers, MechRL and Graph Circuit Learning, are discussed in Section 2.2.

3 Graph Neural Network Based Circuit Discovery

We work in the setting of Section 2.1: given a frozen transformer M with computation graph $G = (V, E)$ and paired clean and counterfactual prompts x and x' (the *corrupt* input in Figure 1), we want a sparse edge mask m such that the intervened model M_m of Equation 1 still performs the task while every edge outside the circuit carries counterfactual activations. Rather than approximating each edge's effect to first order (Syed et al., 2024; Hanna et al., 2024) or optimizing fresh mask parameters for every task (Cao et al., 2021; Bhaskar et al., 2024), we train a graph neural network (GNN) to predict circuits from activations; a circuit is naturally a graph in which an edge's importance depends on the components around it. Figure 1 shows the pipeline: cached clean and counterfactual activations are pooled into node and edge features, message passing assigns each edge a score, the scores are softened into gates that sum to an edge budget, and a faithfulness loss on the gated model's output is backpropagated through the frozen transformer into the GNN. At test time the scorer ranks all edges in one forward pass, with no attribution-derived features and no optimization. The rest of this section walks through each stage.

Node and edge features from activations. The scorer sees each node and edge through a feature vector built from the cached activations (Figure 1b). Let $A_v(x)$ be the sequence of activation vectors of node v across token positions. A shared trainable projector P_θ applies layer normalization, a linear down-projection, and a GELU to each token's activation, and a pooling function Q_θ then collapses the projected sequence into one vector. We run this map over the clean activations and over their difference from the counterfactual activations, and concatenate the two summaries into an activation descriptor r_v:

$$r_v \;=\; \big[\, Q_\theta\big(P_\theta(A_v(x))\big) \,\big\|\, Q_\theta\big(P_\theta(A_v(x) - A_v(x'))\big) \,\big]. \tag{4}$$

The difference term tells the scorer where the clean and counterfactual runs disagree.

A node's feature vector ϕ_v is its descriptor r_v plus a small static encoding ϕ_v^{static} of its place in the graph (normalized layer and head indices and the component type). The feature vector ϕ_e of an edge $e = (u, v)$ combines an analogous static encoding ϕ_e^{static} with the descriptors r_u and r_v of its endpoints and their elementwise product $r_u \odot r_v$:

$$\phi_v = \big[\phi_v^{\text{static}} \,\|\, r_v\big], \qquad \phi_e = \big[\phi_e^{\text{static}} \,\|\, r_u \,\|\, r_v \,\|\, r_u \odot r_v\big] \quad \text{for } e = (u, v). \tag{5}$$

Note that we do not use any attribution-derived features, such as gradients of a task metric.

Edge-scoring GNN. The scorer keeps hidden states $h_v^{(t)}$ per node and $g_e^{(t)}$ per edge, with t the message-passing round. Learned encoders f_{node} and f_{edge} initialize them from the features:

$$h_v^{(0)} = f_{\text{node}}(\phi_v), \qquad g_e^{(0)} = f_{\text{edge}}(\phi_e). \tag{6}$$

Message passing then lets an edge's score depend on the activity of the components around its endpoints (Figure 1c).

Messages flow against the direction of the computation graph. For an edge $e = (u, v)$, the GNN sends a message from the consumer v back to the producer u, which lets an upstream component find out about the downstream components that read its output; the transformer computation and the interventions keep their original direction. At round t, the message $\mu_{v\to u}^{(t)}$ and its degree-normalized aggregate $\bar{\mu}_u^{(t)}$ are

$$\mu_{v\to u}^{(t)} = F_t\big(h_v^{(t)},\, h_u^{(t)},\, g_{(u,v)}^{(t)}\big), \qquad \bar{\mu}_u^{(t)} = \frac{1}{\sqrt{\max\big(1,\, |\mathcal{N}^+(u)|\big)}} \sum_{v \in \mathcal{N}^+(u)} \mu_{v\to u}^{(t)}, \tag{7}$$

where F_t is a gated MLP and $\mathcal{N}^+(u) = \{v : (u, v) \in E\}$ are the downstream neighbors of u. A residual update injects $\bar{\mu}_u^{(t)}$ into the node state, and each edge state is then updated from its previous value and its endpoints' new states. Like stacked convolutions, one round moves information one hop, so we repeat the update T times to let a wider neighborhood influence each score.

After the last round, a shared readout R_θ turns each edge, taken in its original orientation, into a scalar score s_e:

$$s_e = R_\theta\big(h_u^{(T)},\, h_v^{(T)},\, g_e^{(T)},\, \bar{h}^{(T)},\, \bar{g}^{(T)}\big), \qquad e = (u, v), \tag{8}$$

where the mean node and edge states $\bar{h}^{(T)}$ and $\bar{g}^{(T)}$ give the readout some global context. Larger scores indicate edges of higher priority to include in the circuit.

Learning edge rankings under interventions. We train the scorer so that keeping its highest-scoring edges recovers the task behavior of interest. Ideally, training would use the same binary top-K masks used at evaluation, but hard edge selection is not differentiable, and our attempts to train with straight-through estimators were unsuccessful. During training we therefore relax the binary mask of Equation 1 to soft gates.

We first standardize the scores to zero mean and unit variance, with a numerical floor on the scale, and write $\hat{s}$ for the result. The gate p_e of edge e is a sigmoid σ of the standardized score, with temperature τ,

$$p_e = \sigma\left(\frac{\hat{s}_e - b}{\tau}\right), \qquad \sum_{e \in E} p_e = K, \tag{9}$$

where at every training step we solve numerically for the bias b that makes the gates sum to the edge budget K (Figure 1d). K is varied over a range of sparsities during training, so the same scorer can be asked for circuits at any budget.

