InterpBench: Semi-Synthetic Transformers for Evaluating Mechanistic Interpretability Techniques

Rohan Gupta [1 *] **Iván Arcuschin** [2 *] **Thomas Kwa** [1] **Adrià Garriga-Alonso** [3]

Abstract

Mechanistic interpretability methods aim to identify the algorithm a neural network implements, but it is difficult to validate such methods when the true algorithm is unknown. This work presents InterpBench, a collection of semi-synthetic yet realistic transformers with known circuits for evaluating these techniques. We train these neural networks using a stricter version of Interchange Intervention Training (IIT) which we call Strict IIT (SIIT). Like the original, SIIT trains neural networks by aligning their internal computation with a desired high-level causal model, but it also prevents non-circuit nodes from affecting the model's output. We evaluate SIIT on sparse transformers produced by the Tracr tool and find that SIIT models maintain Tracr's original circuit while being more realistic. SIIT can also train transformers with larger circuits, like Indirect Object Identification (IOI). Finally, we use our benchmark to evaluate existing circuit discovery techniques.

1. Introduction

The field of mechanistic interpretability (MI) aims to reverse-engineer the algorithm implemented by a neural network (Elhage et al., 2021). The current MI paradigm holds that the neural network (NN) represents concepts as *features*, which may have their dedicated subspace (Olah et al., 2020a; Bushnaq et al., 2024) or be in *superposition* with other features (Olah et al., 2020b; Elhage et al., 2022; Engels et al., 2024). The NN arrives at its output by composing many *circuits*, which are subcomponents that implement particular functions on the features (Olah et al., 2020b; Cammarata et al., 2021; Geiger et al., 2023a). To date, the field has been very successful at reverse-engineering toy models on simple tasks (Nanda et al., 2023; Zhong et al., 2023; Chan et al., 2022; Chughtai et al., 2023; Brinkmann et al., 2024). For larger models, researchers have discovered circuits that perform clearly defined subtasks (Wang et al., 2023; Hanna et al., 2024; Heimersheim & Janiak; Lieberum et al., 2023).

How confident can we be that the NNs implement the claimed circuits? The central piece of evidence for many circuit papers is *causal consistency*: if we intervene on the network's internal activations, does the circuit correctly predict changes in the output? There are several competing formalizations of consistency (Chan et al., 2022; Wang et al., 2023; Geiger et al., 2023a; Jenner et al., 2023) and many ways to ablate NNs, each yielding different results (Sanh & Rush, 2021; Conmy et al., 2023; Zhang & Nanda, 2023). This problem is especially dire for *automatic* circuit discovery methods, which search for subgraphs with the highest consistency (Geiger et al., 2023b; Wu et al., 2023) or faithfulness (Conmy et al., 2023; Syed et al., 2023) measurements[1].

These results would be on much firmer ground if we had an agreed-upon protocol for thoroughly checking a hypothesized circuit. To declare a candidate protocol *valid*, we need to check whether, in practice, it correctly distinguishes *true* circuits from false circuits. Unfortunately, we do not know the true circuits of the models we are interested in, so we cannot validate any protocol. Previous work has sidestepped this in two ways. One method is to rely on qualitative evidence (Chan et al., 2022; Olsson et al., 2022), perhaps provided by human-curated circuits (Conmy et al., 2023; Syed et al., 2023), which is expensive and possibly unreliable.

The second way to obtain neural networks with known circuits is to construct them. Tracr (Lindner et al., 2023) is a tool for compiling RASP programs (Weiss et al., 2021) into standard decoder-only transformers. By construction, it outputs a model which implements the specified algorithm, making it suitable for evaluating MI methods. Unfortunately, Tracr-generated transformers are quite different from those

[*] Equal contribution [1] Independent [2] University of Buenos Aires [3] FAR AI. Correspondence to: Rohan Gupta <cybershiptrooper@gmail.com>.

Mechanistic Interpretability Workshop at International Conference on Machine Learning 2024, Vienna, Austria. Copyright 2024 by the author(s).

[1] Faithfulness is a weaker form of consistency: if we ablate every part of the NN that is not part of the circuit, does the NN still perform the task? (Wang et al., 2023; Chan et al., 2022)

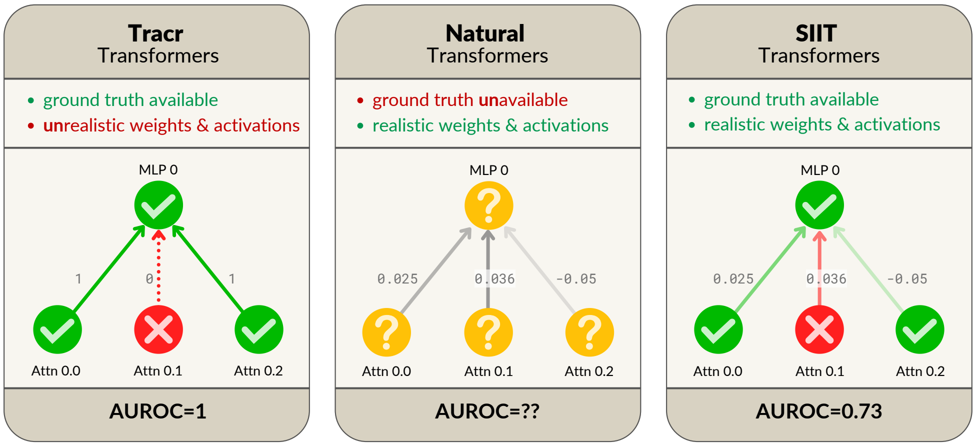

Figure 1: SIIT transformers implement a known ground-truth circuit, but their weights and activations are similar to the ones in naturally trained transformers, letting us measure, in a realistic setting, how accurate circuit discovery methods are at finding the true circuit.

trained using gradient descent: most of their weights and activations are zero, none of their features are in superposition, and they use only a small portion of their activations for the task at hand. Figure 2 shows how different the weights of a Tracr-generated transformer are from those of a transformer trained with gradient descent. This poses a very concrete threat to the validity of any evaluation that uses Tracr-generated transformers as subjects: we cannot tune the inductive biases of circuit evaluation algorithms with such unrealistic neural networks.

1.1. Contributions

In this work, we present INTERPBENCH, a collection of 17 semi-synthetic yet realistic transformers with *known circuits* for evaluating mechanistic interpretability techniques. We collected 16 Tracr circuits plus 1 circuit from the literature (Indirect Object Identification (Wang et al., 2023)), and trained new transformers to implement these circuits using Strict Interchange Intervention Training (SIIT).

SIIT is an extension of Interchange Intervention Training (IIT) (Geiger et al., 2022). Under IIT, we predefine which subcomponents of a *low-level* computational graph (the transformer to train) map to nodes of a *high-level* graph (the circuit). During training, we apply the same interchange interventions (Geiger et al., 2021; Chan et al., 2022) to both the low- and high-level models, and incentivize them to behave similarly with the loss.

Our extension, SIIT, improves upon IIT by also intervening on subcomponents of the low-level model that do not correspond to any high-level component. This prevents the low-level model from using them to compute the output, ensuring the high-level model correctly represents the circuit the NN implements.

We make INTERPBENCH models and the SIIT code used to train them all publicly available.[2]

In summary, the contributions of this article are:

- We present INTERPBENCH, a benchmark of 17 realistic semi-synthetic transformers with known circuits for evaluating mechanistic interpretability techniques.
- We introduce Strict Interchange Intervention Training (SIIT), an extension of IIT which also trains nodes not in the high-level graph. Using systematic ablations, we validate that SIIT correctly generates transformers with known circuits, even when IIT does not.
- We show that SIIT-generated transformers are realistic enough to evaluate MI techniques on, by checking whether circuit discovery methods behave similarly on SIIT-generated and natural transformers.
- We demonstrate the benchmark's usefulness by evaluating five circuit discovery techniques: Automatic Circuit DisCovery (ACDC, Conmy et al., 2023), Subnetwork Probing (SP, Sanh & Rush, 2021) on nodes and edges, Edge Attribution Patching (EAP, Syed et al., 2023), and EAP with integrated gradients (EAP-ig, Marks et al., 2024). On INTERPBENCH, the results conclusively favor ACDC over Node SP, showing that there is enough statistical evidence (p-$value \approx 0.0004$) to tell them apart, whereas the picture in Conmy et al. (2023) was much less clear. Interestingly, the results also show that EAP with integrated gradients is a strong

[2] Code: https://github.com/FlyingPumba/circuits-benchmark (MIT license). Trained networks & labels: https://huggingface.co/cybershiptrooper/InterpBench (CC-BY license).

