SELF-DISCOVERING CIRCUIT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Circuit discovery is a central task in mechanistic interpretability, aiming to identify concise sets of transformer components responsible for a given behavior using either task-agnostic patching (e.g., zero, mean, resampling) or task-specific counterfactual patching. Obtaining concise circuits via *post-hoc* discovery methods is computationally demanding, while faster approximations often produce suboptimal, less faithful circuits. Moreover, patching combines activations in ways the model never encounters during training, which can distort the recovered circuits. To mitigate these limitations, we propose a self-supervised fine-tuning approach yielding *Self-Discovering Circuit Models* (SDCMs): transformers trained under task-agnostic patching to produce input-specific circuits for their own predictions as an integral part of their output, while preserving comparable predictive performance. Aggregated across examples, these input-level circuits can form a single circuit for an entire behavior. Empirically, under task-agnostic patching, SDCMs discover circuits far more efficiently than post-hoc methods, and these circuits are substantially more faithful and concise, with a single circuit outperforming post-hoc circuits extracted separately for each patching scheme. Moreover, although counterfactual patching is undefined on general training text, our training reshapes the model parameters so that post-hoc methods recover more faithful and concise circuits from SDCMs than from conventionally trained models under such patching. Lastly, we find that SDCM circuits remain superior to post-hoc circuits even on a conventionally trained model that never received SDCM training, indicating that their advantage lies also in the circuits themselves, and not only in the trained model. This opens the door to using SDCMs as "twin models" that efficiently supply concise and faithful circuits for their conventionally trained counterparts.

1 INTRODUCTION

Neural networks have grown rapidly in scale and capability, making their internal computations increasingly difficult to understand. In response, *mechanistic interpretability* (MI) has emerged as a central research direction, seeking to reverse-engineer trained networks into components and functional modules that people can reason about (Olah et al., 2020; Elhage et al., 2021; Bereska & Gavves, 2024). A central problem in MI is *circuit discovery* (Olah et al., 2020): identifying a subgraph, called a *circuit*, that is responsible for a behavior of interest (Wang et al., 2023; Conmy et al., 2023). Useful circuits should be both *concise*, containing only a small fraction of the model, and *faithful*, reproducing its behavior when components outside the circuit are intervened on.

Securing both properties, however, is difficult in the *post-hoc* setting, where circuits are sought after training with the model's parameters fixed. The first main difficulty is computational cost. Exhaustively testing candidate components is prohibitively expensive at transformer scale, while scalable approximations may recover circuits that are larger or less faithful than necessary (Conmy et al., 2023; Syed et al., 2024; Hanna et al., 2024; Mueller et al., 2025). A second difficulty comes from *patching*, the intervention used to evaluate circuit faithfulness by replacing activations outside the circuit. A variety of patching schemes have been proposed for choosing these replacements, including *task-agnostic* schemes such as zero, mean, and resampling, as well as *task-specific counterfactual* activations (Zhang & Nanda, 2024; Mueller et al., 2025). However, all such interventions can produce off-distribution activation combinations that the model did not encounter during training, even when the replacements individually come from real inputs. As a result, post-hoc circuits may partly reflect the model's behavior on these off-distribution activation combinations rather than the computation under study (Chan et al., 2022; Zhang & Nanda, 2024; Miller et al., 2024).

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.

Our contributions. Motivated by these limitations, we propose *Self-Discovering Circuit Models* (SDCMs), obtained through a self-supervised fine-tuning procedure that trains a pretrained language model to expose concise circuits for its own predictions. Trained jointly with a lightweight *circuit predictor* on general text, without task supervision, an SDCM produces two outputs for every input: its next-token prediction and an input-specific circuit for that prediction. Each training step performs two forward passes: a standard pass producing both outputs, and a second pass with all edges outside the circuit patched. The objective balances three goals: preserving the model's predictive performance, keeping the patched output faithful to the full model, and keeping the circuit concise. Since the model is trained on patched passes, these are less off-distribution. After training, circuits come from one forward pass and can be aggregated into task-level circuits at any density.

SDCM training is compatible with any patching scheme that can be defined on general, unlabeled text, such as resample, zero, or mean patching. Some interventions, however, are inherently task-specific: counterfactual patching requires knowing which features of the input to alter for the behavior under study, and thus cannot be used during training. In such settings, we train the SDCM under a task-agnostic scheme and then apply a standard post-hoc method to its trained backbone, rather than its circuit predictor, under the desired counterfactual intervention. SDCMs therefore support both direct circuit extraction by design and post-hoc extraction for task-specific interventions.

Empirically, both uses yield substantial improvements. Under task-agnostic patching, SDCM circuits are substantially more faithful and concise than post-hoc circuits, and are extracted much faster. Notably, a single SDCM circuit significantly outperforms post-hoc circuits extracted separately for each task-agnostic scheme, including schemes *never seen during training*. Under counterfactual patching, standard post-hoc methods recover more faithful and smaller circuits from SDCMs than from models trained with the language-modeling objective alone, while predictive performance remains comparable. Interestingly, we also find that SDCM circuits remain superior to post-hoc circuits even on a conventionally trained model that *never received SDCM training*, indicating that their advantage lies also in the circuits themselves, not only in the trained model. This opens the door to a third use, in which an SDCM serves as a "twin model" that efficiently supplies faithful and concise circuits for its conventionally trained counterpart.

More broadly, we regard this work as an important step toward highlighting an alternative direction for mechanistic interpretability: rather than treating circuit discovery solely as a difficult post-hoc search problem, models can be trained to expose concise representations of the computations behind their own predictions, while also making those computations better recovered using existing methods.

2 Circuit Discovery and the Shortcomings of Post-Hoc Methods

We first formalize circuit discovery (Section 2.1) and then examine two shortcomings of post-hoc methods that motivate our approach (Section 2.2).

2.1 Circuits, Patching, and Faithfulness

Notation. Let $f_G : \mathcal{X} \to \mathbb{R}^{|\mathcal{V}|}$ be a language model with parameters θ, mapping an input to next-token logits, and let $p_G(x) := \mathrm{softmax}(f_G(x))$ be the distribution it induces. We write $G := \langle V, E \rangle$ for its computation graph at the edge-level granularity standard in transformer circuit discovery (Conmy et al., 2023; Bhaskar et al., 2024; Mueller et al., 2025): V consists of the outputs (*writers*) and inputs (*readers*) of residual-stream components, with separate query, key, and value readers per head, one reader per MLP, an input writer, and a final reader producing the logits; E consists of the writer-to-reader edges. A *circuit* is a subgraph $C \subseteq G$ identified with a mask $m \in \{0,1\}^{|E|}$, where $m_e = 1$ retains edge e, with size $|C| := \sum_e m_e$ and *edge density* $d(C) := \frac{|C|}{|E|}$.

Circuit patching. Evaluating a circuit requires supplying values along the edges it discards, a procedure known as *patching* (Conmy et al., 2023). A *patching assignment* α specifies, for every writer j, a replacement α_j for its output y_j, so that the input to a reader r becomes

$$h_r(m, \alpha) := \sum_{e=(j \to r) \in E} \Big[m_e \, y_j + (1 - m_e) \, \alpha_j \Big]. \tag{1}$$

We write $f_C(x \mid \overline{C} = \alpha)$ for the resulting pass on x with the complement $\overline{C} := G \setminus C$ patched by α, and $p_C(x \mid \overline{C} = \alpha)$ for its softmax; $C = G$ recovers f_G. We consider two general families of patching schemes: (i) *Task-agnostic* schemes are defined on arbitrary text: *zero* patching sets $\alpha_j := 0$, *mean* patching $\alpha_j := \mathbb{E}_{x' \sim \mathcal{D}}[y_j(x')]$, and *resampling* $\alpha := y(\tilde{x})$ with $\tilde{x} \sim \mathcal{D}$, for a reference distribution $\mathcal{D}$ (Conmy et al., 2023; Zhang & Nanda, 2024). (ii) *Counterfactual* patching sets $\alpha := y(x')$, where x' is a minimal, task-specific corruption of x that alters the information the behavior depends on (Wang et al., 2023; Syed et al., 2024); constructing it presupposes knowing which part of the input carries the behavior, so it is defined only on the templated inputs of a given task. We write $\alpha \sim S(x)$ for the assignment that a *patching scheme* S supplies on input x.

Circuit faithfulness and conciseness. Conciseness is measured by $|C|$, or equivalently by $d(C)$. Faithfulness is measured by the KL divergence between $p_G(x)$ and $p_C(x \mid \overline{C} = \alpha)$, averaged over a dataset and over $\alpha \sim S(x)$ and written $\mathcal{M}_S(C)$ (Conmy et al., 2023; Hanna et al., 2024; Mueller et al., 2025). Since raw values are not comparable across models, tasks, or schemes, it is common to normalize by each model's own extremes (Marks et al., 2025; Mueller et al., 2025):

$$\mathcal{F}_S(C) := \frac{\mathcal{M}_S(C) - \mathcal{M}_S(\emptyset)}{\mathcal{M}_S(G) - \mathcal{M}_S(\emptyset)} \;=\; 1 - \frac{\mathcal{M}_S(C)}{\mathcal{M}_S(\emptyset)}, \tag{2}$$

where the second form uses $\mathcal{M}_S(G) = 0$, since the full circuit patches nothing. The empty circuit scores 0 and the full model 1. Circuit discovery then targets the following objective: given a tolerance τ and a scheme S, find a circuit of minimum $|C|$ among those satisfying $\mathcal{F}_S(C) \geq \tau$.

2.2 Why Post-Hoc Methods Fall Short

Post-hoc discovery methods pursue this objective on a fixed model, which raises two difficulties:

The search is intractable and approximate. This objective is a combinatorial optimization problem over the $2^{|E|}$ subsets of E, each requiring its own patched forward pass. Exhaustive search is out of reach at transformer scale, and many circuit queries are provably intractable in general (Adolfi et al., 2025), so efficient exact algorithms are unlikely to exist. Discovery at scale instead relies on approximations, typically retaining the top-ranked edges under attribution patching, which linearly estimates each edge's effect from gradients (Syed et al., 2024; Hanna et al., 2024; Mueller et al., 2025). These estimates score each edge separately, ignoring their interactions, and carry no optimality guarantee: the recovered circuits may be larger or less faithful than necessary.

Patching pushes the model off its data distribution. In a patched pass, each reader receives the clean input's contributions along retained edges and replacement values along all others (Eq. 1), a combination that no real input produces. This is clearest under zero patching, since components rarely output zero vectors (Chan et al., 2022; Zhang & Nanda, 2024), but it holds under every scheme: even when each replacement is a real activation, as under resampling or counterfactual patching, a reader mixes activations from two different inputs. A conventionally trained model never sees such inputs, so its behavior on them may not reflect the computation a circuit is meant to capture. A circuit may then appear unfaithful because replacement values activate components irrelevant to the input being explained, forcing additional edges to compensate, or appear faithful for reasons unrelated to how the prediction is made. Post-hoc methods cannot correct this, since they evaluate a *fixed* model: a more accurate search recovers these distortions more precisely rather than removing them. We instead train the model on patched passes (Section 3), so that these combinations are no longer unfamiliar to it. We expect the benefit to be largest where patching moves furthest from the data, and this is what we observe: the gains are greatest under zero patching (Section 4.1).

3 Training Self-Discovering Circuit Models (SDCMs)

Post-hoc methods face both obstacles of Section 2.2: they must search a fixed model for a concise faithful circuit, which is intractable in general, and cannot change how it behaves on patched activation combinations unseen in training. We aim to mitigate both during training instead. We fine-tune a pretrained language model jointly with a lightweight *circuit predictor*, yielding a *Self-Discovering Circuit Model* (SDCM) that produces *two outputs* for every input: its next-token prediction and a circuit for that prediction. The SDCM is trained to preserve its prediction when everything outside the circuit is patched (Figure 1). We call the fine-tuned model f_G, with parameters θ, its *backbone*.