The gates replace the binary mask in Equation 1; along edge $e = (u, v)$ the contribution becomes

$$\tilde{a}_e = p_e \, a_u(x) + (1 - p_e) \, a_u(x'), \tag{10}$$

so a gate near one mostly keeps the clean contribution and a gate near zero mostly swaps in the counterfactual one. This is the interchange forward pass drawn in Figure 1e, where the kept edges carry clean activations and the dropped edges carry counterfactual ones.

Let M_p be the soft-gated model and $\Delta(M_p, x)$ its task logit difference on prompt x. B is the batch size, and let x_i and p_i be the i-th prompt in the batch and its gates. Our Max-LD objective pushes this quantity up,

$$\mathcal{L}_{\text{Max-LD}} = -\frac{1}{B} \sum_{i=1}^{B} \Delta(M_{p_i}, x_i), \tag{11}$$

and rewards larger logit differences, including ones that exceed the clean model's. One mismatch remains between training and evaluation. Soft gates can spread their mass over more edges than the binary circuit will keep, so we add a penalty on gate mass that falls outside the top-K edges:

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \frac{\lambda}{B|E|} \sum_{i=1}^{B} \sum_{e \in E} p_{i,e} \big(1 - m_{i,e}^{\text{hard}}\big), \tag{12}$$

where λ weights the penalty, $\mathcal{L}_{\text{task}}$ is either of the objectives in Equations 11 and 13, and m_i^{hard} is the binary top-K mask, treated as constant during differentiation. Gradients flow through the intervened transformer, the scorer, and then feature projector, while the transformer itself stays frozen (Figure 1e). At evaluation, we only take into account the ranking and keep the top-K highest-scoring edges (Figure 1d).

4 Experiments

In this section we evaluate the GNN scorer of the previous section against attribution-based and mask-learning baselines, under the counterfactual ablation protocol of Section 2.1. We ask three questions. (Q1) Does a scorer trained under interventions find more faithful circuits than attribution and mask-learning methods (Section 4.2)? (Q2) What do the circuits it finds look like, and why does scoring edges jointly help (Section 4.3)? (Q3) Can one scorer, given no task label, discover circuits for several tasks at once (Section 4.4)?

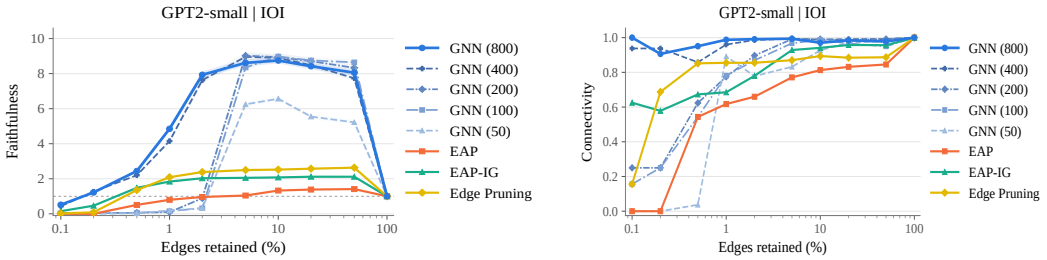
4.1 Setup

Models and tasks. For the task-specific experiments (Sections 4.2 and 4.3) we follow the MIB circuit-localization suite (Mueller et al., 2025): IOI on GPT-2, Qwen2.5-0.5B, Gemma-2-2B, and Llama-3.1-8B; MCQA on Qwen, Gemma, and Llama; ARC-Easy on Gemma and Llama; and ARC-Challenge, arithmetic addition, and arithmetic subtraction on Llama. The harder tasks are run only on the more capable models. For the joint-training experiments (Section 4.4) we use GPT-2 alone, and since it solves only IOI among the MIB tasks, we instead use seven synthetic task families that GPT-2 solves reliably: induction, IOI, subject–verb agreement, pronoun prediction, numerical comparison, lookup, and initials, each with 4,096 training, 256 validation, and 512 test examples (Appendix A). Every example is paired with counterfactual prompt that changes the task-relevant information that produces a different answer and all methods use the counterfactual ablation of Section 2.1.

Baselines. Of the methods in Section 2.2, we compare against EAP, EAP-IG (input-interpolation variant), and Edge Pruning, adapted to learn masks over edges only so that its search space matches ours (Appendix B); all use the same counterfactual pairs as our scorer. Edge Pruning is reported for GPT-2 only: the method operates on a gate-instrumented reimplementation of the target architecture, and the official release provides these for GPT-2, Llama, and Tracr models; no Gemma-2 or Qwen2.5

Table 1: Edge-level $\mathsf{CPR}_{\log}$ (↑) on the MIB public test sets: normalized log-weighted CPR (Section 2.3) over retained-edge fractions from 0.1% to 100%. Q: Qwen2.5-0.5B; G: Gemma-2-2B; L: Llama-3.1-8B. Edge Pruning is reported for GPT-2 only (–: not reported).

	IOI				MCQA			ARC-Easy		ARC-Challenge	Addition	Subtraction
Method	GPT-2	Q	G	L	Q	G	L	G	L	L	L	L
EAP	0.93	1.13	0.87	0.66	0.61	0.87	0.90	0.99	1.05	1.09	0.40	0.47
EAP-IG	1.66	1.61	2.72	2.22	0.79	1.11	1.15	1.24	1.15	1.28	1.11	1.09
Edge Pruning	1.86	–	–	–	–	–	–	–	–	–	–	–
GNN (Max-LD)	5.71	3.00	5.32	5.82	4.25	4.34	7.13	4.37	6.97	7.95	2.13	2.33



(a) Faithfulness against retained edges.