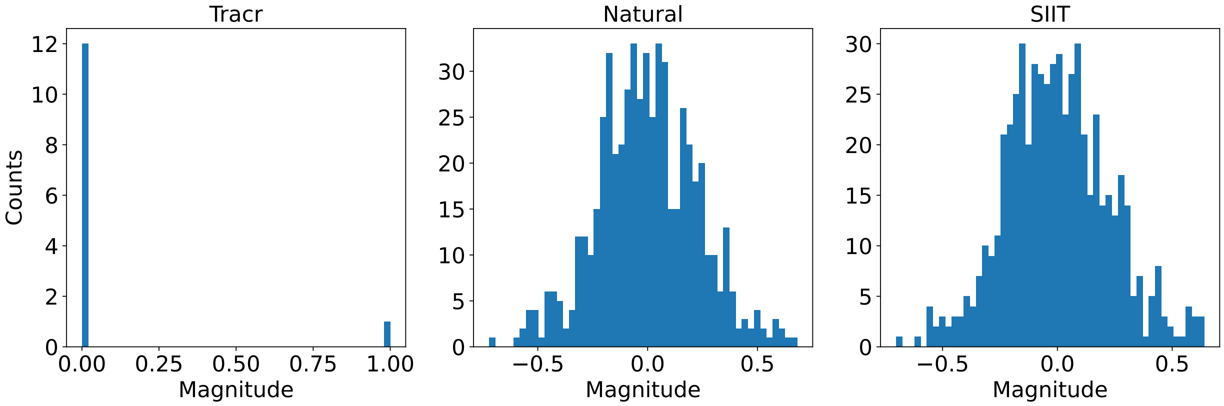

Figure 2: A histogram of the weights for the MLP output matrix in Layer 0 of a Tracr, SIIT and "natural" transformer, i.e. trained by gradient descent to do supervised learning. All these transformers implement the `frac_prevs` task (Lindner et al., 2023). The weight distribution of a SIIT-trained transformer is much closer to the natural than the Tracr transformer. Yet, we know the ground-truth algorithm that the SIIT transformer implements.

contender against ACDC. Regular EAP on the other hand performs poorly, which is understandable given the issues that have been raised about it (Kramár et al., 2024).

2. Related work

Linearly compressed Tracr models. Lindner et al. (2023) compress the residual stream of their Tracr-generated transformers using a linear autoencoder, to make them more realistic. However, this approach does not change the model's structure, and components that are completely zero remain in the final model.

Features in mechanistic interpretability. While this work focuses on circuits, the current MI paradigm also studies *features*: hypothesized natural variables that the NN algorithm operates on. The most popular hypothesis is that features are most of the time inactive, and many features are in *superposition* in a smaller linear subspace (Elhage et al., 2022; Scherlis et al., 2022). This inspired sparse autoencoders (SAEs) as the most popular feature extraction method (Cunningham et al., 2023; Bricken et al., 2023; Templeton et al., 2024; Rajamanoharan et al., 2024; Braun et al., 2024). SAEs produce many features which are qualitatively human-interpretable and are mostly able to reconstruct the residual stream, but this does not imply that they are natural features for the NN . Indeed, some features seem to be circular and do not fit in the superposition paradigm (Engels et al., 2024). Nevertheless circuits on SAE features can be faithful and causally relevant (Marks et al., 2024).

A benchmark which pairs NNs with their known circuits is also a good way to test feature discovery algorithms (like SAEs): the algorithms should naturally recover the values of computational nodes of the true circuit. Conversely, examining how SIIT-trained models represent their circuits' concepts could help us understand how natural NNs represent features. This article omits the comparison because its models only perform one task, and thus have too few features to show superposition.

Other mechanistic interpretability benchmarks. RAVEL (Huang et al., 2024) is a dataset of prompts containing named entities with different attributes that can be independently varied. Its purpose is to develop and evaluate methods which can causally isolate the representations of these attributes in the NN. ORION (Variengien & Winsor, 2023) is a collection of retrieval tasks intended to let MI researchers investigate how large language models (LLMs) follow instructions. FIND (Schwettmann et al., 2023) is a dataset and evaluation protocol for tools which automatically describe model neurons or other components (Bills et al., 2023; Shaham et al., 2024). The test subject must accurately describe a function, based on interactively querying input-output pairs from it.

We see INTERPBENCH as complementary to ORION and RAVEL, and slightly overlapping with FIND. INTERPBENCH is very general in scope: its purpose is to evaluate *any* interpretability methods which discover or evaluate circuits or features. However, it is not suitable for evaluating natural language descriptions of functions like FIND is, and its NNs are about as simple as FIND functions.

3. Strict Interchange Intervention Training

An interchange intervention (Geiger et al., 2020; 2021), or resample ablation (Jenner et al., 2023), returns the output of the model on a *base* input when some of its internal activa-

Algorithm 1 Pseudocode for Strict Interchange Intervention Training (SIIT).

```
Input: High-level and low-level models $\mathcal{M}^H$ and $\mathcal{M}^L$ with variables $\mathcal{V}^H$ and $\mathcal{V}^L$, an alignment $\Pi$ that maps a $V^H \in \mathcal{V}^H$ to a $\mathbf{V}^L \subset \mathcal{V}^L$, low-level model parameters $\theta^L$, learning rate $\ell$, training dataset $\mathcal{D}$
while not converged or we have training budget do
    for $\mathrm{b}, \mathrm{s} \in \mathcal{D} \times \mathcal{D}$ do
        // Calculate IIT loss
        $V^H \sim \mathcal{V}^H$ // Sample a high-level variable
        $\mathbf{V}^L = \Pi(V^H)$ // Aligned low-level variables
        with no grads:
            $\mathrm{o}^H = \textsc{IntInv}(\mathcal{M}^H, \mathrm{b}, \mathrm{s}, V^H)$
        $\mathrm{o}^L = \textsc{IntInv}(\mathcal{M}^L, \mathrm{b}, \mathrm{s}, \mathbf{V}^L)$
        $\mathcal{L}_{IIT} = \textsc{Loss}(\mathrm{o}^H, \mathrm{o}^L) * \mathrm{Weight}_{IIT}$
        $\theta^L \leftarrow \theta^L - \ell \nabla_{\theta^L} \mathcal{L}_{IIT}$

        // Calculate Strictness loss
        $V^L \sim \{V^L \notin \Pi(V^H),\ \forall\, V^H \in \mathcal{V}^H\}$ // Sample a non-aligned low-level variable
        $\mathrm{o}^L = \textsc{IntInv}(\mathcal{M}^L, \mathrm{b}, \mathrm{s}, V^L)$
        $\mathrm{o}^b$ = The correct output for input b
        $\mathcal{L}_{SIIT} = \textsc{Loss}(\mathrm{o}^b, \mathrm{o}^L) * \mathrm{Weight}_{SIIT}$
        $\theta^L \leftarrow \theta^L - \ell \nabla_{\theta^L} \mathcal{L}_{SIIT}$

        // Calculate Behavior loss
        $\mathrm{o}^\emptyset = \mathcal{M}^L(\mathrm{b})$
        $\mathcal{L}_{behavior} = \textsc{Loss}(\mathrm{o}^b, \mathrm{o}^\emptyset) * \mathrm{Weight}_{behavior}$
        $\theta^L \leftarrow \theta^L - \ell \nabla_{\theta^L} \mathcal{L}_{behavior}$
    end for
end while
```