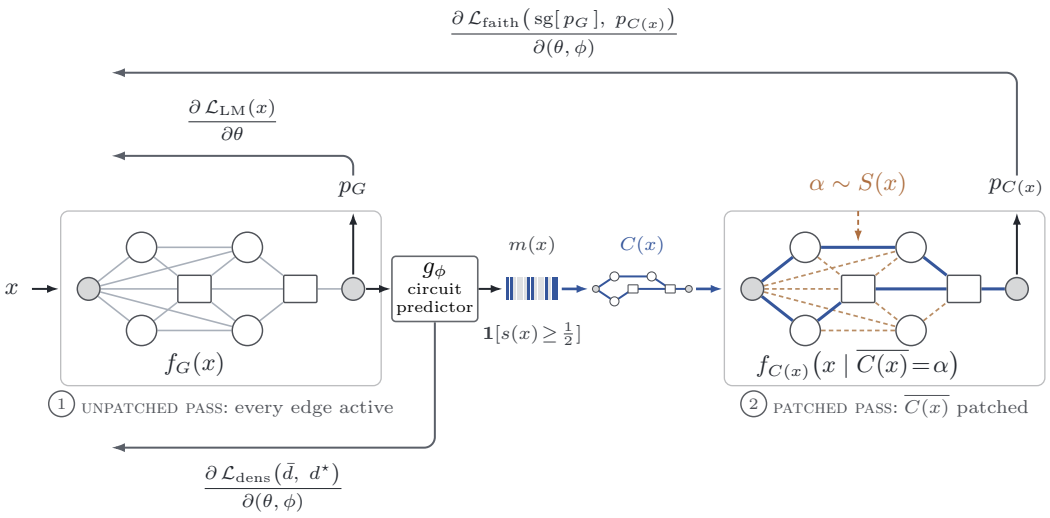


Figure 1: **Training an SDCM.** The circuit predictor g_ϕ turns the unpatched pass into a mask $m(x)$ that defines the circuit $C(x)$; a second pass runs $C(x)$ with its complement patched by $\alpha \sim S(x)$, giving $p_{C(x)}$. Grey arrows show where each loss of Eq. 4 sends its gradient.

3.1 Input-level Circuits as a Training Objective

Two forward passes. Each training step takes an input x with replacement activations $\alpha \sim S(x)$ from a task-agnostic patching scheme S. The first, *standard* pass runs the backbone on x, producing the next-token distribution $p_G(x)$ and the final residual-stream representation at the prediction position. The circuit predictor g_ϕ, a single affine layer with a sigmoid, maps this representation to edge scores $s(x) \in (0,1)^{|E|}$, and retaining the edges with $s_e(x) \geq \frac{1}{2}$ gives the mask $m(x)$ of the input-level circuit $C(x)$. The second, *patched* pass reruns the backbone on x with every edge outside $C(x)$ patched by α (Eq. 1), producing $p_C(x) := p_{C(x)}\big(x \mid \overline{C(x)} = \alpha\big)$. Gradients reach the scores through a straight-through estimator (Bengio et al., 2013).

Three training objectives. First, to remain a good language model, the SDCM keeps the next-token cross-entropy of $p_G(x)$ as a *language-modeling loss* $\mathcal{L}_{\text{LM}}$. Second, its circuits should be faithful, measured by a *faithfulness loss*, the KL divergence between full and patched predictions:

$$\mathcal{L}_{\text{faith}} := \mathrm{D}_{\text{KL}}\Big(\text{sg}\big[p_G(x)\big] \,\Big\|\, p_C(x)\Big). \tag{3}$$

This is the faithfulness metric of Eq. 2, computed on a single input. The stop-gradient sg keeps the full prediction fixed, so only the patched pass changes to match it. Both terms are trivially satisfied by keeping every edge, so a third goal is conciseness. We require the average soft density $\bar{d} := \mathbb{E}_x[\tilde{d}(x)]$, where $\tilde{d}(x) := |E|^{-1}\sum_e s_e(x)$ is the mean edge score, to match a target $d^\star$. Constraining only the average lets circuits grow or shrink with the input. We enforce this with an augmented Lagrangian (Wang et al., 2020; Bhaskar et al., 2024), which contributes a *density loss* $\mathcal{L}_{\text{dens}}$ with multipliers λ_1, λ_2 (Appendix B.3). With loss weights $\beta, \gamma > 0$, the full objective is

$$\mathcal{L}_{\text{AL}} := \mathbb{E}_{x,\,\alpha}\Big[\beta\,\mathcal{L}_{\text{LM}} + \gamma\,\mathcal{L}_{\text{faith}}\Big] + \mathcal{L}_{\text{dens}}, \quad \text{where} \quad \mathcal{L}_{\text{dens}} := \lambda_1 c + \lambda_2 c^2, \quad c := d^\star - \bar{d}. \tag{4}$$

Patching during training. The patched pass addresses the second obstacle of Section 2.2. Since training uses *general*, unlabeled text, the scheme S must be task-agnostic (Section 2.1); we train one SDCM each with resampling, mean, and zero patching. Replacements come from the standard pass itself, using another minibatch input under resampling and the minibatch mean under mean patching, so *patching requires no extra forward pass*. Unlike post-hoc methods, which fix the model, joint training lets faithfulness improve in two ways: the predictor can select better circuits, and the backbone can learn to behave consistently on patched activation combinations.

3.2 From Input-Level to Behavior-Level Circuits

An SDCM produces a separate circuit for each input, whereas circuit discovery usually seeks a single circuit for a behavior, such as indirect-object identification, shared across its inputs. We obtain one by aggregating the edge scores over examples $x_1, \ldots, x_n$ of the behavior in logit space,

$$\bar{\ell}_e := \frac{1}{n} \sum_{i=1}^{n} \operatorname{logit}\big(s_e(x_i)\big) \quad \text{for } e \in E, \tag{5}$$

which favors edges retained or discarded confidently across inputs. For any target density d, the behavior-level circuit keeps the $\lfloor d|E| \rfloor$ highest-ranked edges, after removing those that lie on no input-to-output path. A single ranking thus yields circuits at every density, computed with one standard pass per example and no reference inputs, gradients, or task-specific optimization.

3.3 Beyond Self-Discovery on SDCMs

The backbone as a host for post-hoc discovery. Some interventions cannot be used during training: counterfactual patching, for instance, requires task-specific input pairs that general text does not provide. Training nonetheless changes the backbone itself, not only the predictor, and these changes reside in its parameters, where any discovery method can benefit from them. Since counterfactual patching also combines activations from different inputs (Section 2.2), we apply standard post-hoc methods such as EAP-IG to the trained backbone under such interventions, extending the benefits of SDCM training to schemes it never saw (Section 4.2).

The predictor's circuits for other models. In practice, one often wants to explain a model that cannot be modified, such as one already deployed. Since fine-tuning leaves the computation graph unchanged, an SDCM's circuits are also circuits of any model with the same architecture, including the model it was fine-tuned from or a conventionally trained counterpart. An SDCM can thus serve as a companion to such a model, supplying its circuits through a single forward pass per example while the model itself remains unchanged (Section 4.3).

4 Experimental Evaluation

Experimental setup. We fine-tune Qwen2.5-0.5B and GPT-2 Small (hereafter GPT-2) from MIB (Mueller et al., 2025) on general text with target density $d^\star = 3\%$, training three SDCMs per model, one each with resample, zero, and mean patching. As a control, we train each model for the same number of steps on the same data with $\mathcal{L}_{\mathrm{LM}}$ alone, which we call *standard* training. We evaluate four standard tasks from prior work: IOI (Wang et al., 2023) and SVA (Finlayson et al., 2021; Marks et al., 2025) on both models, Greater-Than (Hanna et al., 2023) on GPT-2, and MCQA (Mueller et al., 2025) on Qwen2.5-0.5B. For each task, we build a ranking from 100 examples and evaluate its circuits on held-out examples at densities from 0.1% to 100%, reporting the standard circuit discovery metric of *area under the faithfulness curve* (AUC) (Mueller et al., 2025). We compare against two state-of-the-art attribution methods that also produce edge rankings, EAP-IG-activations (Marks et al., 2025; Hanna et al., 2024) and EAP-IG-inputs (Hanna et al., 2024); the latter is undefined under zero and mean patching. SDCM training leaves language-modeling and task performance largely unchanged, with only resample training slightly increasing perplexity and mean-trained SDCMs even slightly exceeding standard training on several tasks (Appendix B.5); full details are in Appendices B and C.

4.1 Inherent SDCM Circuits vs. Post-Hoc Circuits of Standard Models

For each task, we extract a single ranking from each of the three SDCMs, one per training patching scheme, and run EAP-IG on the standard-trained model separately under each patching scheme, yielding one EAP-IG circuit per extraction scheme. We then evaluate every resulting circuit under all three task-agnostic schemes, so that each column of Table 1 compares the SDCMs with EAP-IG extracted under the evaluated scheme as well as under the others.

SDCM circuits are more faithful at matched density. Table 1 shows that every SDCM circuit outperforms every EAP-IG circuit on every task and under every scheme, including EAP-IG circuits

Table 1: AUC of normalized KL faithfulness (%, higher is better) and extraction time (seconds, lower is better) on Qwen2.5-0.5B. Ext. is the scheme under which EAP-IG is extracted on the standard-trained model; Train is the scheme under which each SDCM is trained. Every circuit is evaluated under all schemes (columns). **Gold**, silver, and bronze mark the top three per column (ties share a medal); bold marks the fastest.

Method	Scheme		IOI				SVA				MCQA			
	Ext.	Train	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)
EAP-IG-inp.	Res.	–	81.1	55.4	83.9	38.6	93.6	76.4	91.0	15.4	82.6	61.5	86.6	66.8
EAP-IG-act.	Res.	–	78.4	50.1	80.6	2477.8	93.9	69.1	91.1	804.1	79.0	55.3	85.2	4321.5
	Zero	–	71.5	56.5	75.6	2472.0	74.9	54.1	82.4	804.0	66.8	49.5	80.9	4309.6
	Mean	–	77.6	53.1	83.8	3034.8	88.0	75.6	92.1	756.6	83.7	69.6	88.4	4740.6
SDCM (ours)	–	Res.	**90.5**	72.6	90.3	**7.1**	**97.5**	88.7	96.5	**6.1**	**94.2**	86.5	93.9	**10.1**
	–	Zero	86.3	**87.0**	87.5	**6.9**	96.0	**96.5**	94.7	**6.0**	91.8	**94.0**	93.2	**8.0**
	–	Mean	90.4	72.7	**91.4**	**6.8**	97.2	86.0	**96.7**	**5.8**	93.7	86.5	**94.9**	**7.6**

extracted under the very scheme being evaluated, often by a wide margin, and Figure 2 shows that these gains hold across most densities. GPT-2 shows the same picture in most of its task–scheme combinations (Appendix D). In Appendix H, we further examine how the training density target affects faithfulness across evaluation densities.

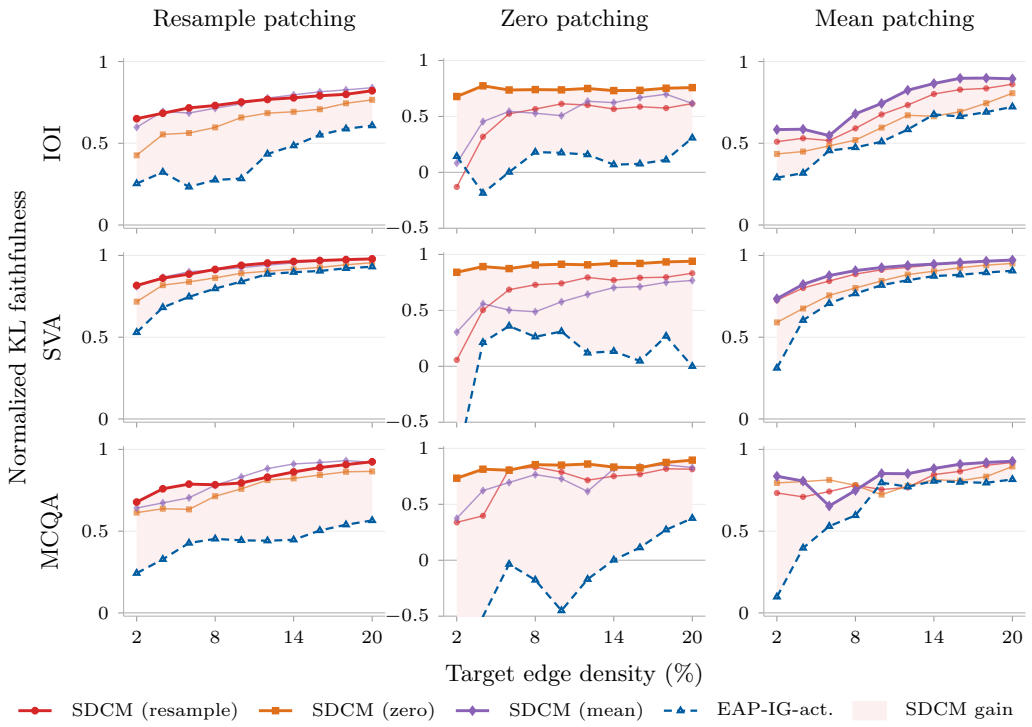


Figure 2: Normalized KL faithfulness on Qwen2.5-0.5B at target densities from 2% to 20%. Thicker SDCM curves indicate matching training and evaluation patching; other variants are faded. EAP-IG-act. uses the standard-trained model. Zero-patching panels clip values below -0.5.