(b) Connectivity against retained edges.

Figure 2: **IOI on GPT-2.** (a) Faithfulness of the top-K circuit as the retained-edge fraction varies, for the Max-LD scorer after 50, 100, 200, 400, and 800 updates and for EAP and EAP-IG. (b) Fraction of the selected edges that lie on a complete path from the token embeddings to the logits, for the same scorer checkpoints and baselines.

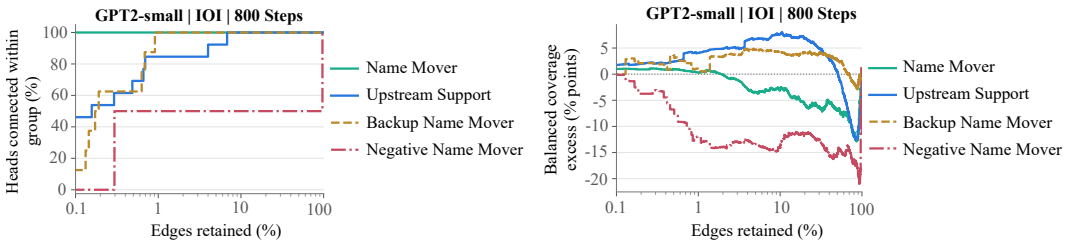
variant is publicly available. We do not run ACDC, which is iterative, spending one forward pass per candidate edge per threshold; the ACDC paper itself stops at GPT-2-scale models, and with the far larger edge counts of the Gemma-2, Qwen2.5, and Llama-3.1 graphs the search is prohibitive. MIB does not include it either. We also omit Graph Circuit Learning, which requires ground-truth edge labels that do not exist for these models, and MechRL, which never searches the full graph.

Circuit extraction and evaluation. The scorer produces example-dependent edge scores; after training we average them over the training split into one task-level ranking, so each task has a single circuit, and keep the top-K edges at each budget. We then apply connectivity pruning, which removes selected edges not on a complete path from the token embeddings to the logits; *connectivity* is the fraction of selected edges that survive. Faithfulness is always computed on the pruned circuit, with the budget ρ referring to the fraction of edges selected before pruning. Circuits are evaluated on the held-out test split under the intervention of Equation 1, with no optimization at test time; we report faithfulness $f(m)$ as a function of $\rho \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}\%$, summarized by $\mathsf{CPR}_{\log}$ in Table 1 and $\mathsf{CMD}_{\log}$ in Table 2, both as defined in Section 2.3.

Implementation. The scorer uses three rounds of message passing with hidden width 128 and is trained for 800 updates with Adam, cycling through edge budgets from 0.1% to 50% so that one scorer serves every sparsity (Appendix B). Appendix E ablates the input features and the message-passing direction on GPT-2: clean activations plus their difference from the counterfactual ones, propagated against the direction of computation, perform best, and we use them throughout.

4.2 Task-Specific Scorers

We first train a separate scorer for each model–task pair, i.e., we train a GNN from scratch on the training samples of each task and evaluate it on the held-out samples. Table 1 reports the CPR and Table 2 the CMD results. **Learned scorers find more faithful circuits than attribution methods on every model–task pair.** The Max-LD scorer's CPR is between 1.9× (IOI on Qwen and on Gemma) and 6.2× (MCQA on Llama) that of EAP-IG, the stronger of the two attribution baselines, and the margin over EAP is larger still.



(a) Heads on complete input-to-logits paths.

(b) Wiring selection relative to same-layer heads.

Figure 3: **Coverage of the manual IOI head groups** (Wang et al., 2023) in GPT-2 by the Max-LD scorer after 800 updates (seed 40). (a) Percentage of heads in each group that lie on a complete input-to-logits path; "upstream support" groups the duplicate-token, previous-token, induction, and S-inhibition heads. (b) Mean wiring coverage relative to the other heads in the same layer, in percentage points: for each head, incoming and outgoing edge-retention fractions are weighted equally before subtracting the mean coverage of the other 11 heads in its layer. Both panels vary the percentage of edges selected before path pruning.

Faithfulness above one. CPR values above one mean that the circuit's logit difference exceeds the full model's. This is possible under counterfactual ablation: an edge whose clean contribution lowers the logit difference, such as the output of a negative name-mover head in IOI, raises it when swapped for its counterfactual contribution, and Max-LD (Equation 11) rewards dropping such edges. The same effect lifts EAP-IG above one on most pairs, and Section 4.3 identifies the edges involved.

The gap opens at small budgets. Figure 2a plots faithfulness against the retained-edge fraction ρ for IOI on GPT-2, for the scorer and for EAP and EAP-IG. The scorer is substantially more faithful than both attribution methods at every budget: it reaches a faithfulness of 5.53 at $\rho = 1\%$ and 8.76 at $\rho = 10\%$, against 1.85 and 2.07 for EAP-IG. The difference is already present at the smallest budgets, and the gap widens sharply from around $\rho = 1\%$. The same pattern holds for the other models and tasks; the full set of curves is given in Figure 5 of Appendix C.

4.3 What Circuits Does the Scorer Find?

A high faithfulness score alone does not make a circuit a good explanation, since individually useful edges could score well without forming a coherent mechanism, which brings us to our second question: whether the circuits are qualitatively sensible. We use IOI on GPT-2, the only model–task pair in our suite for which a circuit has been found manually, using path patching (Wang et al., 2023).