tions have been replaced with activations that correspond to a *source* input. Formally, an interchange intervention $\textsc{IntInv}(\mathcal{M}, \mathit{base}, \mathit{source}, V)$ takes a model $\mathcal{M}$, an input *base*, an input *source*, and a variable V (i.e., a node in the computational graph of the model), and returns the output of the model $\mathcal{M}$ for the input *base*, except that the activations of V are set to the value they would have if the input were *source*. This same definition can be extended to intervene on a set of variables $\mathbf{V}$, where the activations of all variables in $\mathbf{V}$ are replaced. Geiger et al. (2022) define Interchange Intervention loss as:

$$\sum_{\mathrm{b},\mathrm{s} \in \mathrm{dataset}} \textsc{Loss}\big(\textsc{IntInv}(\mathcal{M}^H, \mathrm{b}, \mathrm{s}, V^H), \textsc{IntInv}(\mathcal{M}^L, \mathrm{b}, \mathrm{s}, \Pi(V^H))\big) \tag{1}$$

where $\mathcal{M}^H$ is the high-level model, $\mathcal{M}^L$ is the low-level model, V^H is a high-level variable, $\Pi(V^H)$ is the set of low-level variables that are aligned with (mapped to) V^H, and $\textsc{Loss}$ is some loss function, such as cross-entropy or mean squared error. We use the notation $\mathcal{M}(\mathit{base})$ to denote the output of the model $\mathcal{M}$ when run without interventions on input *base*.

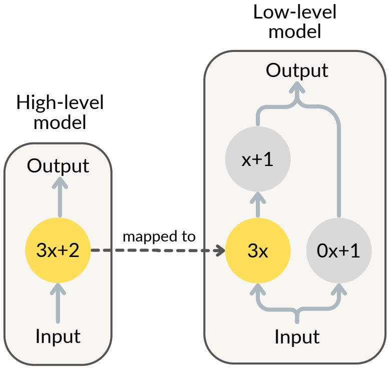

Figure 3: Example of a low-level model that has a perfect accuracy, with aligned low-level nodes (in yellow) that are causally consistent with the high-level model, but has non-aligned nodes (in grey) that affect the output.

The main shortcoming of the above definition is that, by sampling only high-level variables V^H and intervening on the low-level variables that are aligned with it (i.e., $\Pi(V^H)$), IIT never intervenes on low-level nodes that are not aligned with any node in the high-level model. This can lead to scenarios in which the nodes that are not intervened during training end up performing non-trivial computations that affect the low-level model's output, even when the nodes that are aligned with the high-level model are correctly implemented and causally consistent.

As an example, suppose that we have a high-level model $\mathcal{M}^{\mathcal{H}}$ such that $\mathcal{M}^{\mathcal{H}}(x) = 3x + 2$, and we want to train a low-level model $\mathcal{M}^{\mathcal{L}}$ that has three nodes, only one of which is part of the circuit. If we train this low-level model using IIT, we may end up with a scenario like the one depicted in Figure 3. In this example, even though the low-level model has perfect accuracy and the aligned nodes are causally consistent, the non-aligned nodes still affect the output in a non-trivial way. This shows some of the issues that arise when using IIT: aligned low-level nodes may not completely contain the expected high-level computation, and non-aligned low-level nodes may contain part of the high-level computation.

To correct this shortcoming, we propose an extension to IIT called *Strict Interchange Intervention Training* (SIIT). Its pseudocode is shown in Algorithm 1. The main difference between IIT and SIIT is that, in SIIT, we also sample low-level variables that are not aligned with any high-level variable. This allows us to penalize the low-level model for modifying the output when intervened on these non-aligned variables. We implement this modification as a new loss function (*Strictness loss*) that is included in the training loop of SIIT. Formally:

$$\sum_{\mathrm{b},\mathrm{s} \in \mathrm{dataset}} \textsc{Loss}\big(y_b, \textsc{IntInv}(\mathcal{M}^L, \mathrm{b}, \mathrm{s}, V^L)\big) \tag{2}$$

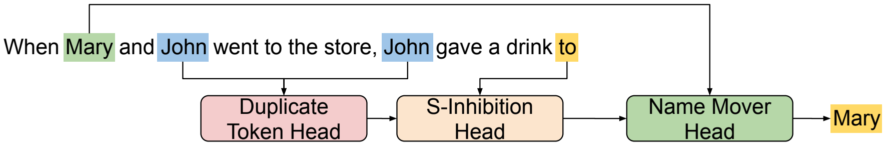

Figure 4: Circuit for Indirect Object Identification task in InterpBench. This circuit is a simplified version of the one manually discovered by Wang et al. (2023). The *Duplicate token head* outputs the first position of duplicated tokens, if there is any; otherwise it outputs -1. The *S-Inhibition head* copies the token from the previous position and outputs it to the *Name mover head*, which increases the logits of all names except the ones that are inhibited.

where y_b is the correct output for input b and V^L is a low-level variable that is not aligned with any high-level variable V^H. In other words, this loss incentivizes the low-level model to avoid performing non-trivial computations for this task on low-level components that are not aligned with any high-level variable. This makes the non-aligned components constant for the inputs in the task distribution, but not necessarily for the ones outside of it. Notice however that under the *Strictness loss* the non-aligned components can still contribute to the output in a constant way, as long as they do not change the output when intervened on. The extent of this effect is analyzed in Appendix A.

As proposed by Geiger et al. (2022), we also include in Algorithm 1 a behavior loss that ensures the model is not overfitting to the IIT and *Strictness* losses. The behavior loss is calculated by running the low-level model without intervening on any components and comparing the output to the correct output.

4. InterpBench

InterpBench is composed of 16 semi-synthetic transformers generated by applying SIIT to Tracr-generated transformers and their corresponding circuits, plus a semi-synthetic transformer trained on GPT-2 and a simplified version of its IOI circuit (Wang et al., 2023). This benchmark can be freely accessed and downloaded from HuggingFace (see Appendix C). We generated the 16 RASP programs using few-shot prompts on GPT-4.

The architecture for the SIIT-generated transformers was made more realistic (compared to the original Tracr ones) by increasing the number of attention heads up to 4 (usually only 1 or 2 in Tracr-generated transformers), which lets us define some heads as not part of the circuit, and by halving the internal dimension of attention heads. The residual stream size on the new transformers is calculated as $d_{\text{head}} \times n_{\text{heads}}$, and the MLP size is calculated as $d_{\text{model}} \times 4$.

Using IIT's terminology, the Tracr-generated transformers are the high-level models, the SIIT-generated transformers are the low-level ones, and the variables are attention heads and MLPs (i.e., the nodes in the transformer's computational graph). Each layer in the high-level model is mapped to the same layer in the low-level model. High-level attention heads are mapped to randomly selected low-level attention heads in the same layer. High-level MLPs are mapped to low-level MLPs in the same layer.

We train all SIIT models by using the Algorithm 1 as described in Section 3, fixing the Weight$_{SIIT}$ to values between 0.4 and 10, depending on the task. Both the Weight$_{IIT}$ and Weight$_{behavior}$ are set to 1. We use Adam as the optimizer for all models, with a fixed learning rate of 0.001, batch size of 512, and Beta coefficients of $(0.9, 0.999)$. All models are trained until they reach 100% Interchange Intervention Accuracy (IIA) and 100% *Strict* Interchange Intervention Accuracy (SIIA) on the validation dataset. IIA, as defined by Geiger et al. (2023b), measures the percentage of times that the low-level model has the same output as the high-level model when both are intervened on the same aligned variables. The *Strict* version of this metric measures the percentage of times that the low-level model's output remains unchanged when intervened on non-aligned variables.