A single circuit transfers across patching schemes. Strikingly, SDCM circuits outperform EAP-IG circuits even under patching schemes they *never saw during training*, and even when EAP-IG is extracted under those very same schemes. Post-hoc circuits, by contrast, are tied to their extraction scheme: on IOI, resample-extracted EAP-IG-act. circuits drop from 78.4 to 50.1 under zero patching. Nor does matching the scheme reliably help: under zero patching, zero-extracted

EAP-IG-act. circuits are its weakest on SVA and MCQA. The gap is largest under zero patching, which moves furthest from the data distribution (Section 2.2): the zero-trained SDCM reaches 87.0, 96.5, and 94.0 on IOI, SVA, and MCQA, against at most 56.5, 76.4, and 69.6 for any EAP-IG circuit. Matching training and evaluation patching schemes gives the best result in every column, yet even a mismatched SDCM stays well ahead of post-hoc extraction.

Extraction is substantially cheaper. Since SDCM circuits come from forward passes alone, extraction requires no reference inputs or gradients, and a single ranking serves all three schemes. It is thus significantly faster than EAP-IG-inp. and EAP-IG-act., which must moreover be rerun for each scheme. The one-time training cost is amortized over all subsequent extractions (Appendix G).

4.2 Post-Hoc Counterfactual Discovery: SDCMs vs. Standard Models

Counterfactual patching requires task-specific input pairs, which general text cannot provide, so it cannot be used during training. However, like task-agnostic patching, it combines activations from different inputs (Section 2.2), so SDCM training may still make its circuits easier to find. We test this by running EAP-IG-act. on the backbone of the resample-trained SDCM, without its circuit predictor, and on the standard-trained model, under established task-specific counterfactual interventions from prior work (Mueller et al., 2025; Marks et al., 2025; Hanna et al., 2023) (Appendix C.3).

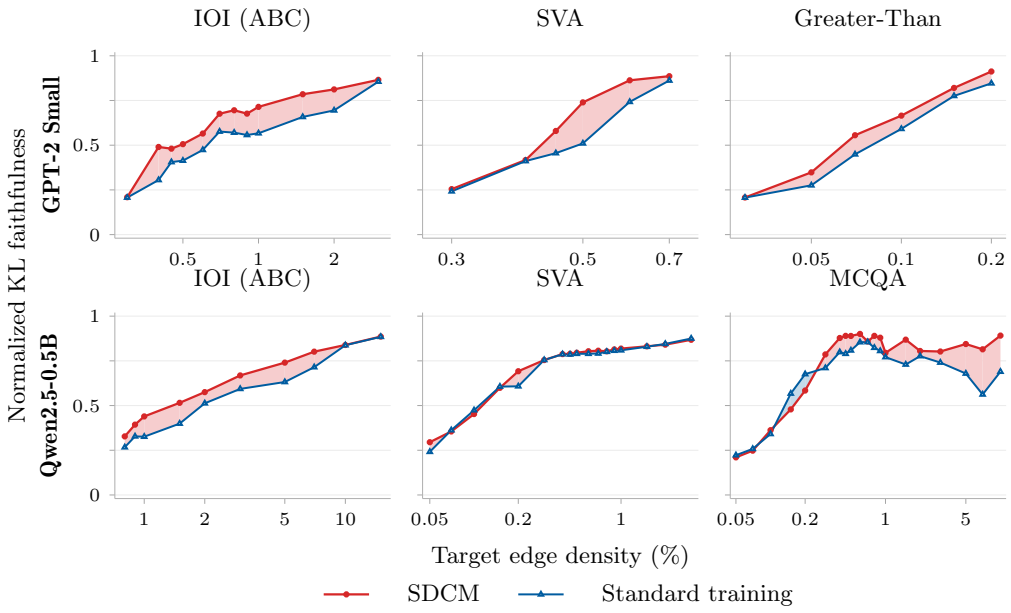


Figure 3: Normalized KL faithfulness of EAP-IG-act. circuits under counterfactual patching on the SDCM and the standard-trained model, where the latter's faithfulness is in $[0.2, 0.9]$.

Post-hoc circuits are more faithful and smaller. As Figure 3 shows, circuits extracted from SDCMs are generally more faithful than those extracted from a standard-trained model at the same size. The gain is clearest as a reduction in size at fixed faithfulness. On Qwen2.5-0.5B's IOI, matching the standard model's faithfulness takes $5{,}205$ rather than $8{,}872$ edges, $1.7\times$ fewer. Overall, EAP-IG-act. achieves a higher AUC on SDCMs in five of the six settings (Appendix E).

Where the empirical gains are largest, and why. Improvements are largest at densities where the standard model's circuits recover only part of the behavior, and shrink as circuits approach full faithfulness. This is the setting of most interest for interpretability: a circuit is only informative if it is much smaller than the model. Since the gains hold under interventions unseen in training, and for a discovery method other than the predictor, they indicate that SDCM training reshapes the backbone. Results for GPT-2, EAP-IG-inp., and all eight IOI counterfactuals are in line (Appendix E).

A few edges can make the difference. Figure 4 compares the SVA circuits that EAP-IG-act. extracts from the SDCM (151 edges) and from the standard model (153 edges). The two share 139 edges and differ in only 26, yet the SDCM circuit is far more faithful (0.740 vs. 0.487), and remains

so *even on the standard model itself* (0.705 vs. 0.510), a transfer we examine further in Section 4.3. Swapping just a handful of edges thus cuts the circuit's KL divergence from the full model by 40–50%, whichever model runs it. An additional IOI example is given in Appendix I.

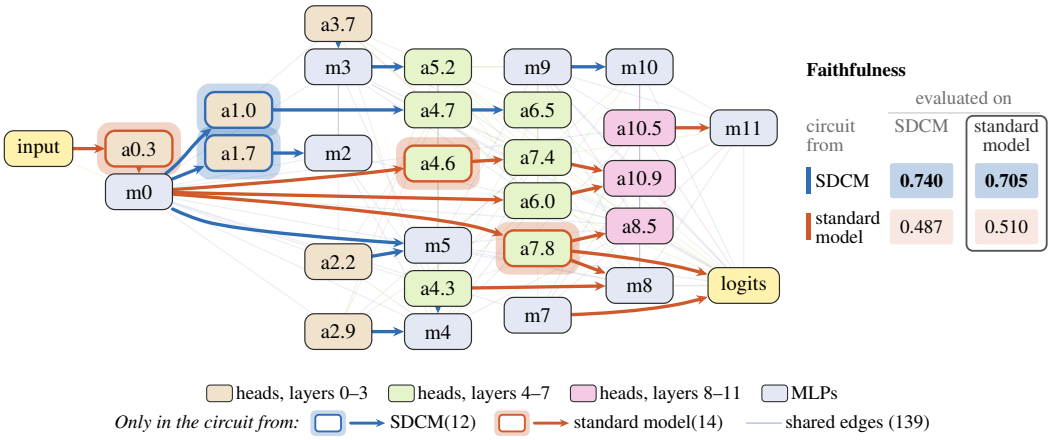


Figure 4: **Post-hoc circuits from an SDCM and from a standard model.** The SDCM's circuit is both smaller (151 vs. 153 edges) and more faithful, *even on the standard model itself* (right), with the two differing in only 26 edges (bold). Both are EAP-IG-act. circuits for SVA on GPT-2 under counterfactual patching. Shared edges are faint; six nodes with identical connections are omitted.

4.3 SDCM Circuits on Conventionally Trained Models

SDCM training shapes both the circuits and the model that runs them, and both contribute to the results of Sections 4.1 and 4.2. However, one might argue that the faithfulness of SDCM circuits stems mainly from the model having been trained alongside them, rather than from the circuits themselves. We now ask whether the circuits are strong on their own, independently of the model they were trained with. To this end, we take the circuits produced by the SDCMs and evaluate them on the standard-trained model, *which never received SDCM training*, comparing them with EAP-IG circuits extracted directly from that same model. Results are depicted in Table 2.

Table 2: AUC of normalized KL faithfulness (%, higher is better) and extraction time (seconds, lower is better) on Qwen2.5-0.5B, with *all circuits evaluated on the standard-trained model*, from which EAP-IG circuits are also extracted; SDCM circuits are used unchanged. Ext. and Train are as in Table 1. **Gold**, silver, and bronze mark the top three per column; bold marks the fastest method.

Method	Scheme		IOI				SVA				MCQA			
	Ext.	Train	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)
EAP-IG-inp.	Res.	–	81.1	55.4	83.9	38.6	93.6	76.4	91.0	15.4	82.6	61.5	86.6	66.8
EAP-IG-act.	Res.	–	78.4	50.1	80.6	2477.8	93.9	69.1	91.1	804.1	79.0	55.3	85.2	4321.5
	Zero	–	71.5	56.5	75.6	2472.0	74.9	54.1	82.4	804.0	66.8	49.5	80.9	4309.6
	Mean	–	77.6	53.1	83.8	3034.8	88.0	75.6	92.1	756.6	83.7	69.6	88.4	4740.6
SDCM (ours)	–	Res.	**89.7**	68.6	90.0	**7.1**	**97.0**	85.0	95.8	**6.1**	**94.3**	86.1	94.4	**10.1**
	–	Zero	85.9	**84.0**	87.4	**6.9**	95.9	**95.2**	94.6	**6.0**	91.7	**94.5**	93.3	**8.0**
	–	Mean	89.6	69.5	**91.0**	**6.8**	96.8	84.3	**96.3**	**5.8**	93.5	85.3	**95.0**	**7.6**

SDCM circuits remain stronger even on standard-trained models. Table 2 shows that SDCM circuits outperform the standard model's own EAP-IG circuits on every task and under every patching scheme, and are nearly as faithful as on the SDCMs themselves. Their advantage therefore lies in the circuits they identify, not only in the adapted model. Results on GPT-2 are largely consistent (Appendix F). This suggests an interesting practical use-case: an SDCM can serve as a "twin model" of a conventionally trained model, supplying faithful and concise circuits for it at the cost of a forward pass, instead of a separate post-hoc search per patching scheme.

5 Related Work

Post-hoc circuit discovery. Circuits were first studied in convolutional vision models (Olah et al., 2020) and later in transformers (Elhage et al., 2021; Wang et al., 2023). Automated discovery methods include greedy patching (Conmy et al., 2023), attribution patching (Nanda, 2023; Syed et al., 2024) including its integrated-gradients variant EAP-IG (Hanna et al., 2024), and learned edge masks (Bhaskar et al., 2024; Li & Janson, 2024). Later work extends them to token positions and sparse features (Haklay et al., 2025; Marks et al., 2025; Dunefsky et al., 2024). All of them seek concise, faithful circuits, yet faithfulness depends on the patching scheme and metric (Zhang & Nanda, 2024; Miller et al., 2024), and recovered circuits are often unstable (Méloux et al., 2025; Bayat Makou et al., 2026). These methods search a fixed model, whereas SDCMs train the model to produce its own circuits (see Appendix A for an extended discussion).

Training for circuit analysis. Closer to our work, some methods add structural constraints during training so that circuits are easier to analyze, such as sparse weights (Gao et al., 2026) or attention (Draye et al., 2025), routed gradients (Cloud et al., 2024), or architectural modifications (Tamkin et al., 2024; Pearce et al., 2025). On the resulting simplified models, circuits are then discovered and analyzed entirely separately, post hoc. SDCMs instead extract circuits inherently, by training directly on the circuit-discovery objective so that each prediction comes with its own circuit.

Self-explaining neural networks. In the *explainability* literature, self-explaining neural networks output an input explanation alongside each prediction (Alvarez-Melis & Jaakkola, 2018; Ji et al., 2025), typically as feature attributions, input subsets, or concepts (Agarwal et al., 2021; You et al., 2025; Bassan et al., 2025; Koh et al., 2020). SDCMs are related in spirit, but whereas these models explain a prediction through its inputs, SDCMs expose the subgraph of the model's own computation behind it; to our knowledge, ours is the first work to pursue a similar direction for circuit discovery.

6 Limitations and Future Work

A central limitation of SDCM training is the need to tune extra hyperparameters arising from its multiple objectives, which we mitigated with a staged search, tuning one group of settings at a time (Appendix B.4). The dual-pass training is also roughly twice as costly as standard fine-tuning (Appendix G). However, this cost is paid only once, and training is performed *offline*; afterwards, circuit extraction is substantially faster than post-hoc discovery: it requires a single forward pass per example, the same model serves all tasks, and one ranking can serve many task-agnostic patching schemes (Section 4.1). A further limitation is that SDCM training requires a patching scheme definable on general text, precluding counterfactual patching; still, its benefits extend to this setting through the model's parameters (Section 4.2). Lastly, while SDCM training leaves language-modeling and task performance largely unchanged, resample training slightly increases perplexity (Appendix B.5). Interestingly, mean-trained SDCMs never fall below standard training on any task and slightly exceed it on several. Future work could investigate this effect further, to assess whether SDCM training can serve not only to produce inherent circuits but also to improve predictive performance. Another promising direction is training SDCMs from scratch rather than by fine-tuning.