The circuits are connected without being asked to be. Figure 2b plots the fraction of selected edges that survive connectivity pruning at each budget. The scorer's circuits are nearly fully connected even at the smallest budgets: 0.94 of its selected edges lie on complete paths at $\rho = 0.1\%$ and 0.98 at $\rho = 1\%$, against 0.63 (20 of 32 edges) and 0.67 (216 of 324) for EAP-IG, and 0.00 and 0.62 for EAP, whose 32 highest-scoring edges at $\rho = 0.1\%$ form no complete path at all; both attribution methods reach the scorer's connectivity only once many more edges are retained. Nothing in the loss encourages this: there is no connectivity term, and the scorer sees only activations. The same pattern holds across models and tasks (Figure 6).

Heads that raise the logit difference enter first; heads that hedge enter last. Although the comparison is not exact, since Wang et al. (2023) identify attention heads and their functional roles whereas our method selects individual edges, their manual IOI circuit is the most thoroughly validated reference for this task, so we compare against it by asking which of its heads lie on the scorer's selected paths. Figure 3 shows, for each head group of the manual circuit, the fraction of its heads that lie on a complete input-to-logits path as the budget grows. The duplicate-token, previous-token, induction, and S-inhibition heads, together with the main and backup name movers, are wired into the circuit at small budgets, whereas the negative name movers are included last: most of their incoming edges stay unselected until the budget is far larger. The negative name movers suppress prediction confidence and are hypothesized by Wang et al. (2023) to help the model hedge against large cross-entropy penalties when it is wrong. Hence it makes sense that they are the last to be included in the circuit.

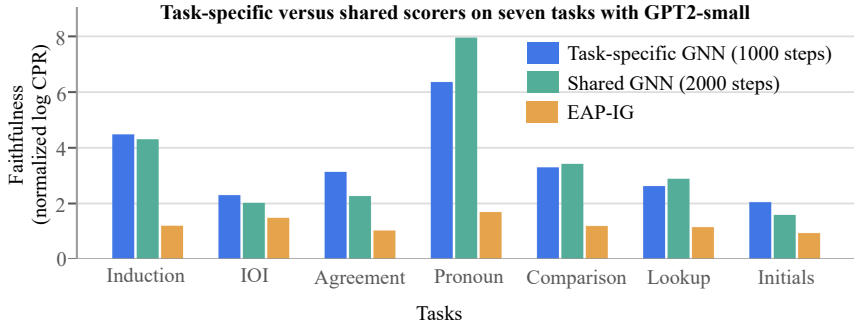


Figure 4: **One scorer for seven tasks on GPT-2.** $\mathsf{CPR}_{\log}$ (Section 2.3) on the test split of each task family: the trapezoidal integral of ratio-of-means faithfulness against $\log$ budget from 0.1% to 100%, divided by $\log 1000$, so that a circuit matching the full model at every budget would score 1. It rewards faithfulness above 1 and is not the linearly integrated MIB CPR. Bars compare a scorer trained on that family alone (1,000 updates), a single scorer trained on all seven families with no task label (2,000 updates), and five-step EAP-IG. All three rank edges from the same 4,096 training pairs and are evaluated on the same 512 test pairs; one training seed, so no error bars.

4.4 One Scorer for All Tasks

We next ask whether a single scorer can discover circuits for several tasks without being told which task it is looking at. We train one shared scorer on the seven task families of Section 4.1 for GPT-2. The shared scorer sees only the clean and counterfactual activations of each example, just as a task-specific scorer does. We compare it with a scorer trained on each family alone and with EAP-IG. All settings follow Section 4.1, except that each task-specific scorer is trained for 1,000 updates and the shared scorer for 2,000.

One scorer for seven tasks. Figure 4 reports $\mathsf{CPR}_{\log}$ on the test split of each family. The shared scorer lands between 28% below and 25% above the task-specific scorer: it exceeds it on pronoun ($+25\%$), lookup ($+10\%$), and comparison ($+4\%$), and trails it on induction (-4%), IOI (-12%), initials (-22%), and agreement (-28%). It outperforms EAP-IG on all seven families, by a factor of 1.4 (IOI) to 4.7 (pronoun). It does so with 2,000 updates against $7 \times 1{,}000 = 7{,}000$ for the specialists, under a third of their combined training. Because it is given no task label, the shared scorer must infer the task and the corresponding circuit from the activations alone. The pronoun gain comes almost entirely from the 1% budget (9.3 versus 2.8); from 2% upward the two scorers are within about 10% of each other, so with one seed we do not read it as a systematic advantage. On agreement and initials the shared scorer's deficit is spread across every budget above 1%.

5 Discussion and Conclusion

We formulate circuit discovery as learning a direct mapping from model activations to the corresponding circuit through a graph neural network. Trained under interventions, the scorer finds circuits that are more faithful than those of prior automated methods across models and tasks, with the largest gains at the sparsest budgets, and the circuits have meaningful structure: they are almost fully connected without any constraint, and on IOI they include the manually identified components in a sensible order. A single scorer trained on seven tasks comes close to seven task-specific scorers at under a third of their training, suggesting circuit discovery could generalize across different tasks.

These results come with some caveats that future work could address. First, the scorer optimizes the quantity it is evaluated on, a logit difference under one ablation protocol, so gains in the objective need not mean better recovery of the model's computation; training across several metrics and ablation schemes would reduce this risk, and which metrics constitute reliable evidence of mechanistic understanding when optimized directly remains open. Second, the shared scorer has not been tested on unseen tasks or beyond GPT-2. Broadening the tasks and models, and building datasets for this setting, are the natural steps toward a fully amortized graph neural network that identifies a meaningful circuit in a single forward pass for any language model and any behavior of interest.