The training dataset is composed of 20k-120k randomly sampled inputs, depending on each task. The validation dataset is randomly sampled to achieve 20% of the training dataset size. The expected output is generated by running the Tracr-generated transformer on each input sequence. The specific loss function to compare the outputs depends on the task: cross-entropy for Tracr categorical tasks, and mean squared error for Tracr regression tasks.

To show that SIIT can also be used to train transformers with non-RASP circuits coded manually, InterpBench includes a transformer trained on a simplified version of the IOI task and the circuit hypothesized by Wang et al. (2023), shown in Figure 4. We train a semi-synthetic transformer with 6 layers and 4 heads per layer, $d_{\text{model}} = 64$, and $d_{\text{head}} = 16$. Each high-level node in the simplified IOI circuit is mapped to an entire layer in the low-level model. We train this transformer using the same algorithm and hyperparameters as for the Tracr-generated transformers, but with a different loss function. We apply the IIT and SIIT losses to the last token of the output sequence, and the cross-entropy loss to all other tokens. The final loss is a weighted average of these losses, with the IIT and SIIT losses upweighted by a factor of 10. All the hyperparemeters remained the same during the experiments.

The semi-synthetic transformers included in INTERPBENCH were trained on a single NVIDIA RTX A6000 GPU. The training time varied depending on the task and the complexity of the circuit but was usually around 1 to 8 hours.

Appendix C explains how to download INTERPBENCH and the license under which it is released. Appendix D contains a detailed description of the tasks included in the benchmark, and Appendix E provides instructions on how to use it.

5. Evaluation

In order to investigate the effectiveness of SIIT and the usefulness of the proposed benchmark, we conducted an evaluation to answer the following research questions (RQs):

RQ1 *(IIT): Do the transformers trained using IIT correctly implement the desired circuits?*

RQ2 *(SIIT): Do the transformers trained using SIIT correctly implement the desired circuits?*

RQ3 *(Realism): Are the transformers trained using SIIT realistic?*

RQ4 *(Benchmark): Are the transformers trained using SIIT useful for benchmarking mechanistic interpretability techniques?*

5.1. Results

RQ1 & RQ2. In this evaluation we compare the semi-synthetic transformers trained using IIT and SIIT. Unless specified, the SIIT models are taken from InterpBench (Section 4). We use the same setup for IIT models, except that we set the Weight_{SIIT} to 0.

To understand if a trained low-level model correctly implements a circuit we need to check that (1) the low-level model has the same output as the high-level model when intervening on aligned variables, and that (2) the non-circuit nodes do not affect the output. As we mentioned in Section 4, all low-level models in our experiments are trained to achieve 100% IIA on the validation sets, which ensures that the first condition is always met.

We answer the second condition by measuring the *node effect* and *normalised KL divergence* after intervening on each node in the model. Node effect measures the percentage of times that the low-level model changes its output when intervened on a specific node. As mentioned before, a node that is not part of the circuit should not affect the output of the model and thus should have a low node effect. Formally, for a node V in a model $\mathcal{M}$, and a pair of inputs (x_b, x_s) with corresponding labels (y_b, y_s), we define the node effect as follows:

$$\text{effect}_V(x_b, x_s, y_b) = \mathbb{1}\left[\textsc{IntInv}(\mathcal{M}, x_b, x_s, V) \neq y_b\right],$$

where $\mathbb{1}[\cdot]$ is the indicator function. The normalized KL divergence is:

$$d_V(x_b, x_s, y_b) = \frac{d_{KL}(\textsc{IntInv}(\mathcal{M}, x_b, x_s, V), y_b) - d_{KL}(\mathcal{M}(x_b), y_b)}{d_{KL}(\mathcal{M}(x_s), y_b) - d_{KL}(\mathcal{M}(x_b), y_b)}.$$

If a semi-synthetic transformer correctly implements a Tracr's circuit, the effect of all aligned nodes will be similar to their corresponding counterparts in the Tracr transformer. For the normalized KL divergence, it is not always possible to have a perfect match with the Tracr-generated transformer, as Tracr does not minimize cross-entropy loss in categorical programs but only fixes the weights so that they output the expected labels. Still, we expect a clear separation between nodes in and out of the circuit.

Figure 5 shows the node effect for nodes in and out of the circuit for 7 randomly sampled tasks in the benchmark, averaged over a test dataset. Each boxplot shows the analysis for a Tracr, IIT or SIIT transformer on a different task. We can see that the boxplots for IIT and Tracr are different, with the IIT ones consistently having high node effect for nodes that are not in the circuit (red boxplots). On the other hand, the SIIT boxplots are more similar to the Tracr ones, with low node effect for nodes that are not in the circuit, and high node effect for nodes that are in the circuit.

Similarly, Figure 6 shows the average normalized KL divergence for nodes in and out of the circuit for 5 randomly sampled categorical tasks in the benchmark. Again, most of the boxplots for IIT have high KL divergence for nodes that are not in the circuit, while the SIIT boxplots have low values for these nodes. We can see that even though the SIIT transformer does not exactly match the Tracr behavior, there is still a clear separation between nodes in the circuit and those not in the circuit, which does not happen for the IIT transformers. It is worth pointing out that the higher error bar across cases for KL divergence is due to the fact that we are optimizing over accuracy instead of matching the expected distribution over labels.

Finally, Figure 7 shows a scatter plot comparing the average node effect for nodes in and out of the circuit for IIT and SIIT transformers for all tasks in the benchmark. We can see that there are several nodes not in the circuit that have a higher node effect for IIT than for SIIT.

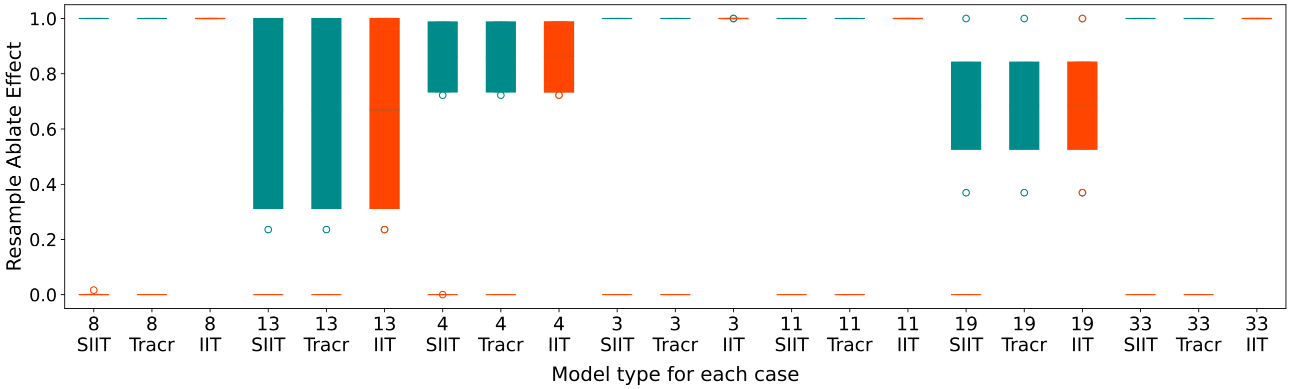

Figure 5: Average effect on accuracy for nodes in the circuit (green) and out of the circuit (red) for the models of 7 randomly sampled tasks in the benchmark. Green boxplots display, for each task and model, the average proportion of model outputs that change when intervening on nodes that belong to the circuit. Red boxplots display the same metric for nodes that are not in the circuit. For all regression tasks, we deem an intervention to have an effect when the new scalar output differs by 0.05 or more from the original. We can see that for Tracr and SIIT models, nodes not in the circuit have much lower effects, but that is not the case for IIT models.