7 Conclusion

Concise and faithful circuits are a central goal of mechanistic interpretability but are difficult to discover post hoc: the search is costly, and patching combines activations in ways the model never sees in training. We mitigate these limitations with *Self-Discovering Circuit Models* (SDCMs), trained via self-supervised fine-tuning on general text to produce circuits for their own predictions. Under task-agnostic patching, SDCM circuits are substantially more faithful and concise than post-hoc circuits and extracted significantly faster, with a single circuit outperforming those extracted separately for each patching scheme. For counterfactual interventions unavailable during training, post-hoc methods still recover better circuits from SDCMs than from conventionally trained models. SDCM circuits also remain superior on a model that never received SDCM training, so an SDCM can serve as a "twin model" that supplies circuits for its conventionally trained counterpart. Overall, we view this work as an important step toward treating a model's amenability to circuit analysis as something shaped during training, rather than solely searched for after the fact.

AI USE STATEMENT

Generative AI tools were used in this work to implement methods, specifically basic components of our pipeline such as helper functions, baseline implementations, data-loading code, and plotting scripts for figures. They were not used to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on the research methodology or experiments, or to interpret results. Since this work contains no theoretical results, formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable. Likewise, generating synthetic data sets, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. We additionally used generative AI tools to create and edit software code, to create and modify scientific figures, and to edit the paper for readability (rephrasing and polishing). Two authors carefully reviewed, edited, and tested all AI-generated code, and carefully reviewed and edited all AI-generated text. We take full responsibility for all content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Federico Adolfi, Martina G. Vilas, and Todd Wareham. The computational complexity of circuit discovery for inner interpretability. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.08025.

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E. Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4699–4711, 2021.

David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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.

Alaa Anani, Tobias Lorenz, Bernt Schiele, Mario Fritz, and Jonas Fischer. Certified circuits: Stability guarantees for mechanistic circuits. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2602.22968.

Steve Azzolin, Sagar Malhotra, Andrea Passerini, and Stefano Teso. Beyond topological self-explainable GNNs: A formal explainability perspective. In *Proceedings of the 42nd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2025.

Shahaf Bassan, Ron Eliav, and Shlomit Gur. Explain yourself, briefly! Self-explaining neural networks with concise sufficient reasons. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alireza Bayat Makou, Jingcheng Niu, Subhabrata Dutta, and Iryna Gurevych. Many circuits, one mechanism: Input variation and evaluation granularity in circuit discovery. *arXiv preprint arXiv:2606.06267*, 2026.

Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. Discovering knowledge-critical subnetworks in pretrained language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6549–6583. Association for Computational Linguistics, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. doi: 10.48550/arXiv.1308.3432. URL https://arxiv.org/abs/1308.3432.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for AI safety – a review. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ePUVetPKu6.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 18506–18534. Curran Associates, Inc., 2024. doi: 10.52202/079017-0587. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/20fdaf67581e6d7157376d1ed584040a-Abstract-Conference.html.

Moritz Böhle, Mario Fritz, and Bernt Schiele. B-cos networks: Alignment is all we need for interpretability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10329–10338, 2022.

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Nick Cammarata, Gabriel Goh, Shan Carter, Ludwig Schubert, Michael Petrov, and Chris Olah. Curve detectors. *Distill*, 2020. doi: 10.23915/distill.00024.003. URL https://distill.pub/2020/circuits/curve-detectors.

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.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. AI Alignment Forum, 2022. URL https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 883–892. PMLR, 2018.

Pratim Chowdhary, Peter Chin, and Deeparnab Chakrabarty. K-MSHC: Unmasking minimally sufficient head circuits in large language models with experiments on syntactic classification tasks. *arXiv preprint arXiv:2505.12268*, 2025.

Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks. *arXiv preprint arXiv:2410.04332*, 2024.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pp. 16318–16352. Curran Associates, Inc., 2023. doi: 10.52202/075280-0719. URL https://papers.nips.cc/paper_files/paper/2023/hash/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Abstract-Conference.html.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Are neural nets modular? Inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*, 2021.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? Interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 3243–3255. Association for Computational Linguistics, 2020.

Florent Draye, Anson Lei, Hsiao-Ru Pan, Ingmar Posner, and Bernhard Schölkopf. Intrinsically interpretable attention via sparse post-training, 2025. URL https://arxiv.org/abs/2512.05865v5. Version 5, revised May 2026.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *Advances in Neural Information Processing Systems*, volume 37, pp. 24375–24410, 2024.

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. URL https://transformer-circuits.pub/2021/framework/index.html.

Javier Ferrando and Elena Voita. Information flow routes: Automatically interpreting language models at scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17432–17445. Association for Computational Linguistics, 2024.

Jeremias Ferrao, Luhan Mikaelson, Keenan Pepper, and Natalia Perez-Campanero Antolin. Self-ablating transformers: More interpretability, less sparsity. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. arXiv:2505.00509.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1828–1843. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.144. URL https://aclanthology.org/2021.acl-long.144/.

Gabriel Franco, Lucas M. Tassis, Azalea Rohr, and Mark Crovella. Finding interpretable prompt-specific circuits in language models. In *Mechanistic Interpretability Workshop at ICML 2026*, 2026. arXiv:2602.13483.

Dan Friedman, Alexander Wettig, and Danqi Chen. Learning transformer programs. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Jose Gallego-Posada, Juan Ramirez, Akram Erraqabi, Yoshua Bengio, and Simon Lacoste-Julien. Controlled sparsity via constrained optimization or: How I learned to stop tuning penalties and love constraints. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2208.04425.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V. Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. Inducing causal structure for interpretable neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7324–7338. PMLR, 2022.

Rohan Gupta, Iván Arcuschin, Thomas Kwa, and Adrià Garriga-Alonso. InterpBench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 37, 2024.

Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. Position-aware automatic circuit discovery. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2792–2817. Association for Computational Linguistics, 2025.

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*, volume 36, pp. 76033–76060. Curran Associates, Inc., 2023. doi: 10.52202/075280-3322. URL https://papers.nips.cc/paper_files/paper/2023/hash/efbba7719cc5172d175240f24be11280-Abstract-Conference.html.

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://openreview.net/forum?id=TZ0CCGDcuT.

Aliyah R. Hsu, Georgia Zhou, Yeshwanth Cherapanamjeri, Yaxuan Huang, Anobel Y. Odisho, Peter R. Carroll, and Bin Yu. Efficient automated circuit discovery in transformers using contextual decomposition. In *The Thirteenth International Conference on Learning Representations*, 2025.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4459–4473. Association for Computational Linguistics, 2020.

Yang Ji, Ying Sun, Yuting Zhang, Zhigaoyuan Wang, Yuanxin Zhuang, Zheng Gong, Dazhong Shen, Chuan Qin, Hengshu Zhu, and Hui Xiong. A comprehensive survey on self-interpretable neural networks. *Proceedings of the IEEE*, 2025. doi: 10.1109/JPROC.2025.3635153.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 2020.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 107–117. Association for Computational Linguistics, 2016.

Maximilian Li and Lucas Janson. Optimal ablation for interpretability. In *Advances in Neural Information Processing Systems*, volume 37, pp. 109233–109282. Curran Associates, Inc., 2024. doi: 10.52202/079017-3468. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c55e6792923cc16fd6ed5c3f672420a5-Abstract-Conference.html.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *The Thirteenth International Conference on Learning Representations*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Philipp Mondorf, Sondre Wold, and Barbara Plank. Circuit compositions: Exploring modular structures in transformer-based language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14934–14955. Association for Computational Linguistics, 2025.

Aaron Mueller, Atticus Geiger, Sarah Wiegreffe, Dana Arad, Iván Arcuschin, Adam Belfki, Yik Siu Chan, Jaden Fried Fiotto-Kaufman, Tal Haklay, Michael Hanna, Jing Huang, Rohan Gupta, Yaniv Nikankin, Hadas Orgad, Nikhil Prakash, Anja Reusch, Aruna Sankaranarayanan, Shun Shao, Alessandro Stolfo, Martin Tutek, Amir Zur, David Bau, and Yonatan Belinkov. MIB: A mechanistic interpretability benchmark. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45069–45108. PMLR, 2025. URL https://proceedings.mlr.press/v267/mueller25a.html.

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

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024.001, 2020. doi: 10.23915/distill.00024.001. URL https://distill.pub/2020/circuits/zoom-in.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Michael T. Pearce, Thomas Dooms, Alice Rigg, Jose M. Oramas, and Lee Sharkey. Bilinear MLPs enable weight-based mechanistic interpretability. In *The Thirteenth International Conference on Learning Representations*, 2025.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David Blei. Hypothesis testing the circuit hypothesis in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 94539–94567, 2024.

Jiuding Sun, Jing Huang, Sidharth Baskaran, Karel D'Oosterlinck, Christopher Potts, Michael Sklar, and Atticus Geiger. HyperDAS: Towards automating mechanistic interpretability with hypernetworks. In *The Thirteenth International Conference on Learning Representations*, 2025.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 407–416. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.blackboxnlp-1.25. URL https://aclanthology.org/2024.blackboxnlp-1.25/.

Alex Tamkin, Mohammad Taufeeque, and Noah D. Goodman. Codebook features: Sparse and discrete interpretability for neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2024.

Curt Tigges, Michael Hanna, Qinan Yu, and Stella Biderman. LLM circuit analyses are consistent across training and scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Kevin Ro 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 *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

(EMNLP), pp. 6151–6162. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.496. URL https://aclanthology.org/2020.emnlp-main.496/.

Weiqiu You, Helen Qu, Marco Gatti, Bhuvnesh Jain, and Eric Wong. Sum-of-parts: Self-attributing neural networks with end-to-end learning of feature groups. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 72747–72785. PMLR, 2025.

Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations*, 2021.

Lei Yu, Jingcheng Niu, Zining Zhu, Xi Chen, and Gerald Penn. Sheaf discovery with joint computation graph pruning and flexible granularity. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8822–8837. Association for Computational Linguistics, 2025.

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.

Appendix

Appendix Contents

A Extended Related Work **17**

B Training Specifications **17**

B.1 Models and Training Variants 18

B.2 Training Data and Sampling 18

B.3 Training Procedure 18

B.4 Hyperparameter Selection and Training Configurations 19

B.5 Capability of the Trained Models 20

C Experimental Setup and Evaluation Details **21**

C.1 Task Datasets and Examples 21

C.2 Data Selection and Preprocessing 22

C.3 Patching Protocols 22

C.4 Circuit Extraction and Evaluation 23

D Circuit Faithfulness under Task-Agnostic Patching **24**

D.1 GPT-2 Results under Task-Agnostic Patching 24

D.2 Comparisons within Trained SDCMs 25

E SDCM Training and Post-Hoc Circuit Discovery **25**

F SDCM Circuits on Conventionally Trained Models **27**

G Runtime and Computational Cost **28**

H Training Density and Circuit Size **29**

I Additional Case Studies of Extracted Circuits **30**

A Extended Related Work

Post-hoc circuit discovery. Circuits were first reverse-engineered by hand in convolutional vision models (Cammarata et al., 2020; Olah et al., 2020), and the residual-stream view of Elhage et al. (2021) carried this program to transformers, where early circuits such as induction heads, indirect-object identification, and greater-than comparison were traced with activation patching (Olsson et al., 2022; Wang et al., 2023; Hanna et al., 2023). Automated methods followed. ACDC prunes edges greedily (Conmy et al., 2023), attribution patching and its refinements estimate edge effects from gradients (Nanda, 2023; Syed et al., 2024; Kramár et al., 2024; Hanna et al., 2024; Haklay et al., 2025), and other methods score components without a separate patched pass (Hsu et al., 2025; Ferrando & Voita, 2024; Franco et al., 2026). Mask-based methods, which grew out of differentiable masking and subnetwork analysis (De Cao et al., 2020; Csordás et al., 2021; Cao et al., 2021; Bayazit et al., 2024), learn a sparse edge mask for each task (Bhaskar et al., 2024; Li & Janson, 2024; Yu et al., 2025), and feature-level circuits replace components with sparse-autoencoder or transcoder latents (Marks et al., 2025; Dunefsky et al., 2024; Ameisen et al., 2025). Our training objective has an aspect that resembles the augmented-Lagrangian density constraint of the mask family (Wang et al., 2020; Bhaskar et al., 2024), but optimizes the model jointly with an input-conditioned predictor on general text, rather than a fixed mask on a fixed model.