AI USAGE STATEMENT

In this work, we used generative AI tools to polish the text of the manuscript, for example by fixing typos and grammatical errors, to carry out supplementary literature searches, and to assist with the implementation of the research code. All AI-assisted text was reviewed by the authors, and all AI-assisted code was tested and verified by the authors, who take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies the internal mechanisms of publicly released base language models on public benchmark data and on synthetic tasks; it involves no human subjects and no personal data, and it does not modify model weights. Tools that locate the components responsible for a behavior could in principle be used to find and disable safety-relevant mechanisms, but the same capability is what makes auditing models at scale possible, and we believe the latter benefit outweighs the risk for methods, such as ours, that operate on narrow behaviors of open base models. A further concern is epistemic: our scorer optimizes the faithfulness metric it is evaluated on, so high scores should not be read as a complete account of the model's computation; we discuss this limitation in Section 4 and above.

REPRODUCIBILITY STATEMENT

The method is specified in Section 3, with the training objectives and gating given by the equations there and in Appendix D, and the evaluation protocol in Sections 2.3 and 4.1. All models are public checkpoints; the MIB tasks and their public splits are those of Mueller et al. (2025), and the seven synthetic task families are described in Appendix A. Architecture, optimizer, and budget-schedule hyperparameters are listed in Appendix B, the baselines are described in Section 4.1, and the ablations in Appendix E report their seeds and confidence intervals. Code, configuration files, and the synthetic task generator will be released publicly upon publication.

REFERENCES

Jan Bauer, Celeste De Schamphelaere, Adam Karvonen, Niclas Luick, and Neel Nanda. Building better activation oracles. In *Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2606.02609.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2406.16778.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. OpenAI, 2023. URL https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/.

Steven Cao, Victor Sanh, and Alexander Rush. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 960–966. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.74. URL https://aclanthology.org/2021.naacl-main.74/.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. SelfIE: Self-interpretation of large language model embeddings. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2403.10949.

Dami Choi, Vincent Huang, Kevin Meng, Daniel D. Johnson, Jacob Steinhardt, and Sarah Schwettmann. Scaling automatic neuron description. Transluce, 2024. URL https://transluce.org/neuron-descriptions.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2304.14997.

Anthony Costarelli, Mat Allen, and Severin Field. Meta-models: An architecture for decoding LLM behaviors through interpreted embeddings and natural language, 2024. URL https://arxiv.org/abs/2410.02472.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2309.08600.

Xander Davies, Max Nadeau, Nikhil Prakash, Tamar Rott Shaham, and David Bau. Discovering variable binding circuitry with desiderata. In *ICML 2023 Workshop on Challenges in Deployable Generative AI*, 2023. URL https://arxiv.org/abs/2307.03637.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

Kit Fraser-Taliente, Subhash Kantamneni, Euan Ong, Dan Mossing, Christina Lu, Paul C. Bogdan, Emmanuel Ameisen, James Chen, Dzmitry Kishylau, Adam Pearce, Julius Tarng, Alex Wu, Jeff Wu, Yang Zhang, Daniel M. Ziegler, Evan Hubinger, Joshua Batson, Jack Lindsey, Samuel Zimmerman, and Samuel Marks. Natural language autoencoders produce unsupervised explanations of LLM activations. *Transformer Circuits Thread*, 2026. URL https://transformer-circuits.pub/2026/nla/index.html.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.02997.

Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pp. 160–187. PMLR, 2024.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/ghandeharioun24a.html.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2305.00586.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2403.17806.

Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *International Conference on Learning Representations*, 2022.

Vincent Huang, Dami Choi, Daniel D. Johnson, Sarah Schwettmann, and Jacob Steinhardt. Predictive concept decoders: Training scalable end-to-end interpretability assistants, 2025. URL https://arxiv.org/abs/2512.15712.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, and Samuel Marks. Activation oracles: Training and evaluating LLMs as general-purpose activation explainers. *arXiv preprint arXiv:2512.15674*, 2025. URL https://arxiv.org/abs/2512.15674.

Barsat Khadka. MechRL: Reinforcement learning agents perform circuit discovery for mechanistic interpretability, 2026. URL https://arxiv.org/abs/2605.26343v2.

Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in CNNs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Lauro Langosco, Neel Alex, William Baker, David Quarel, Herbie Bradley, and David Krueger. Towards meta-models for automated interpretability, 2024. URL https://openreview.net/forum?id=1zDOkoZAtl.

Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit faithfulness metrics are not robust. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=zSf8PJyQb2.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, et al. MIB: A mechanistic interpretability benchmark. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2504.13151.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

Alexander Pan, Lijie Chen, and Jacob Steinhardt. LatentQA: Teaching LLMs to decode activations into natural language. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7f19b99e63762d20e9df91144056f1ee-Abstract-Conference.html.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2410.13928.

Konstantin Schürholt, Dimche Kostadinov, and Damian Borth. Self-supervised representation learning on neural network weights for model characteristic prediction. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2110.15288.

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. HyperDAS: Towards automating mechanistic interpretability with hypernetworks. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2503.10894.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *BlackboxNLP Workshop at EMNLP*, 2024. URL https://arxiv.org/abs/2310.10348.

Chester Tan, Moritz Lampert, Courtney Maynard, Ankit Ramakrishnan, Tina Eliassi-Rad, and Ingo Scholtes. Can graph learning learn circuits?, 2026. URL https://arxiv.org/abs/2608.08536.

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. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2211.00593.