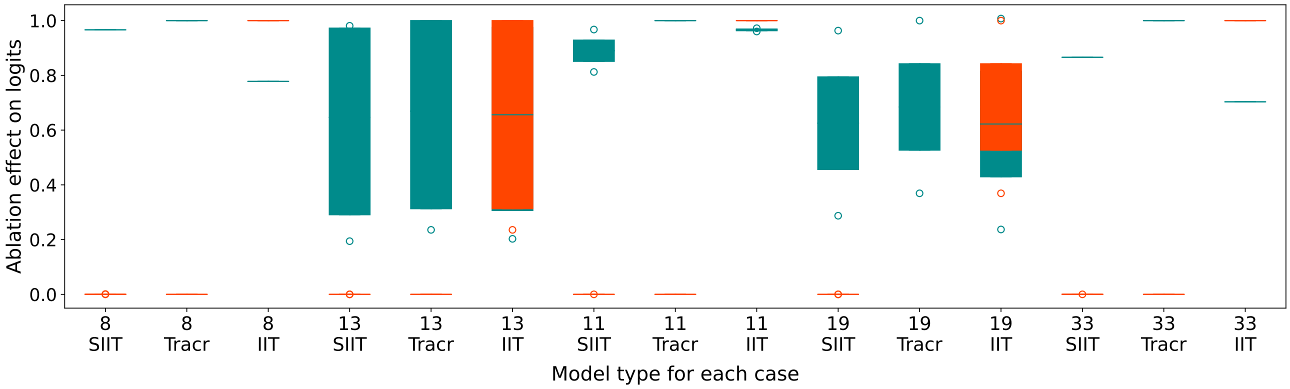

Figure 6: Normalized effect on KL divergence for nodes in the circuit (green) and out of the circuit (red) for the models of 5 randomly sampled categorical tasks in the benchmark. Green boxplots display, for each task, the differences in KL divergence before and after intervening on each node that belongs to the circuit. Red boxplots display the same metric for each node that is not in the circuit. We can see that in Tracr and SIIT nodes are very well separated into in/out of the circuit by their effect size, whereas that is not the case for IIT models.

RQ 1: IIT-generated transformers do not correctly implement the desired circuits: nodes that are not in the circuit affect the output.

RQ 2: SIIT-generated transformers correctly implement the desired circuits: nodes in the circuit have a high effect on the output, while nodes that are not in the circuit do not affect the output.

Appendix A extends Figures 4-6 to all tasks in InterpBench, for SIIT and the original circuit only. It also repeats the experiments but with mean and zero ablations (Zhang & Nanda, 2023). Other types of ablation are a robustness check for InterpBench, which was trained with interchange interventions. Under mean ablations, only nodes in the circuit have an effect, but that is not the case under zero ablations. This may indicate that InterpBench circuits are not entirely true, but also matches the widely held notion that zero ablation is unreliable (Zhang & Nanda, 2023).

RQ3. To analyze the realism of the trained models, we run ACDC (Conmy et al., 2023) on Tracr, SIIT, and "naturally" trained transformers (i.e., using supervised learning). We measure the accuracy of these models after mean-ablating (Zhang & Nanda, 2023) all the nodes rejected by ACDC, i.e. the ones that ACDC deems to not be in the circuit. This lets us check whether SIIT and "natural" models behave similarly from the point of view of circuit discovery techniques. A model that is more realistic should have a score similar to the transformers trained with supervised learning.

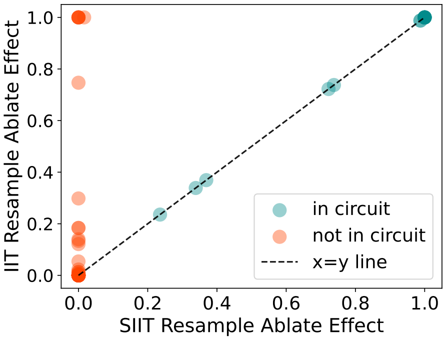

Figure 7: Scatter plot comparing the effect for nodes in the circuit (green) and not in the circuit (red) for IIT and SIIT transformers for all tasks in the benchmark. The *x* and *y* axes display the average node effect when resample ablating on IIT and SIIT models, respectively. For each task, both models have a one-one correspondence for their nodes. We can see that some IIT nodes that are not in the circuit have much higher effects than they should have.

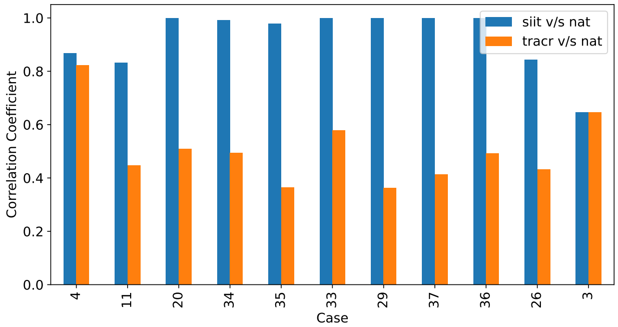

Figure 8: Correlation coefficients between the accuracy achieved by the SIIT and "natural" models, and the Tracr and "natural" models, for 11 randomly selected cases, after mean ablating the nodes rejected by ACDC over different thresholds (see Appendix A). These coefficients are consistently higher when comparing the SIIT and "natural" models than when comparing the Tracr and "natural" models.

Figure 8 displays the difference in correlation coefficients when comparing the accuracy of the SIIT and Tracr models to the "natural" models, showing that SIIT models have a higher correlation with "natural" models than Tracr ones.

Another proxy for realism is: do the weights of "natural" and SIIT models follow similar distributions? Figure 2 shows a histogram of the weights for the MLP output matrix in Layer 0 of a Tracr, SIIT, and "natural" transformer. The SIIT and "natural" weight distributions are very similar.

RQ 3: SIIT-generated transformers are more realistic than Tracr ones, with behavior similar to the transformers trained using supervised learning.

RQ4. To showcase the usefulness of the benchmark, we run ACDC (Conmy et al., 2023), Subnetwork Probing (SP) (Sanh & Rush, 2021), edgewise SP, Edge Attribution Patching (EAP) (Syed et al., 2023), and EAP with integrated gradients (Marks et al., 2024) on the SIIT transformers and compare their performance. Edgewise SP is similar to regular SP, but instead of applying masks over all available nodes, they are applied over all available edges. We compute the Area Under the Curve (AUC) for the edge-level ROC on these algorithms as a measure of their performance.

Figure 9 displays boxplots of the AUC ROCs, and Figure 10 shows the difference in AUC ROC for all circuit discovery techniques against ACDC. For measuring statistical significance, we rely on the well-established Wilcoxon-Mann-Whitney U-test and Vargha-Delaney A_{12} effect size (Arcuri & Briand, 2014). From these tests we get that ACDC is statistically different (*p-value* < 0.05) to all the other algorithms except EAP with integrated gradients, with an effect size A_{12} ranging from 0.54 to 0.91.

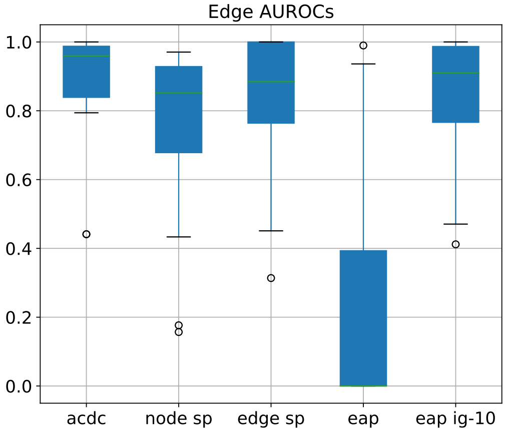

Figure 9: Area under the curve for the ROCs of ACDC, node SP and edge SP on INTERPBENCH. The boxplots for ACDC show the best ROC AUC, obtained by varying the threshold. The boxplots for SP and edgewise SP show the ROC AUC obtained by varying the regularization coefficient, after running for 3000 epochs. EAP with integrated gradients uses 10 samples.