Conciseness and faithfulness. Smaller circuits are generally considered more informative, and several works study minimality directly: Adolfi et al. (2025) show that many circuit queries are intractable, Chowdhary et al. (2025) search for minimally sufficient sets of heads, and Shi et al. (2024) find that published circuits satisfy sufficiency and minimality criteria only partially. Conciseness, however, is only meaningful relative to a faithfulness criterion, and that criterion is fragile: it depends on the patching scheme and metric (Zhang & Nanda, 2024; Miller et al., 2024; Hanna et al., 2024), and recovered circuits are often unstable across extraction runs and inputs (Méloux et al., 2025; Bayat Makou et al., 2026; Anani et al., 2026). Benchmarks such as MIB and InterpBench (Mueller et al., 2025; Gupta et al., 2024) standardize evaluation, and studies of circuit consistency across tasks, scales, and training (Merullo et al., 2024; Tigges et al., 2024; Mondorf et al., 2025) rely on stable extraction. Section 2.2 traces part of this fragility to patching pushing a fixed model off its data distribution, which training on patched passes addresses directly.

Training for circuit analysis. Closest to our work, several methods add structural constraints during training so that circuits are easier to interpret. Gao et al. (2026) train weight-sparse transformers whose task circuits are substantially smaller and human-readable, Draye et al. (2025) sparsify attention through post-training, and related work uses self-ablation (Ferrao et al., 2025), gradient routing (Cloud et al., 2024), or architectural modifications such as codebooks and bilinear MLPs (Tamkin et al., 2024; Pearce et al., 2025). Other work trains models to implement a known program or causal model (Friedman et al., 2023; Geiger et al., 2022; Gupta et al., 2024), trains replacement models that preserve the network's output distribution (Braun et al., 2024), or trains a hypernetwork that locates causal variables in a fixed model (Sun et al., 2025). In all of these, circuits are still discovered and analyzed separately, post hoc, whereas SDCMs extract circuits inherently by training directly on the circuit-discovery objective.

Self-explaining neural networks. In the *explainability* literature, self-explaining neural networks produce an explanation alongside each prediction rather than relying on post-hoc methods (Alvarez-Melis & Jaakkola, 2018; Ji et al., 2025). Most explain a prediction through its input, via feature attributions (Agarwal et al., 2021; Böhle et al., 2022; You et al., 2025), selected feature subsets or rationales (Chen et al., 2018; Lei et al., 2016; Jain et al., 2020; Bassan et al., 2025), subgraphs of an input graph (Yu et al., 2021; Azzolin et al., 2025), or concepts and prototypes (Koh et al., 2020; Chen et al., 2019). SDCMs are related in spirit, but they expose the subgraph of the model's own computation behind a prediction, with faithfulness defined by patching activations rather than by perturbing inputs; to our knowledge, ours is the first work to pursue a similar direction for circuit discovery.

B Training Specifications

In this appendix, we describe the training data, optimization procedure, and hyperparameter selection, and report the predictive performance of the trained models.

B.1 Models and Training Variants

We use GPT-2 and Qwen2.5-0.5B, the two smallest models in the MIB suite (Mueller et al., 2025). SDCM training jointly updates the language model and circuit predictor and requires two model forward passes per step, placing larger models beyond our compute budget. For each model family, we train three SDCMs, one per task-agnostic patching scheme (resample, zero, and mean), using identical hyperparameters across schemes, alongside a standard-trained model optimized with the language-modeling objective alone. All four models start from the same pretrained checkpoint and share the training data, model optimizer settings, and number of updates.

B.2 Training Data and Sampling

We train GPT-2 on OpenWebText, a public reconstruction of the WebText corpus used for GPT-2's original pretraining. For Qwen2.5-0.5B, we use FineWeb-Edu, a public source of general text, since its original pretraining corpus is unavailable to us. We partition each dataset into consecutive subsets containing 98% of the stored tokenized rows for training, 1% for validation, and 1% for testing. The validation split is used to monitor progress during training.

For each training example, we draw a random 64-token window from a randomly selected stored sequence and sample a prediction position uniformly from its 64 positions. In the unmasked pass, causal attention ensures that the representation at the selected position depends only on the prefix ending there. This gives an effective context length uniformly distributed from 1 to 64 tokens while keeping the input tensor length fixed.

Each prediction position corresponds to a different effective context and can induce a different predicted circuit. Evaluating all positions under their respective circuits would require recomputing the activations under each position's mask, which a single patched forward pass with one shared mask cannot provide. We therefore sample one prediction position per example, generate a mask $m(x)$ from the hidden representation at that position, and apply it throughout the sequence in the patched pass. Both the language-modeling loss $\mathcal{L}_{\mathrm{LM}}$ and the faithfulness loss $\mathcal{L}_{\mathrm{faith}}$ are computed only at the sampled position.

B.3 Training Procedure

At each optimization step t, we use the minibatch form of the augmented-Lagrangian objective $\mathcal{L}_{\mathrm{AL}}$ defined in Section 3.1. Let $\bar{d}_t$ denote the mean of the soft densities $\tilde{d}(x)$ over the global minibatch, and let $d_t^\star$ denote the current target soft density. With the constraint residual $c_t := d_t^\star - \bar{d}_t$, the objective is

$$\mathcal{L}_{\mathrm{AL},t} := \beta\,\overline{\mathcal{L}}_{\mathrm{LM},t} + \gamma\,\overline{\mathcal{L}}_{\mathrm{faith},t} + \lambda_{1,t} c_t + \lambda_{2,t} c_t^2. \tag{6}$$

The barred losses are minibatch averages of the language-modeling loss and the KL faithfulness loss defined in Section 3.1, evaluated at the prediction positions sampled as described in Appendix B.2. The final two terms implement $\mathcal{L}_{\mathrm{dens}}$ using the minibatch estimate of the average soft density.

We optimize this objective using Adam (Kingma & Ba, 2015), with gradient descent on the language-model parameters θ and circuit-predictor parameters ϕ, and gradient ascent on λ_1 and λ_2. The gradients with respect to these two coefficients are c_t and c_t^2, respectively. The patched forward pass uses the binary masks defined in Section 3.1, with gradients propagated through them using the straight-through estimator. Gradients also propagate through the model representation supplied to the predictor. Within the faithfulness loss, the target distribution $p_G(x)$ is treated as a constant. The replacement activations α are also detached from the gradient computation.

The target soft density decreases linearly from 1 to $d^\star$ over the first 6,864 optimization steps and remains fixed thereafter. When the soft density remains above its target, the residual is negative, and repeated ascent updates can drive λ_1 to a large negative value. This multiplier may continue to encourage pruning after the target is reached, causing the density to fall below the target.

To mitigate this overshooting, we use a modified dual-restart rule inspired by Gallego-Posada et al. (2022). We monitor an exponential moving average of the constraint residual. After the smoothed residual reaches $-\epsilon$ or below, its subsequent return to zero or above triggers a reset of λ_1 to zero if λ_1 is negative; λ_2 is left unchanged. The rule is rearmed only after the smoothed residual again reaches

$-\epsilon$ or below. Smoothing and the tolerance threshold reduce sensitivity to minibatch fluctuations. The training objective itself uses the unsmoothed residual c_t.

We initialize the language model from its pretrained checkpoint, set the predictor's affine weights to zero, and set its biases to $\log(0.99/0.01)$. Each edge therefore initially receives a soft retention score of 0.99, so the initial binary mask retains all edges. Both density-control coefficients are initialized to zero. All reported SDCM runs use a single joint-training stage, with separate parameter groups and constant learning rates for the language model, predictor, and density-control coefficients. Learning rates, loss weights, restart settings, and the remaining training hyperparameters are reported in Appendix B.4.

B.4 Hyperparameter Selection and Training Configurations

We use a staged hyperparameter search with resample patching. Selection considers language-modeling performance, measured by predictive accuracy and cross-entropy loss, together with the KL faithfulness loss, favoring configurations that reduce the faithfulness loss while limiting degradation in predictive performance. The search proceeds through the four stages summarized in Table 3.

Table 3: Hyperparameters examined in the staged search.

Stage	Hyperparameters	Values examined
1	Model learning rate and schedule	10^{-5} or 2×10^{-6}; constant or late fivefold reduction
2	Loss weights (γ, β)	$(1, 4)$, $(2, 3)$, $(3, 2)$
3	Density warmup duration	1,000, 3,000, or 6,864 updates
4	Target soft density $d^\star$	2%, 3%, 5%, or 10%

The learning-rate schedule comparison and the 2% density target are evaluated for GPT-2 only. The warmup duration is selected based on the GPT-2 comparisons and used for both model families. The main comparisons in Stages 1-3 use 8,602 optimization steps per run; the density sweep extends training to 9,775 steps.

We select a constant learning-rate schedule, loss weights $\beta = 4$ and $\gamma = 1$, a density warmup of 6,864 steps, and a target soft density of 3% for both model families. The model learning rate is 10^{-5} for GPT-2 and 2×10^{-6} for Qwen2.5-0.5B. The lower learning rate for Qwen2.5-0.5B was chosen because preliminary comparisons showed that 10^{-5} caused greater degradation in its language-modeling performance. Appendix H examines how the training density target affects circuit faithfulness across extraction budgets.

Final training configurations. Within each model family, the selected hyperparameters are shared across resample, zero, and mean training. Table 4 lists the final settings.

Table 4: Final SDCM training settings for both model families.

Setting	GPT-2	Qwen2.5-0.5B
Model learning rate	10^{-5}	2×10^{-6}
Circuit-predictor learning rate	10^{-4}	10^{-4}
Density-control learning rate	10^{-1}	10^{-1}
Language-modeling weight β	4	4
Faithfulness weight γ	1	1
Target soft density $d^\star$	3%	3%
Density warmup steps	6,864	6,864
Optimization steps	9,775	9,775
Global batch size	64	64
Dual-restart tolerance ϵ	0.005	0.005
Residual EMA coefficient	0.995	0.995

All learning rates remain constant. Adam uses moment coefficients $(0.9, 0.999)$ and a numerical stability constant of 10^{-8}, with no weight decay or gradient clipping. We use seed 0 for each re-

ported configuration. Training uses distributed data parallelism across eight GPUs with BF16 mixed precision. Each final run samples 625,600 windows of 64 tokens, with incomplete minibatches retained.

B.5 Capability of the Trained Models

We assess full-model capability without circuit masking through three evaluations: held-out test data from the training corpora, WikiText-2, and the task benchmarks used in our experiments. For each model family, we compare the standard-trained model with SDCMs trained using resample, zero, or mean patching.

Held-out performance on the training corpora. We evaluate GPT-2 on the held-out OpenWebText test split and Qwen2.5-0.5B on the held-out FineWeb-Edu test split, following the partition described in Appendix B.2. These test splits come from the same corpora used for training but were excluded from all training runs reported here. Within each model family, all checkpoints are evaluated on the same 5,000 sampled windows and prediction positions. Each window contains 64 tokens, with one prediction position sampled uniformly, giving an effective context length between 1 and 64 tokens. We report next-token accuracy and perplexity.

Table 5: Full-model performance on held-out test data from the training corpora.

	GPT-2		Qwen2.5-0.5B	
Checkpoint	Perplexity	Accuracy (%)	Perplexity	Accuracy (%)
Standard-trained	36.28	36.10	19.76	41.68
SDCM, resample	36.95	35.70	19.96	41.64
SDCM, zero	36.45	36.02	19.76	41.64
SDCM, mean	36.46	36.38	19.82	41.60

As shown in Table 5, across both model families, SDCM perplexity differs from that of the standard-trained model by at most 1.9%. The largest accuracy decrease is 0.40 percentage points, for GPT-2 trained with resample patching.

WikiText-2 evaluation. We additionally evaluate on the raw WikiText-2 test split (Merity et al., 2017). Non-empty lines are joined with blank-line separators and tokenized using each model's tokenizer without adding special tokens. The resulting stream is divided into consecutive, non-overlapping blocks of 64 or 1,024 tokens, discarding the final incomplete block. Each block is evaluated independently: its first token serves only as context, and all subsequent tokens are prediction targets. We report next-token accuracy and perplexity over all prediction positions. All checkpoints within each model family use identical tokenization, blocks, and scoring.

Table 6: Full-model WikiText-2 performance with block lengths of 64 and 1,024 tokens.