Xiaojun Xu, Qi Wang, Huichen Li, Nikita Borisov, Carl A. Gunter, and Bo Li. Detecting AI trojans using meta neural analysis. In *IEEE Symposium on Security and Privacy*, 2021.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16042.

A TASKS

For the generalization experiments, we use seven synthetic tasks easy enough for GPT2. Our Python generator samples structured instances while adding variety and structure specific to each task. We retain task conditions for which GPT-2 achieves at least 75% clean answer-pair accuracy on a separate qualification set. More details will soon be available in the accompanying code repository.

Induction. The model recalls the token that previously followed a repeated cue. For example, "Mary John Robert cup key Mary" should continue with "John".

Indirect Object Identification. The model identifies the recipient rather than the repeated giver. For example, "When Mary and John visited the store, John gave a book to" should continue with "Mary". These examples are generated independently of the benchmark IOI dataset.

Subject–verb agreement. The model selects "is" or "are" according to the subject's number despite intervening nouns. For example, "In London, the author near the teachers" should continue with "is".

Pronoun prediction. The model predicts a pronoun associated with a named person after a short context. For example, "The key was nearby. Mary went home after lunch." has the target continuation "She".

Greater-Than. The model selects the larger or smaller of two two-digit integers. For example, "The larger number of 32 and 47 is" should continue with "47".

Lookup. The model retrieves an object assigned to a named container. Given assignments such as "box A: apple" and "box B: book", the query "box A:" should be completed with "apple".

Initials. The model completes the abbreviation of a three-word name using its initial letters. For example, "The Federal Trade Organization (FT" should continue with "O".

B IMPLEMENTATION DETAILS

Scorer. The GNN uses three message-passing rounds with hidden dimension 128 and dropout 0.1. We use temperature $\tau = 1$, consistency-penalty weight $\lambda = 1$, and a numerical floor of 10^{-6} on the score standard deviation. The projector has width 128 per input stream, and Q_θ applies layer normalization, a linear projection, and GELU before mean pooling over non-padding token positions.

Optimization. We train for 800 updates with Adam and an effective batch size up to 32 depending on model size, using a separate mask for each example. The learning rate warms up to 10^{-3} over 20 updates and then decays to 10^{-4} on a cosine schedule.

Budget schedule. To train one scorer for every sparsity, we cycle through retained-edge fractions $\{0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50\}\%$, one budget per update. Training uses continuous sigmoid gates with an exact total mass (Equation 9). Over the first half of training, the soft-mask budget follows a cosine schedule from 25% down to the selected target and then stays at the target whenever it is selected; targets above 25% are fixed throughout.

Evaluation. After training we average the per-example scores over the training split and select edges by hard top-K ranking at each budget (Section 4.1).

C FULL MIB BENCHMARK RESULTS

We report faithfulness and connectivity across sparsity budgets for all MIB tasks for the CPR results presented in Section 4.

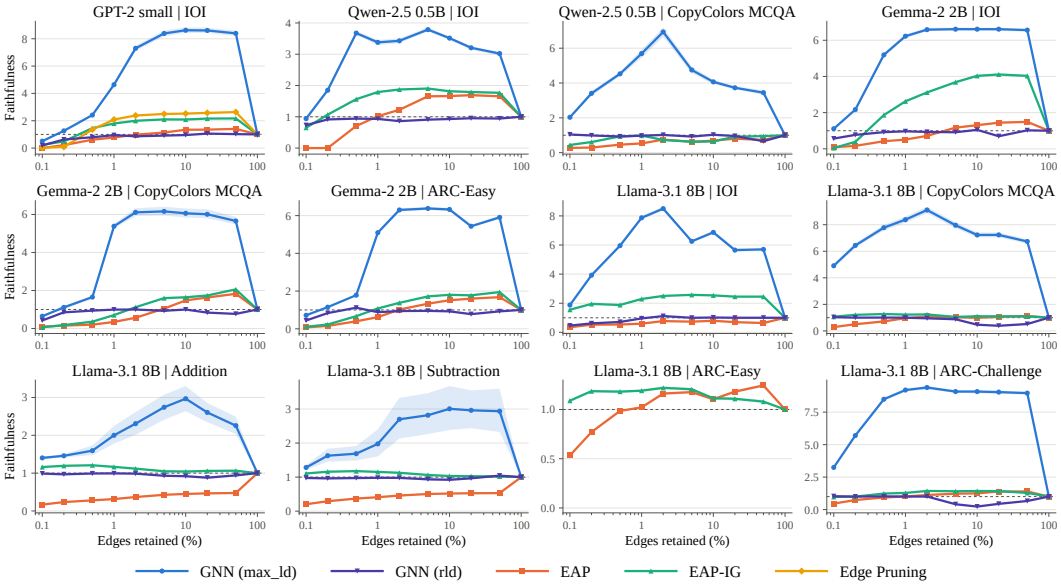


Figure 5: **Faithfulness against retained-edge fraction for every MIB model–task pair.** One panel per pair of Table 1; curves for the Max-LD scorer (legend: GNN (max_ld)), the scorer trained with the matching objective of Equation 13 (legend: GNN (rld)), EAP, EAP-IG, and, on GPT-2, Edge Pruning. Shaded bands are 95% bootstrap confidence intervals over test examples: at each budget, the task's test pairs are resampled with replacement 2,000 times, ratio-of-means faithfulness is recomputed on each resample, and the band spans the 2.5th to 97.5th percentiles.