Interestingly, previous evaluations of performance between SP and ACDC on a small number of tasks, including Tracr ones, did not show a significant difference between the two – SP was about as good as ACDC, achieving very similar ROC AUC across tasks when evaluated on manually discovered circuits (Conmy et al., 2023). On the other hand, results on INTERPBENCH clearly show that ACDC outperforms SP on small models that perform algorithmic tasks (*p-value* ≈ 0.0004 and large effect size $\hat{A}_{12} \approx 0.742$).

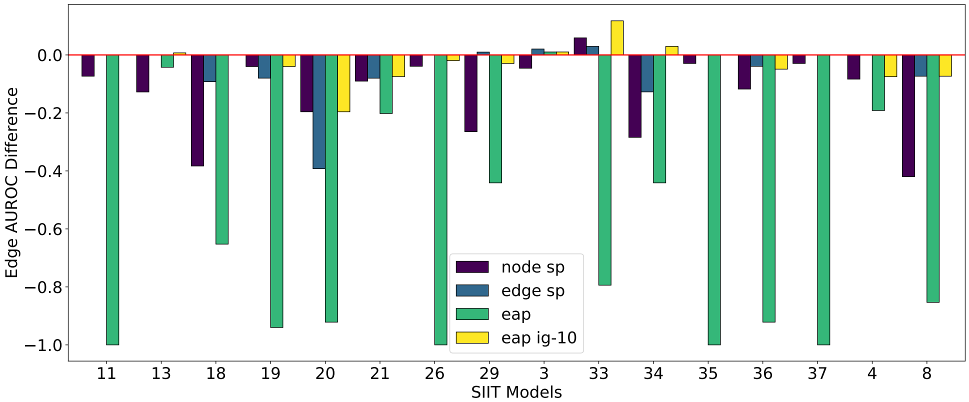

Figure 10: Difference in Edge AUC ROC for all circuit discovery techniques against ACDC, separated by benchmark model.

One important difference between ACDC and other techniques is that this method uses causal interventions to find out which edges are part of the circuit, while SP and EAP rely on the gradients of the model. After manual inspection, we found that the gradients of the SIIT models were very small, possibly due to these models being trained up to 100% IIA and 100% SIIA, which could explain why SP and regular EAP are not as effective as ACDC. This however does not seem to negatively affect EAP with integrated gradients, since the results show that this method is not statistically different from ACDC (*p-value* ≥ 0.05), which means that it is as good as ACDC for the tasks in the benchmark.

It is worth pointing out that there are still some cases where ACDC is not the best technique, as can be seen in Figure 10. Notably, in Case 33, ACDC is outperformed by all the other techniques except EAP. We leave investigating why to future work.

Finally, there is not enough statistical evidence to say EAP with integrated gradients is different than edgewise SP (*p-value* ≥ 0.05), which means that the later is a close third to ACDC and EAP with integrated gradients. Appendix B contains further details on the statistical tests and the evaluation of the circuit discovery techniques.

RQ 4: INTERPBENCH can be used to evaluate mechanistic interpretability techniques, and has yielded unexpected results: ACDC is significantly better than SP and egewise SP, but statistically indistinguishable from EAP with integrated gradients.

6. Conclusion

In this work, we presented INTERPBENCH, a collection of 17 semi-synthetic transformers with known circuits for evaluating mechanistic interpretability techniques. We introduced Strict Interchange Intervention Training (SIIT), an extension of IIT, and checked whether it correctly generates transformers with known circuits. This evaluation showed that SIIT is able to generate semi-synthetic transformers that correctly implement Tracr-generated circuits, whereas IIT fails to do so. Further, we measured the realism of the SIIT transformers and found that they are comparable to "natural" ones trained with supervised learning. Finally, we showed that the benchmark can be used to evaluate existing mechanistic interpretability techniques, showing that ACDC (Conmy et al., 2023) is substantially better at identifying true circuits than node- and edge-based Subnetwork Probing (Sanh & Rush, 2021), but statistically indistinguishable from Edge Attribution Patching with integrated gradients (Marks et al., 2024).

Limitations. INTERPBENCH has proven useful for evaluating circuit discovery methods, but the models it contains, while realistic for their size, are very small. They also contain very little functionality: only one algorithmic circuit per model, as opposed to the many subtasks which next-token prediction encompasses. Therefore, results on INTERPBENCH may not accurately represent the results of the larger models that the MI community is interested in. As an example, we have not evaluated sparse autoencoders in this paper, because we think that the small true number of features and size of the SIIT models will make it impossible to extract meaningful conclusions.

Future work. There are many ways to improve on this benchmark. One is to train SIIT transformers at higher granularities, like subspaces instead of heads, which would allow us to evaluate circuit and feature discovery techniques such as DAS (Geiger et al., 2023b) and Sparse Autoencoders (Cunningham et al., 2023). One could also make the benchmark models more realistic by making each model implement many circuits. This would also let us greatly increase the number of models without manually implementing more tasks.

Societal impacts. If successful, this line of work will accelerate progress in mechanistic interpretability, by putting its results in firmer ground. Better MI makes AIs more predictable and controllable, which makes it easier to use (and misuse) AI. However, it also introduces the possibility of eliminating *unintended* biases and bugs in NNs, so we believe the impact is overall good.

Acknowledgements

RG, IA, and TK were funded by *AI Safety Support Ltd* and *Long-Term Future Fund* (LTFF) research grants. This work was produced as part of the *ML Alignment & Theory Scholars* (MATS) Program – Winter 2023-24 Cohort, with mentorship from Adrià Garriga-Alonso. Compute was generously provided by FAR AI. We thank Niels uit de Bos for his help setting up the Subnetwork Probing algorithm, and Oam Patel for his help in generating the RASP programs used in the benchmark. We also thank Matt Wearden for providing feedback on our manuscript, Juan David Gil for discussions during the research process, and ChengCheng Tan for excellent copyediting.

Author contributions

RG implemented the SIIT algorithm, performed the experiments for the evaluation, and set up the IOI task. IA performed the statistical tests, set up the Tracr tasks, and wrote the initial draft of the manuscript. Both RG and IA helped setting up the circuit discovery techniques. TK provided the initial implementation of IIT. AGA proposed the initial idea for the project, provided feedback and advice throughout the project, and did the final editing of the manuscript.

References

Arcuri, A. and Briand, L. C. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Softw. Test., Verif. Reliab.*, 24(3):219–250, 2014.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 2023.

Braun, D., Taylor, J., Goldowsky-Dill, N., and Sharkey, L. Identifying functionally important features with end-to-end sparse dictionary learning. *CoRR*, 2024. URL http://arxiv.org/abs/2405.12241v2.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Brinkmann, J., Sheshadri, A., Levoso, V., Swoboda, P., and Bartelt, C. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. *CoRR*, 2024. URL http://arxiv.org/abs/2402.11917v2.

Bushnaq, L., Goldowsky-Dill, S. H. N., Braun, D., Mendel, J., Hänni, K., Griffin, A., Stöhler, J., Wache, M., and Hobbhahn, M. The local interaction basis: Identifying computationally-relevant and sparsely interacting features in neural networks. *CoRR*, 2024.

Cammarata, N., Goh, G., Carter, S., Voss, C., Schubert, L., and Olah, C. Curve circuits. *Distill*, 2021. doi: 10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

Chan, L., Garriga-Alonso, A., Goldowsky-Dill, N., Greenblatt, R., Nitishinskaya, J., Radhakrishnan, A., Shlegeris, B., and Thomas, N. Causal scrubbing: a method for rigorously testing interpretability hypotheses. https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing, 2022.

Chughtai, B., Chan, L., and Nanda, N. A toy model of universality: Reverse engineering how networks learn group operations. *CoRR*, 2023.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. In *NeurIPS*, 2023.

Cunningham, H., Ewart, A., Riggs, L., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. *CoRR*, 2023.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Engels, J., Liao, I., Michaud, E. J., Gurnee, W., and Tegmark, M. Not all language model features are linear. *CoRR*, 2024. URL http://arxiv.org/abs/2405.14860v1.