	64-token blocks		1,024-token blocks	
Checkpoint	Perplexity	Accuracy (%)	Perplexity	Accuracy (%)
GPT-2				
Standard-trained	87.68	27.39	30.73	37.92
SDCM, resample	91.51	27.00	32.77	37.02
SDCM, zero	88.92	27.31	31.08	37.81
SDCM, mean	87.85	27.42	31.16	37.76
Qwen2.5-0.5B				
Standard-trained	47.41	34.64	16.13	46.03
SDCM, resample	49.25	34.29	16.52	45.78
SDCM, zero	47.44	34.69	16.18	46.04
SDCM, mean	47.25	34.76	16.16	46.07

Relative to the standard-trained models, the resample-trained SDCMs increase perplexity by 2.4–6.6% across the two model families and block lengths (Table 6). The zero- and mean-trained variants differ from the standard-trained models by at most 1.5%.

Task performance. We use the task datasets and evaluation setup described in Appendix C. For capability evaluation, we use 200 examples for IOI, 200 for Greater-Than, 390 for SVA, and all 50 validation examples for MCQA. IOI and SVA use two-candidate accuracy, comparing the correct answer with the task's incorrect alternative. MCQA uses accuracy among the four answer labels. Greater-Than is scored by the mean difference between the probability mass assigned to valid and invalid year continuations (Hanna et al., 2023).

Table 7: Full-model task performance.

	GPT-2			Qwen2.5-0.5B		
Checkpoint	IOI	GT	SVA	IOI	SVA	MCQA
Standard-trained	0.960	0.796	1.000	0.985	0.987	1.000
SDCM, resample	0.960	0.780	1.000	0.985	0.985	1.000
SDCM, zero	0.965	0.790	1.000	0.990	0.987	1.000
SDCM, mean	0.970	0.796	1.000	0.990	0.995	1.000

Table 7 shows that IOI and SVA accuracy remains within one percentage point of the standard-trained models in both model families. All four Qwen2.5-0.5B models correctly answer all 50 MCQA examples, so this evaluation does not distinguish between them. Greater-Than scores range from 0.780 to 0.796 across the SDCM variants, compared with 0.796 for the standard-trained model.

C Experimental Setup and Evaluation Details

In this appendix, we describe the task datasets, preprocessing, patching protocols, and procedures used for circuit extraction and evaluation.

C.1 Task Datasets and Examples

We evaluate IOI and SVA on both GPT-2 and Qwen2.5-0.5B, Greater-Than on GPT-2, and MCQA on Qwen2.5-0.5B. We describe the task datasets below; sample selection and preprocessing are detailed in Appendix C.2.

Indirect-object identification (IOI). We use the IOI dataset released with MIB (Mueller et al., 2025), following the task studied by Wang et al. (2023). Each prompt introduces two people and then repeats one name as the person giving or offering an object. The task is to complete the sentence with the recipient's name. The dataset varies the names, locations, objects, and sentence templates. For example, the released prompt "After Brad and Louis went to the room, Brad gave a picture frame to" has the correct completion "Louis".

Greater-Than. We use the year-completion task introduced by Hanna et al. (2023), with data generated using the ACDC implementation (Conmy et al., 2023). Each prompt specifies a starting year and the first two digits of an ending year in the same century. The task is to favor two-digit completions that make the ending year greater than the starting year. For example, the prompt "The war lasted from the year 1732 to 17" admits valid completions from "33" through "99". The dataset varies the event noun and starting year, and each prompt can have multiple valid completions.

Subject-verb agreement (SVA). We use the *nounpp* stimuli released by Marks et al. (2025), adapted from Finlayson et al. (2021). In this structure, a prepositional phrase containing a second noun intervenes between the subject noun and the verb. The task is to favor the verb form that agrees with the subject in grammatical number, even when the intervening noun has a different number. For example, the released prompt "The carpenters beside the truck" requires "are" rather than "is": the verb must agree with the plural subject "carpenters", despite the intervening singular noun "truck".

Multiple-choice question answering (MCQA). We use the four-choice version of the CopyColors dataset released with MIB (Mueller et al., 2025). Each prompt explicitly states an object's color

and asks the model to select the corresponding answer label from four options, A through D. For example, the following released prompt has the correct completion "C":

> Question: Coconuts are brown. What color are coconuts?
> A. red
> B. orange
> C. brown
> D. purple
> Answer:

The task therefore tests whether the model can map the stated color to its answer label.

C.2 Data Selection and Preprocessing

We tokenize the full task prompts using each model's tokenizer and evaluate next-token predictions at the end of each prompt. We evaluate Greater-Than only on GPT-2 because Qwen2.5-0.5B tokenizes two-digit year completions into separate tokens, whereas our evaluation requires single-token answers. We evaluate MCQA only on Qwen2.5-0.5B because GPT-2 has low accuracy on this task.

For the main experiments, we use 100 examples per task for circuit extraction, selected after task-specific tokenization checks. Evaluation uses 1,000 examples for IOI and Greater-Than, all 50 validation examples for MCQA, and 390 test examples for SVA after excluding labeled pairs used for extraction. Both model families use the same extraction and evaluation examples for SVA. Within each combination of model and task, the selected examples are shared across checkpoints, extraction methods, patching schemes, metrics, and density budgets.

Excluding matching labeled pairs in SVA does not eliminate all prompt overlap: some prompts occur in both sets with different target verbs. Greater-Than prompts may also overlap between extraction and evaluation because examples are generated with replacement.

C.3 Patching Protocols

We specify how the replacement activations α are constructed for attribution-based circuit extraction and circuit evaluation under the patching schemes defined in Section 2.1. Whenever replacements are obtained from a reference input, we use an unmasked forward pass through the same model checkpoint.

Resample patching. We sample reference inputs from the OpenWebText validation split for GPT-2 and the FineWeb-Edu validation split for Qwen2.5-0.5B, using the data partitions described in Appendix B.2. For each task prompt, we randomly select a stored token sequence and then a contiguous window with the same token length as the prompt. The resulting activations supply replacements at the corresponding token positions. Each prompt's reference input remains fixed across layers and density budgets. Attribution-based extraction and circuit evaluation use separate sampling seeds, 54 and 1,000,054, respectively.

Zero and mean patching. Zero patching replaces each discarded edge's contribution with a zero vector. Mean patching uses the corresponding writer's mean activation at the same absolute token position. For each checkpoint and task, we estimate these means separately from the clean extraction and evaluation examples, using MIB's positional-mean implementation (Mueller et al., 2025). Each set is right-padded to its maximum sequence length, and activations at padding positions are included in the averages. During training, means are instead computed within each local minibatch of fixed-length general-text windows.

Counterfactual patching. For each clean example, a task-specific transformation supplies a paired counterfactual input. Its activations provide the replacements, while the patched forward pass receives the original clean input.

For IOI, the main comparison in Section 4.2 uses the ABC counterfactual: the second occurrence of the subject's name is replaced with a third name absent from the first clause (Mueller et al., 2025). We additionally evaluate all eight IOI counterfactual types listed in Table 8, with results reported in Appendix E.

For Greater-Than, the counterfactual sets the final two digits of the starting year to 01 (Hanna et al., 2023; Conmy et al., 2023).

For SVA, we use the paired prefixes supplied with the released stimuli (Marks et al., 2025). The counterfactual changes the subject's grammatical number, reversing which of the two verb forms agrees with it.

For MCQA, we use MIB's symbol counterfactual, which replaces the answer-choice labels A, B, C, and D with 1, 2, 3, and 4, respectively (Mueller et al., 2025). This changes the required answer token while preserving the question and the order of the answer choices.

C.3.1 IOI COUNTERFACTUALS

We illustrate the eight counterfactuals using the following example from MIB: "After Nick and John spent some time at the car dealership, Nick offered a nail to". The correct completion is "John". Seven counterfactuals combine three transformations: consistently renaming the two people, swapping their order in the first clause, and replacing the later subject with the other person. The ABC counterfactual instead introduces a third person as the later subject, leaving the intended recipient unspecified.

Table 8: IOI counterfactuals. Names are listed in order of appearance; completions correspond to the resulting prompts.

Counterfactual	Name sequence	Completion
Original prompt	Nick, John, Nick	John
ABC	Nick, John, Bob	No unique answer
Renaming	Max, Fred, Max	Fred
First-clause swapping	John, Nick, Nick	John
Subject replacement	Nick, John, John	Nick
Renaming + first-clause swapping	Fred, Max, Max	Fred
Renaming + subject replacement	Max, Fred, Fred	Max
First-clause swapping + subject replacement	John, Nick, John	Nick
All three transformations	Fred, Max, Fred	Max

C.4 CIRCUIT EXTRACTION AND EVALUATION

Baselines. We compare SDCM circuits with EAP-IG-activations (Marks et al., 2025; Hanna et al., 2024) and EAP-IG-inputs (Hanna et al., 2024). EAP-IG-activations accumulates gradients along an interpolation path between clean and replacement activations and serves as our common baseline across patching schemes. EAP-IG-inputs instead interpolates between clean and reference input embeddings. Since zero and mean patching do not specify a reference input, we use EAP-IG-inputs only under resample and counterfactual patching. Both baselines use the official MIB implementations with five interpolation steps, following the MIB evaluation setup (Mueller et al., 2025). For each baseline, circuit extraction and evaluation use the same metric and patching scheme.

Evaluation grids. For resample, zero, and mean patching, we compute AUC using the MIB density grid (Mueller et al., 2025), expressed as percentages of the full graph:

$$\{0.1,\ 0.2,\ 0.5,\ 1,\ 2,\ 5,\ 10,\ 20,\ 50,\ 100\}\%.$$

For counterfactual patching, we use a grid with finer sampling at low densities:

$$\{0.01,\ 0.03,\ 0.05,\ 0.07,\ 0.1,\ 0.15,\ 0.2,\ 0.3,\ 0.4,\ 0.45,\ 0.5,\\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1,\ 1.5,\ 2,\ 3,\ 5,\ 7,\ 10,\ 15,\ 20,\ 50,\ 100\}\%.$$

Both grids are shared across model families and extraction methods.

Faithfulness and AUC. We evaluate normalized KL faithfulness $\mathcal{F}_S(C)$ as defined in Eq. 2, with normalization performed separately for each model. AUC is computed using piecewise-linear interpolation over requested density on a linear scale. Under resample, zero, and mean patching, we integrate over the full evaluated density range using the trapezoidal rule.

Under counterfactual patching, circuits often attain high faithfulness at very low densities. Integration over the full density range can therefore give disproportionate weight to the near-saturation regime, obscuring differences among sparse circuits. We consequently summarize faithfulness over the density region where the standard-trained model's circuits retain room for improvement.

For a fixed model family, task, attribution method, and counterfactual, let $F_M(d)$ denote the piecewise-linear interpolation of circuit faithfulness for model M over requested density d. We define

$$W := \{d \in [d_{\min}, d_{\max}] : 0.2 \le F_{\text{std}}(d) \le 0.9\}, \qquad \text{AUC}_M := \frac{1}{|W|} \int_W F_M(d)\, \mathrm{d}d, \tag{7}$$

where $d_{\min}$ and $d_{\max}$ are the smallest and largest evaluated densities, and $|W|$ is the total width of the selected region.

D Circuit Faithfulness under Task-Agnostic Patching

In this appendix, we extend the experiments in Section 4.1 with GPT-2 results and comparisons between predictor and attribution circuits on the same trained SDCMs.

D.1 GPT-2 Results under Task-Agnostic Patching

We complement Section 4.1 with GPT-2 faithfulness curves in Figure 5 and per-task AUC results in Table 9. Attribution rankings are extracted from the standard-trained model, while predictor circuits are evaluated on their respective SDCMs. Each ranking is reused across resample, zero, and mean patching. The figure follows the plotting conventions of Figure 2, and AUC is computed as described in Appendix C.4.

Predictor circuits generally achieve higher AUC than attribution circuits extracted from the standard-trained model, even when the attribution ranking can be selected separately for each evaluation scheme. The exceptions are Greater-Than under mean patching for all three SDCMs, and SVA under zero patching for the mean-trained SDCM.