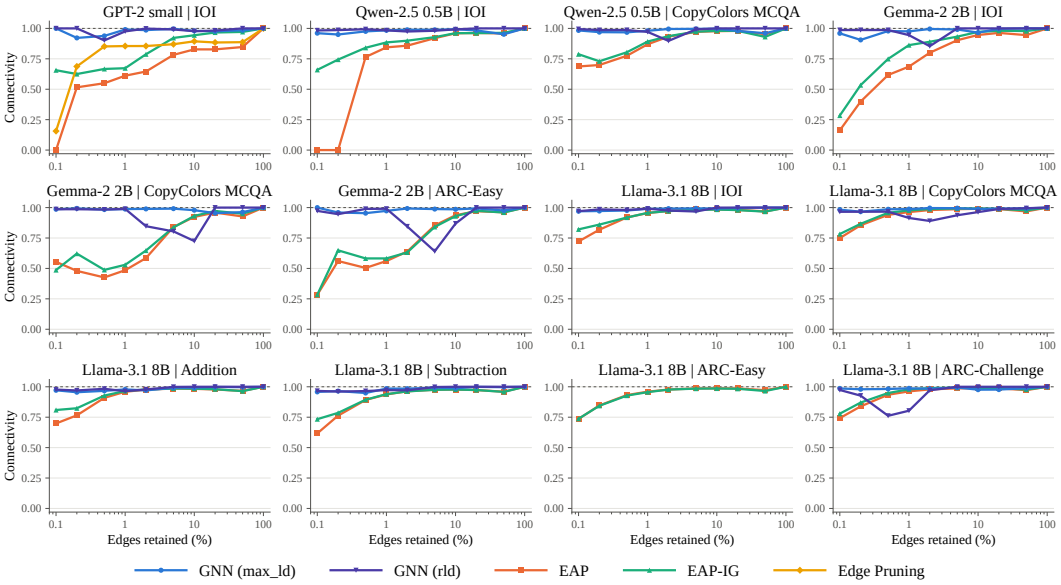


Figure 6: **Connectivity against retained-edge fraction for every MIB model–task pair.** Fraction of selected edges that lie on a complete path from the token embeddings to the logits, for the same pairs and methods as Figure 5.

Table 2: Normalized log-budget CMD (↓) on the MIB public test sets over retained-edge fractions from 0.1% to 100%. EAP and EAP-IG rank edges by absolute attribution; EAP-IG denotes input interpolation. Warm-start Edge Pruning uses absolute importance scores. GNN (Matching) uses the matching objective. Q: Qwen2.5-0.5B; G: Gemma-2-2B; L: Llama-3.1-8B. –: unavailable or not evaluated.

	IOI				MCQA			ARC-Easy		ARC-Challenge	Addition	Subtraction
Method	GPT-2	Q	G	L	Q	G	L	G	L	L	L	L
EAP	0.350	0.354	0.434	0.201	0.366	0.443	0.393	0.362	0.526	0.563	0.099	0.111
EAP-IG	0.249	0.171	0.295	0.096	0.346	0.427	0.370	0.312	0.374	0.357	0.022	0.034
Warm-start Edge Pruning	0.317	–	–	–	–	–	–	–	–	–	–	–
GNN (Matching)	0.163	0.083	0.121	0.125	0.073	0.109	0.074	0.128	0.233	0.255	0.045	0.038

D Training for Circuit Matching

If we want the circuit to reproduce the model's behavior rather than to just get the performance on a particular task, maximizing the logit difference may not necessarily be the objective to optimize for. Mueller et al. (2025) in addition to CPR proposes CMD, which instead of maximizing the integral of faithfulness across sparsity, measures the integrated absolute deviation of faithfulness from one. In other words CMD penalizes both under-recovery and overshoot.

We can modify our training objective accordingly:

$$\mathcal{L}_{\text{match}} = \frac{1}{B}\sum_{i=1}^{B} \frac{|\Delta(M_{p_i}, x_i) - \Delta(M, x_i)|}{\max(|\Delta(M, x_i) - \Delta(M_\emptyset, x_i)|, \epsilon)}, \tag{13}$$

where $M_\emptyset$ denotes the empty-circuit counterfactual baseline and $\epsilon = 10^{-3}$ prevents division by a near zero value.

As with CPR, for reporting CMD, we integrate over the logarithm of the retained-edge fraction to give greater weight to sparse circuits than linear-budget integration:

$$\text{CMD}_{\log} = \frac{1}{\log(b_{\max}/b_{\min})} \int_{b_{\min}}^{b_{\max}} |1 - f(b)| \; d\log b, \tag{14}$$

.

Table 2 reports the available results. Our GNN achieves lower CMD on many tasks, though no single method performs best across all tasks.

E Ablations

We lay out some of the minor architectural ablations we performed before settling on with our final architecture. We only experimented with the IOI task and the Greater-than task with GPT2 for 800 steps for these experiments.

E.1 Inputs

We can think of several different inputs to our Graph Neural Network. For one, we can just provide the activations for the example at hand and the component at hand. However, given that other methods take counterfactuals into account, providing counterfactuals in some form also seems reasonable. Hence, three candidates:

1. **Clean only:** the activations for the clean example.
2. **Clean + delta:** the clean activations, concatenated with the difference between the clean and counterfactual activations.
3. **Clean + corrupt + product:** the clean and counterfactual activations, concatenated with their element-wise product.