Geiger, A., Richardson, K., and Potts, C. Neural natural language inference models partially embed theories of lexical entailment and negation. In Alishahi, A., Belinkov, Y., Chrupała, G., Hupkes, D., Pinter, Y., and Sajjad, H. (eds.), *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 163–173, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.16. URL https://aclanthology.org/2020.blackboxnlp-1.16.

Geiger, A., Lu, H., Icard, T., and Potts, C. Causal abstractions of neural networks. In *NeurIPS*, pp. 9574–9586, 2021.

Geiger, A., Wu, Z., Lu, H., Rozner, J., Kreiss, E., Icard, T., Goodman, N. D., and Potts, C. Inducing causal structure for interpretable neural networks. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7324–7338. PMLR, 2022.

Geiger, A., Potts, C., and Icard, T. Causal Abstraction for Faithful Model Interpretation. *CoRR*, 2023a.

Geiger, A., Wu, Z., Potts, C., Icard, T., and Goodman, N. D. Finding alignments between interpretable causal variables and distributed neural representations. *CoRR*, 2023b. URL http://arxiv.org/abs/2303.02536v4.

Hanna, M., Liu, O., and Variengien, A. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36, 2024.

Heimersheim, S. and Janiak, J. A circuit for Python docstrings in a 4-layer attention-only transformer. https://www.alignmentforum.org/posts/u6KXXmKFbXfWzoAXn/a-circuit-for-python-docstrings-in-a-4-layer-attention-only.

Huang, J., Wu, Z., Potts, C., Geva, M., and Geiger, A. RAVEL: Evaluating interpretability methods on disentangling language model representations. *CoRR*, 2024. URL http://arxiv.org/abs/2402.17700v1.

Jenner, E., Garriga-Alonso, A., and Zverev, E. A comparison of causal scrubbing, causal abstractions, and related methods. https://www.lesswrong.com/posts/uLMWMeBG3ruoBRhMW/a-comparison-of-causal-scrubbing-causal-abstractions-and, 2023.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing LLM behaviour to components. *CoRR*, abs/2403.00745, 2024.

Lieberum, T., Rahtz, M., Kramár, J., Nanda, N., Irving, G., Shah, R., and Mikulik, V. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla. *CoRR*, 2023. URL http://arxiv.org/abs/2307.09458v3.

Lindner, D., Kramár, J., Farquhar, S., Rahtz, M., McGrath, T., and Mikulik, V. Tracr: Compiled transformers as a laboratory for interpretability. In *NeurIPS*, 2023.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *CoRR*, 2024. URL http://arxiv.org/abs/2403.19647v2.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. An overview of early vision in InceptionV1. *Distill*, 2020a. doi: 10.23915/distill.00024.002. https://distill.pub/2020/circuits/early-vision.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020b. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *CoRR*, 2022. URL http://arxiv.org/abs/2209.11895v1.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *CoRR*, 2024. URL http://arxiv.org/abs/2404.16014v2.

Sanh, V. and Rush, A. M. Low-complexity probing via finding subnetworks. In *NAACL-HLT*, pp. 960–966. Association for Computational Linguistics, 2021.

Scherlis, A., Sachan, K., Jermyn, A. S., Benton, J., and Shlegeris, B. Polysemanticity and capacity in neural networks. *CoRR*, 2022. URL http://arxiv.org/abs/2210.01892v3.

Schwettmann, S., Shaham, T., Materzynska, J., Chowdhury, N., Li, S., Andreas, J., Bau, D., and Torralba, A. FIND: A function description benchmark for evaluating interpretability methods. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 75688–75715. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ef0164c1112f56246224af540857348f-Paper-Datasets_and_Benchmarks.pdf.

Shaham, T. R., Schwettmann, S., Wang, F., Rajaram, A., Hernandez, E., Andreas, J., and Torralba, A. A multimodal automated interpretability agent. *CoRR*, 2024. URL http://arxiv.org/abs/2404.14394v1.

Syed, A., Rager, C., and Conmy, A. Attribution patching outperforms automated circuit discovery. *CoRR*, 2023.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Variengien, A. and Winsor, E. Look before you leap: a universal emergent decomposition of retrieval tasks in language models. *CoRR*, 2023. URL http://arxiv.org/abs/2312.10091v1.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *ICLR*. OpenReview.net, 2023.

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 2021.

Wu, Z., Geiger, A., Icard, T., Potts, C., and Goodman, N. D. Interpretability at scale: Identifying causal mechanisms in alpaca. *CoRR*, 2023. URL http://arxiv.org/abs/2305.08809v3.

Zhang, F. and Nanda, N. Towards best practices of activation patching in language models: Metrics and methods. *CoRR*, 2023. URL http://arxiv.org/abs/2309.16042v2.

Zhong, Z., Liu, Z., Tegmark, M., and Andreas, J. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S5wmbQc1We.

A. Thorough evaluation of dataset models

We provide in Tables 1 to 3 an extended version of the data shown in Figure 5, for all SIIT models in the benchmark, using interchange interventions, mean ablations and zero ablations, respectively. The main takeaway from the interchange intervention and mean ablations is that nodes not in the circuit have zero or very close to zero effect, while nodes in the circuit have a much higher effect. On the other hand, zero ablations indicate that there are nodes not in the circuit with significant effects.

Table 4 shows the accuracy of the SIIT models after mean and zero ablating all the nodes that are not in the circuit. Some of the cases in this table present a big drop in accuracy, specially the regression tasks, while the classification tasks are more robust. This is expected since regression tasks are more sensitive with respect to the output logits, as we compare using an absolute tolerance (*atol*) and do not use the argmax function that is used in classification tasks. We also note that using either mean or zero ablations on many nodes at the same time can easily throw the model's activations off-distribution, which is a common issue also present in models found in the wild.

As a reference, we present in Figure 11 the variation of accuracy for case 3's SIIT model, as a function of the absolute tolerance (*atol*) value for comparing outputs. Most of the logits returned by the SIIT model are at a distance between 0.1 and 0.5 from the original outputs, which is why the accuracy is very low for *atol* values below 0.1, but quickly jumps to 28.9% at 0.1, and then to 84.1% at 0.25.

Furthermore, we also studied the relationship between each node's average activation norm and the Pearson correlation coefficient between the outputs of logit lens applied to that node and the model's actual output. Although many nodes are correlated, most of the ones not in the circuit with a high zero ablation effect have very low variances and norms. For example Case 3 final layer attention hook 3, has an effect 0.42 and norm 1.51 ± 0.55. However, there are still some nodes worth noting, such as the one for final layer's MLP in Case 11, with effect 0.11 and normalised activation norm 1.33 ± 0.55. We leave further investigation of these nodes for future work, as its role is not very well understood at the moment. Interactive plots for this analysis can be found online [3].

We present more detailed information on realism in Figure 12, where we plot the accuracy of the SIIT (trained to 100% SIIA), Tracr and "natural" models for 3 randomly selected cases after mean ablating the nodes rejected by ACDC over different thresholds. These plots show that the SIIT models have a closer behavior to the "natural" models than the Tracr models, which is consistent with the results presented in Section 5. To normalise error from a larger number of edges, we train "natural" and SIIT models with the same architecture of its corresponding Tracr model. We use an identity alignment map to train SIIT models in this case. Figure 13 shows this same information in a more aggregated way, by plotting the average accuracy of the circuit across ACDC thresholds for Tracr, SIIT, and "naturally" trained transformers on all tasks.