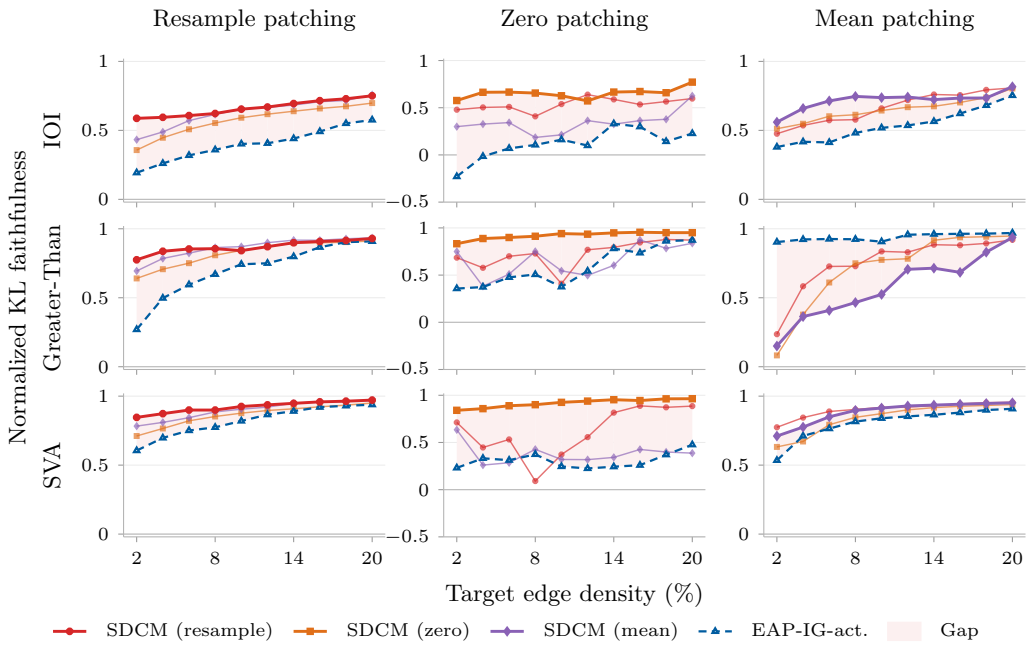


Figure 5: Normalized KL faithfulness on GPT-2, following the plotting conventions of Figure 2.

Table 9: Per-task AUC of normalized KL faithfulness and extraction time on GPT-2, following the conventions of Table 1.

Method	Scheme		IOI				Greater-Than				SVA			
	Ext.	Train	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)
EAP-IG-inp.	Res.	–	72.4	54.9	78.1	14.6	86.3	78.4	93.5	7.6	92.4	57.2	91.8	4.1
EAP-IG-act.	Res.	–	77.5	58.4	84.9		91.1	63.1	96.9		94.0	65.6	92.6	
	Zero	–	59.8	49.7	73.8	493.9	75.7	84.2	92.4	232.3	79.5	72.9	81.5	120.9
	Mean	–	71.4	41.0	83.8		86.1	72.1	**97.3**		89.9	76.9	93.3	
SDCM (ours)	–	Res.	85.3	80.5	87.9		**95.0**	89.1	92.1		**96.9**	88.4	**96.1**	
	–	Zero	81.1	**87.2**	86.8	**1.6**	93.4	**96.6**	91.2	**1.3**	95.2	**96.7**	94.3	**1.3**
	–	Mean	**86.5**	79.1	**90.1**		94.6	89.1	88.9		96.0	74.7	95.6	

D.2 Comparisons within Trained SDCMs

We compare predictor circuits with attribution circuits extracted from the same trained SDCM. Within each training variant, we extract EAP-IG-activations rankings under resample, zero, and mean patching, and an EAP-IG-inputs ranking under resample patching. Each ranking is evaluated under all three schemes on its source model. Tables 10 and 11 report results for GPT-2 and Qwen2.5-0.5B, respectively, holding the model fixed while varying the circuit extraction method.

On Qwen2.5-0.5B, the predictor achieves higher AUC than every attribution baseline in all evaluated settings. On GPT-2, the predictor also achieves the highest AUC in most settings. The main exception is Greater-Than under mean patching, where attribution leads across all three training variants. Two additional exceptions occur for the zero-trained SDCM: IOI under mean patching and SVA under resample patching, with only a marginal difference in the latter.

Table 10: Per-task AUC of normalized KL faithfulness on GPT-2, following the conventions of Table 1, with medals assigned separately within each model group.

Model	Method	Ext.	IOI			Greater-Than			SVA		
			Res. (%)	Zero (%)	Mean (%)	Res. (%)	Zero (%)	Mean (%)	Res. (%)	Zero (%)	Mean (%)
SDCM (Res.)	EAP-IG-inp.	Res.	76.0	71.6	83.5	90.9	85.9	94.9	93.1	79.0	92.4
	EAP-IG-act.	Res.	77.2	57.5	86.2	93.0	84.6	**96.6**	94.9	80.2	93.6
		Zero	65.7	74.4	77.9	81.7	77.6	93.2	84.2	82.6	84.4
		Mean	70.5	65.6	81.4	88.0	83.2	96.5	88.9	76.6	90.5
	Predictor	–	**85.3**	**80.5**	**87.9**	**95.0**	**89.1**	92.1	**96.9**	**88.4**	**96.1**
SDCM (Zero)	EAP-IG-inp.	Res.	77.0	60.5	81.5	90.3	87.2	95.6	94.1	80.6	93.2
	EAP-IG-act.	Res.	78.7	67.6	**88.6**	92.3	90.4	97.1	**95.2**	72.3	93.6
		Zero	66.5	64.3	76.4	77.6	84.4	92.6	85.4	72.4	86.8
		Mean	74.3	68.3	84.1	88.5	89.5	**97.8**	90.3	81.3	93.2
	Predictor	–	**81.1**	**87.2**	86.8	**93.4**	**96.6**	91.2	95.2	**96.7**	**94.3**
SDCM (Mean)	EAP-IG-inp.	Res.	72.4	66.4	77.9	89.5	75.1	94.9	93.2	61.6	92.6
	EAP-IG-act.	Res.	77.9	67.4	86.6	90.5	79.4	96.6	94.5	66.5	93.2
		Zero	62.8	69.4	74.7	80.2	82.3	92.6	84.0	66.6	85.4
		Mean	71.5	54.6	85.9	85.6	86.8	**97.7**	88.4	67.7	92.0
	Predictor	–	**86.5**	**79.1**	**90.1**	**94.6**	**89.1**	88.9	**96.0**	**74.7**	**95.6**

E SDCM Training and Post-Hoc Circuit Discovery

The post-hoc comparisons in Section 4.2 focus on task-specific counterfactual patching. To examine whether the benefits of SDCM training extend across patching schemes and attribution methods, we compare the resample-trained SDCM with the standard-trained model using both EAP-IG-activations and EAP-IG-inputs. The evaluation includes general-text resampling on all evaluated tasks and all eight IOI counterfactuals from MIB (Mueller et al., 2025), described in Appendix C.3.1.

Table 11: Per-task AUC of normalized KL faithfulness on Qwen2.5-0.5B, following the conventions of Table 1, with medals assigned separately within each model group.

Model	Method	Ext.	IOI			SVA			MCQA		
			Res. (%)	Zero (%)	Mean (%)	Res. (%)	Zero (%)	Mean (%)	Res. (%)	Zero (%)	Mean (%)
SDCM (Res.)	EAP-IG-inp.	Res.	82.9	64.1	85.1	94.7	81.3	92.5	82.5	61.6	87.0
	EAP-IG-act.	Res.	81.4	69.1	84.4	94.9	82.1	93.5	83.4	60.0	85.1
		Zero	74.3	55.0	78.8	87.1	42.2	83.5	72.8	56.1	82.2
		Mean	79.0	66.8	82.9	93.3	81.0	92.9	84.9	60.0	89.9
	Predictor	–	**90.5**	**72.6**	**90.3**	**97.5**	**88.7**	**96.5**	**94.2**	**86.5**	**93.9**
SDCM (Zero)	EAP-IG-inp.	Res.	81.2	66.2	84.1	93.5	78.6	91.9	81.5	59.4	86.1
	EAP-IG-act.	Res.	77.9	63.5	80.4	93.7	80.9	92.7	82.2	52.0	86.2
		Zero	72.0	52.6	76.2	76.8	68.8	83.0	71.6	52.9	84.3
		Mean	77.5	66.4	84.6	92.9	82.4	91.7	82.3	59.9	90.9
	Predictor	–	**86.3**	**87.0**	**87.5**	**96.0**	**96.5**	**94.7**	**91.8**	**94.0**	**93.2**
SDCM (Mean)	EAP-IG-inp.	Res.	80.4	62.7	85.8	94.0	73.9	91.3	81.8	54.0	87.3
	EAP-IG-act.	Res.	80.0	58.0	81.3	93.8	77.7	91.9	81.0	49.9	86.9
		Zero	73.3	48.1	78.5	83.1	48.0	83.5	73.1	41.1	80.7
		Mean	80.3	62.3	85.7	93.9	80.2	93.4	85.0	68.2	89.3
	Predictor	–	**90.4**	**72.7**	**91.4**	**97.2**	**86.0**	**96.7**	**93.7**	**86.5**	**94.9**

Tables 12 and 13 report these results together with the task-specific counterfactual comparisons, using the AUC definitions in Appendix C.4. For both methods, the SDCM generally achieves higher AUC. The deficits are typically small, with a larger gap for EAP-IG-inputs on Qwen2.5-0.5B's MCQA task under counterfactual patching.

Table 12: AUC of normalized KL faithfulness for EAP-IG-act., following the conventions of Table 1, with medals assigned within each model family and row.

Task / counterfactual	GPT-2		Qwen2.5-0.5B	
	Standard training (%)	SDCM (%)	Standard training (%)	SDCM (%)
Resample patching				
IOI	**77.5**	77.2	78.4	**81.4**
Greater-Than	91.1	**93.0**	–	–
SVA	94.0	**94.9**	93.9	**94.9**
MCQA	–	–	79.0	**83.4**
Counterfactual patching				
Greater-Than	68.1	**74.2**	–	–
SVA	58.2	**65.2**	**82.9**	82.7
MCQA	–	–	71.7	**84.8**
IOI counterfactuals				
ABC	74.1	**80.3**	73.9	**78.2**
Renaming	79.4	**86.8**	87.4	**90.2**
First-clause swapping	**59.5**	57.5	68.9	**73.0**
Subject replacement	73.5	**76.8**	**76.4**	74.1
Renaming + first-clause swapping	66.7	**66.9**	82.5	**85.5**
Renaming + subject replacement	66.1	**67.8**	81.7	**82.7**
First-clause swapping + subject replacement	86.0	**88.3**	84.1	**86.7**
All three transformations	77.3	**87.7**	87.0	**89.5**

Across the eight IOI counterfactuals, SDCMs achieve higher AUC in 27 of the 32 comparisons spanning the two model families and the two attribution methods (Tables 12 and 13). Under ABC counterfactuals, the gains on GPT-2 are 6.2 percentage points with EAP-IG-activations and 6.4 points with EAP-IG-inputs. The corresponding gains on Qwen2.5-0.5B are 4.3 and 3.0 points,

respectively. The improvement in this setting is therefore reproduced across both model families and both attribution formulations.

The advantage also extends to counterfactuals that combine multiple transformations. When all three transformations are applied, SDCMs achieve higher AUC with both attribution methods in both model families. On GPT-2, for example, AUC with EAP-IG-activations rises from 77.3% on the standard-trained model to 87.7% on the SDCM, a gain of 10.4 percentage points. On Qwen2.5-0.5B, SDCMs achieve higher AUC in seven of the eight IOI settings with each attribution method. For counterfactuals combining renaming and first-clause swapping, the gains are 3.0 percentage points with EAP-IG-activations and 3.2 points with EAP-IG-inputs. In these comparisons, circuits are extracted using the attribution methods without the circuit predictor. These results support the benefit of SDCM training for post-hoc circuit discovery across multiple task-specific counterfactual interventions.

Table 13: AUC of normalized KL faithfulness for EAP-IG-inp., following the conventions of Table 12.

Task / counterfactual	GPT-2		Qwen2.5-0.5B	
	Standard training (%)	SDCM (%)	Standard training (%)	SDCM (%)
Resample patching				
IOI	72.4	**76.0**	81.1	**82.9**
Greater-Than	86.3	**90.9**	–	–
SVA	92.4	**93.1**	93.6	**94.7**
MCQA	–	–	**82.6**	82.5
Counterfactual patching				
Greater-Than	66.5	**72.9**	–	–
SVA	**65.7**	64.6	83.3	**85.2**
MCQA	–	–	**83.7**	78.8
IOI counterfactuals				
ABC	74.6	**81.0**	74.2	**77.2**
Renaming	74.2	**77.2**	86.7	**88.3**
First-clause swapping	69.9	**76.2**	69.5	**71.7**
Subject replacement	70.8	**72.7**	**76.8**	74.8
Renaming + first-clause swapping	**67.1**	66.1	83.3	**86.5**
Renaming + subject replacement	**68.7**	68.0	83.1	**86.6**
First-clause swapping + subject replacement	85.2	**87.2**	85.0	**87.1**
All three transformations	76.0	**78.0**	79.0	**79.9**

F SDCM Circuits on Conventionally Trained Models

In this appendix, we extend the comparison in Section 4.3 to GPT-2, evaluating whether SDCM circuits retain their advantage on the standard-trained model. Circuits from SDCMs trained with resample, zero, or mean patching are applied unchanged to that model and evaluated under all three task-agnostic patching schemes. Table 14 compares their faithfulness with that of circuits extracted directly from the standard-trained model using EAP-IG-activations and EAP-IG-inputs. Evaluation follows the protocol and AUC conventions in Appendix C.4.