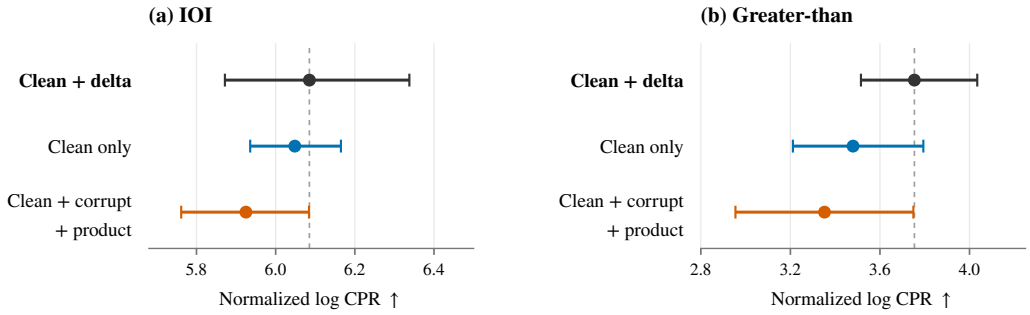


Figure 7: **Input-feature ablations on GPT-2.** We compare clean + delta features (baseline), clean-only features, and clean + corrupt features with their elementwise product, holding reverse message propagation fixed. Points show normalized log-budget CPR averaged over three training seeds; error bars show 95% bootstrap intervals over examples and seeds. The dashed line marks the baseline mean. Higher values indicate greater faithfulness.

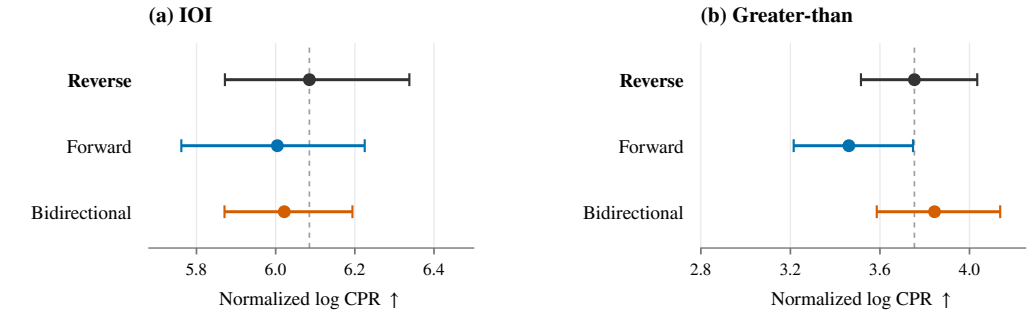


Figure 8: **Message-propagation direction ablations on GPT-2.** We compare reverse (baseline), forward, and bidirectional propagation, holding clean + delta input features fixed. Bidirectional propagation shares weights across directions and averages their updates; all variants have the same parameter count. Points show normalized log-budget CPR averaged over three training seeds; error bars show 95% bootstrap intervals over examples and seeds. The dashed line marks the baseline mean. Axis ranges match Figure 7.

7 illustrates that providing the original activations along with the delta between the original and the counterfactual ablations performs the best. However, note that only giving the model the original activations does not do too much worse.

E.2 Message Propagation Direction

In a Graph Neural Network, the direction of message propagation determines which components can inform the representation of a given component. We tried three options: forward, following the transformer's flow of information from inputs towards outputs; reverse, passing messages in the opposite direction; and bidirectional, passing messages in both directions. For the bidirectional variant, we share weights between the two directions and average their updates at each layer, keeping the parameter count the same across all three variants.

As shown in Figure 8, reverse propagation clearly performs better than forward propagation. The comparison between reverse and bidirectional propagation is less clear, with no clear advantage for either.

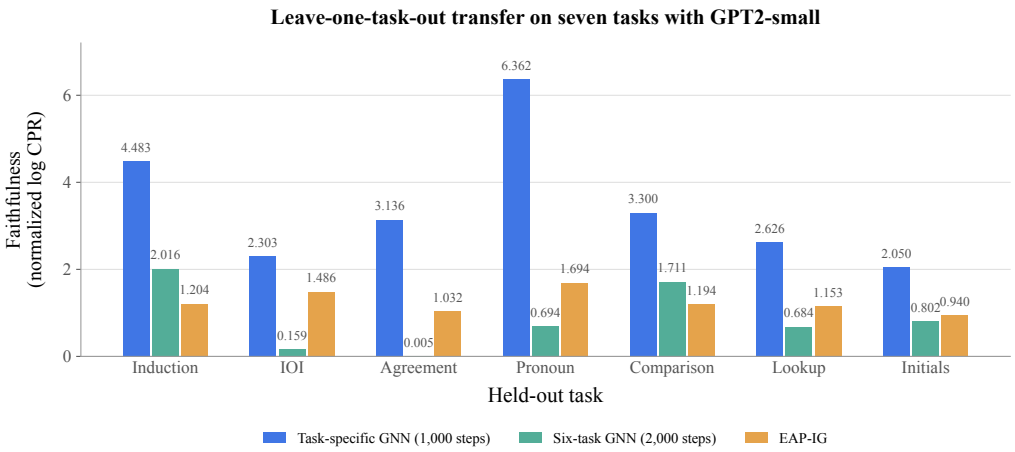


Figure 9: Held-out task performance after joint training on the other six tasks for 2,000 steps, compared with task-specific GNNs and EAP-IG.

F GENERALIZATION TO NEW TASKS

Although our experiments in 4 show that a shared model can perform well across multiple tasks, they do not say anything about generalization to unseen tasks. We evaluate this by training on a subset of the task families and see if it generalizes to unseen tasks. More specifically, we train jointly on six of the seven tasks for 2,000 steps and evaluate on the held-out task, repeating this procedure with each task held out in turn. Figure 9 compares held-out performance with EAP-IG and a task-specific GNN. Transfer varies substantially across tasks. These results suggest that transfer is possible but remains unreliable with the task diversity studied here. Whether training on a broader range of tasks improves generalization remains an open question.