SDCM circuits generally achieve higher AUC than attribution circuits extracted directly from the standard-trained model, even though their predictors were never trained on that model. The largest gains occur under zero patching. As in the evaluations on the SDCMs themselves, attribution remains stronger on Greater-Than under mean patching. The remaining exceptions involve only small differences in AUC.

Table 14: AUC of normalized KL faithfulness and extraction time on GPT-2, following the conventions of Table 2.

Method	Scheme		IOI				Greater-Than				SVA			
	Ext.	Train	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)	Res. (%)	Zero (%)	Mean (%)	Time (s)
EAP-IG-inp.	Res.	–	72.4	54.9	78.1	14.6	86.3	78.4	93.5	7.6	92.4	57.2	91.8	4.1
EAP-IG-act.	Res.	–	77.5	58.4	84.9		91.1	63.1	96.9		94.0	65.6	92.6	
	Zero	–	59.8	49.7	73.8	493.9	75.7	84.2	92.4	232.3	79.5	72.9	81.5	120.9
	Mean	–	71.4	41.0	83.8		86.1	72.1	**97.3**		89.9	76.9	93.3	
SDCM (ours)	–	Res.	84.2	76.7	86.1		**94.7**	88.5	91.8		**96.1**	84.5	**95.5**	
	–	Zero	79.7	**85.9**	84.7	**1.6**	92.1	**95.4**	90.8	**1.3**	94.5	**95.5**	93.8	**1.3**
	–	Mean	**84.8**	74.4	**88.7**		93.8	90.8	88.7		95.4	75.4	95.3	

G Runtime and Computational Cost

The computational cost of using SDCMs includes both training and circuit extraction. Since the same trained checkpoint can support multiple extractions across tasks, we report training costs separately from extraction runtimes and compare the latter with those of EAP-IG-activations and EAP-IG-inputs.

Extraction Time Measurement. We measure extraction time over ten separate runs for each model and task, except for MCQA, which uses a single run. Each run uses 100 task examples on a single NVIDIA L40S GPU, with batch sizes of 25 for GPT-2 and 6 for Qwen2.5-0.5B. Extraction with the circuit predictor uses the primary checkpoint trained with resample patching; EAP-IG-inp. and EAP-IG-act. use the corresponding standard-trained model, with KL attribution, resample patching, and five integration steps. Tables 1 and 2 additionally report times for the zero- and mean-trained SDCMs and for EAP-IG-act. extracted under zero and mean patching, measured in the same way; mean-scheme times come from a single run. Methods run sequentially in separate processes, with GPU synchronization immediately before and after each measurement.

Timing covers edge scoring and construction of the scored graph, including score aggregation for the SDCM. Model loading, initial data preparation, circuit selection and pruning at individual density budgets, and faithfulness evaluation are excluded.

Extraction time. Table 15 reports the measured runtimes. The SDCM is faster than both baselines in every evaluated setting, although the speedup varies across tasks.

Table 15: Extraction time in seconds (lower is better). Entries report mean $\pm$ sample standard deviation over ten runs, except for MCQA, which is reported from a single run.

Model	Task	SDCM	EAP-IG-inp.	EAP-IG-act.
GPT-2	IOI	1.61 ± 0.20	14.60 ± 0.65	493.87 ± 25.01
	Greater-Than	1.27 ± 0.22	7.63 ± 0.50	232.26 ± 0.46
	SVA	1.27 ± 0.25	4.06 ± 0.18	120.88 ± 1.40
Qwen2.5-0.5B	IOI	7.07 ± 1.04	38.63 ± 1.29	2477.78 ± 56.50
	SVA	6.11 ± 1.18	15.41 ± 2.58	804.05 ± 84.55
	MCQA	10.13	66.85	4321.54

Training cost. Table 16 reports the cost of the completed final training runs, each using eight NVIDIA L40S GPUs. Elapsed time includes initialization, training, validation, final testing, and checkpoint saving. GPU-hours are computed as elapsed hours multiplied by eight. Training is incurred once per checkpoint, after which the same model and predictor can be reused across tasks; the overall cost therefore depends on the number of subsequent extractions.

Table 16: Cost of each completed final training run. Elapsed time is reported in hours; GPU-hours account for all eight GPUs used by each run.

	GPT-2		Qwen2.5-0.5B	
Configuration	Time (h)	GPU-hours	Time (h)	GPU-hours
Standard training	1.88	15.00	4.36	34.86
SDCM, resample patching	3.82	30.53	10.47	83.76
SDCM, zero patching	3.79	30.35	10.45	83.61
SDCM, mean patching	3.94	31.49	10.56	84.47

H Training Density and Circuit Size

In this appendix, we examine how the training density target affects circuit faithfulness across different evaluation density budgets. The training density target $d^\star$ controls the average soft edge retention during training, while the evaluation budget determines how many edges are selected from the task-level ranking. These quantities serve different purposes: a single trained model supports circuit extraction at multiple evaluation budgets. Nevertheless, the training target may affect the faithfulness of the resulting circuits. We examine this relationship using the density sweep described in Appendix B.4.

We compare training targets of 2%, 3%, 5%, and 10% for GPT-2, and 3%, 5%, and 10% for Qwen2.5-0.5B, holding the remaining training settings fixed within each model family. All checkpoints are trained with resample patching. We analyze the retained sweep evaluations on IOI, Greater-Than, and SVA for GPT-2, and IOI, MCQA, and SVA for Qwen2.5-0.5B. For each checkpoint and task, we extract circuits from the predictor's aggregated ranking $\bar{\ell}$ following Section 4.1 and evaluate them under resample patching. Faithfulness is measured using normalized KL divergence relative to each checkpoint's full and empty models.

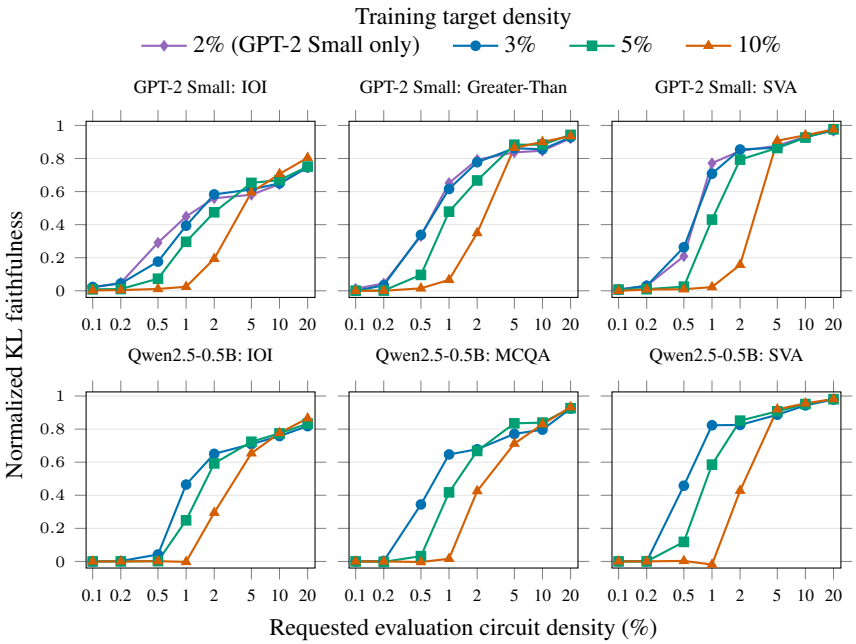


Figure 6: Predictor circuit faithfulness under resample patching for different training density targets. The horizontal axis shows the requested evaluation density, and the vertical axis shows normalized KL faithfulness.

Figure 6 plots the normalized KL faithfulness of predictor circuits against the requested evaluation density, with a separate curve for each training density target. Across all six combinations of model and task, training at 3% yields higher faithfulness than training at 10% at evaluation budgets of both

1% and 2%. The ordering reverses at budgets of 10% and 20%. At a 5% evaluation budget, training at 3% yields higher faithfulness on some tasks, while training at 10% performs better on others. For example, on GPT-2's Greater-Than task, the 3% and 10% training targets yield faithfulness scores of 0.616 and 0.066, respectively, at a 1% evaluation budget. At a 10% evaluation budget, their scores are 0.856 and 0.900. Thus, the substantial advantage of the lower training target for small circuits gives way to a smaller advantage of the higher target at larger budgets.

These results show that the training density target affects circuit faithfulness across evaluation budgets, while the extraction budget can be chosen independently. Lower targets can favor smaller circuits, while higher targets can improve faithfulness at larger budgets. The sweep therefore supports treating $d^\star$ as a choice informed by the intended extraction budgets, without establishing a universally optimal value.

I Additional Case Studies of Extracted Circuits

We complement the SVA example in Figure 4 with a qualitative comparison of IOI circuits on GPT-2. We use EAP-IG-activations under counterfactual patching to extract circuits from the SDCM trained with resample patching and the standard-trained model, and evaluate both circuits on each model. Figure 7 shows the two circuits.

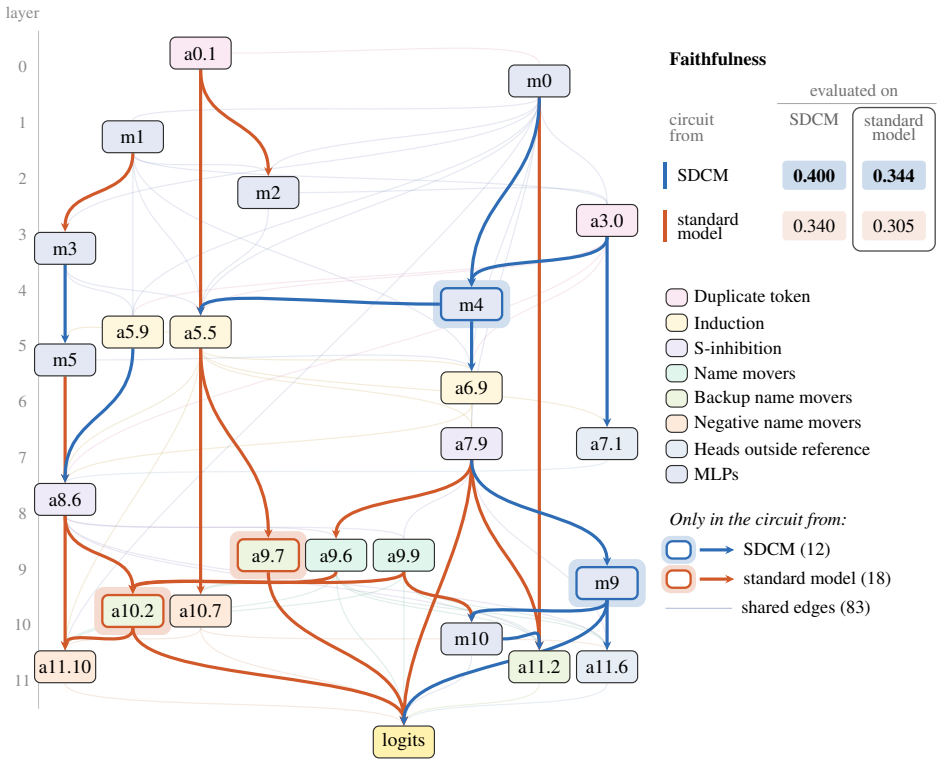


Figure 7: **Post-hoc IOI circuits on GPT-2.** The SDCM circuit is smaller (95 vs. 101 edges) and more faithful on both models (right). Both circuits are extracted with EAP-IG-act. under counterfactual patching. Layers run from top to bottom; the visualization otherwise follows Figure 4, with head colors indicating the reference roles of Wang et al. (2023) and MLPs forming a separate group. Shared edges are faint; nodes whose connections are identical in both circuits are omitted.

The IOI example (Figure 7) reveals how a smaller, more faithful circuit can retain much of the familiar IOI structure while using different connections. The circuits share 83 edges. In the figure, attention heads are grouped according to the reference roles described by Wang et al. (2023), while MLPs are grouped by component type. A visible difference lies in the additional pathways: the SDCM circuit includes routes through MLPs 4 and 9, while the standard-trained circuit includes two

additional backup name movers, heads 9.7 and 10.2. When evaluated on the SDCM, the 95-edge SDCM circuit achieves faithfulness of 0.400, compared with 0.340 for the 101-edge standard-trained circuit. This advantage persists on the standard-trained model, where the corresponding values are 0.344 and 0.305, respectively.