B. Evaluation of circuit discovery techniques

In this work we compare the performance of the following circuit discovery techniques: Automated Circuit DisCovery (ACDC), Subnetwork Probing (SP), Edgewise SP, Edge Attribution Patching (EAP), and EAP using integrated gradients (EAP-IG). ACDC traverses the transformer's computational graph in reverse topological order, iteratively assigning scores to edges and pruning them if their score falls below a certain threshold. EAP assigns scores to all edges at the same time by leveraging gradients information, and again prunes edges below a certain threshold to form the final circuit. EAP-IG uses integrated gradients to smooth out the approximation of gradients and improve the performance of EAP. SP learns, via gradient descent, a mask for each node in the circuit to determine if it is part of the circuit or not, and encourages this mask to be sparse by adding a sparseness term to the loss function. The strength of this sparse penalty is controlled by a regularization hyperparameter. Edgewise SP is a variation of SP that learns a mask for each edge in the transformer model instead of each node.

We use different metrics for each task in the benchmark, depending on whether it is a regression or classification task. For ACDC, SP and Edgewise SP, we use the L_2 distance for regression tasks and the Kullback-Leibler divergence for classification tasks. For EAP and EAP-IG, we use the Mean Absolute Error (MAE) for regression tasks and the cross-entropy loss for classification tasks.

Since each of these techniques can be configured to be more or less aggressive, i.e. to prune more or fewer nodes/edges, we

[3] https://wandb.ai/cybershiptrooper/siit_node_stats/reports/Pearson-Correlation-Plots--Vmlldzo4Njg1MDgy

Case	Weight$_{SIIT}$	Nodes in circuit		Nodes not in circuit	
		Quartiles	Range	Quartiles	Range
11	0.4	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
13	0.4	0.31 - 0.67 - 1.00	0.23 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
18	1.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.01
19	0.4	0.53 - 0.69 - 0.84	0.37 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
20	0.4	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
21	0.5	0.13 - 0.14 - 0.36	0.13 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.04
26	0.4	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
29	0.4	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
3	10.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.02	0.00 - 0.09
33	0.4	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
34	1.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
35	1.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
36	1.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
37	1.0	1.00 - 1.00 - 1.00	1.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.00
4	0.4	0.72 - 0.86 - 0.99	0.71 - 0.99	0.00 - 0.00 - 0.00	0.00 - 0.00
8	0.4	0.25 - 0.50 - 0.75	0.00 - 1.00	0.00 - 0.00 - 0.00	0.00 - 0.01
IOI	0.4	0.86 - 0.99 - 1.00	0.48 - 1.0	0.00 - 0.00 - 0.00	0.00 - 0.001

Table 1: Detailed boxplot values for the effect on accuracy of nodes in the circuit and nodes out of the circuit, for all the SIIT models in the benchmark, measured using the node effect equation described in Section 5. We consider that the intervention has changed the output for regression models when the new output differs by 0.05 or more, and for classification models when the new output is simply different from the original output. We can see that nodes not in the circuit have zero or very close to zero effect, while nodes in the circuit have a much higher effect.

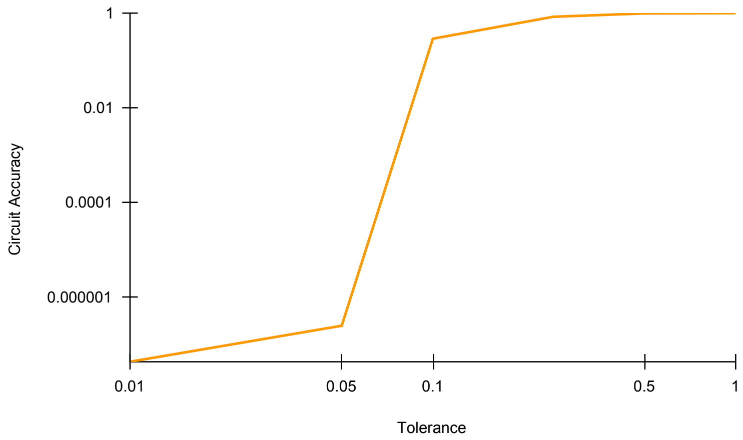

Figure 11: Variation of accuracy for case 3's SIIT model, when mean ablating all the nodes that are not in the ground truth circuit, and varying the absolute tolerance (*atol*) for deciding if an output has changed. For *atol* values below 0.1, the accuracy is very low, close to zero, but it quickly jumps to 28.9% at 0.1. There is a rotund change between 0.1 and 0.25, where the accuracy jumps to 84.1%, and finally, at 0.5, the accuracy reaches 98.9%. This means around 29% of the logits returned by the SIIT model are at a distance closer than 0.1 from the original outputs, 85% are at a distance closer than 0.25, and 99% are at a distance closer than 0.5.

compare their performance using the Area Under the Curve (AUC) of ROC curves. We compute the True Positive Rate (TPR) and False Positive Rate (FPR) for the ROC curves by comparing the discovered circuits with the ground truth circuits, which we have by construction in InterpBench.

Case	Weight$_{SIIT}$	Nodes in circuit		Nodes not in circuit	
		Quartiles	Range	Quartiles	Range
11	0.4	0.54 - 0.55 - 0.56	0.53 - 0.56	0.00 - 0.00 - 0.00	0.00 - 0.00
13	0.4	0.18 - 0.34 - 0.50	0.14 - 0.51	0.00 - 0.00 - 0.00	0.00 - 0.00
18	1.0	0.45 - 0.46 - 0.46	0.45 - 0.47	0.00 - 0.00 - 0.00	0.00 - 0.01
19	0.4	0.27 - 0.31 - 0.35	0.24 - 0.39	0.00 - 0.00 - 0.00	0.00 - 0.00
20	0.4	0.22 - 0.22 - 0.22	0.22 - 0.22	0.00 - 0.00 - 0.00	0.00 - 0.00
21	0.5	0.13 - 0.14 - 0.19	0.11 - 0.31	0.00 - 0.00 - 0.00	0.00 - 0.04
26	0.4	0.57 - 0.57 - 0.57	0.57 - 0.57	0.00 - 0.00 - 0.00	0.00 - 0.00
29	0.4	0.79 - 0.79 - 0.79	0.79 - 0.79	0.00 - 0.00 - 0.00	0.00 - 0.00
3	10.0	0.74 - 0.76 - 0.78	0.71 - 0.80	0.00 - 0.00 - 0.00	0.00 - 0.09
33	0.4	0.56 - 0.56 - 0.56	0.56 - 0.56	0.00 - 0.00 - 0.00	0.00 - 0.00
34	1.0	0.45 - 0.45 - 0.45	0.45 - 0.45	0.00 - 0.00 - 0.00	0.00 - 0.00
35	1.0	0.79 - 0.79 - 0.79	0.79 - 0.79	0.00 - 0.00 - 0.00	0.00 - 0.00
36	1.0	0.31 - 0.31 - 0.31	0.31 - 0.31	0.00 - 0.00 - 0.00	0.00 - 0.00
37	1.0	0.76 - 0.76 - 0.76	0.76 - 0.76	0.00 - 0.00 - 0.00	0.00 - 0.00
4	0.4	0.61 - 0.67 - 0.74	0.61 - 0.76	0.00 - 0.00 - 0.00	0.00 - 0.00
8	0.4	0.20 - 0.39 - 0.59	0.00 - 0.79	0.00 - 0.00 - 0.00	0.00 - 0.01
IOI	0.4	0.59 - 0.79 - 0.94	0.38 - 0.99	0.00 - 0.00 - 0.00	0.00 - 0.00

Table 2: Detailed boxplot values for the effect on accuracy of nodes in the circuit and nodes out of the circuit, for all the SIIT models in the benchmark, measured using *mean ablations*. The mean ablation technique differs from the interchange ablation in that it replaces the activations of the target node with the mean activations for that node in the dataset. In other words, it does not use a different input to replace the activations of the target node. Mean ablation is a robustness check for the SIIT models in InterpBench, which were trained with interchange ablations. We consider that the intervention has changed the output for regression models when the new output differs by 0.05 or more, and for classification models when the new output is simply different from the original output. We can see that nodes not in the circuit have zero or very close to zero effect, while nodes in the circuit have a much higher effect.

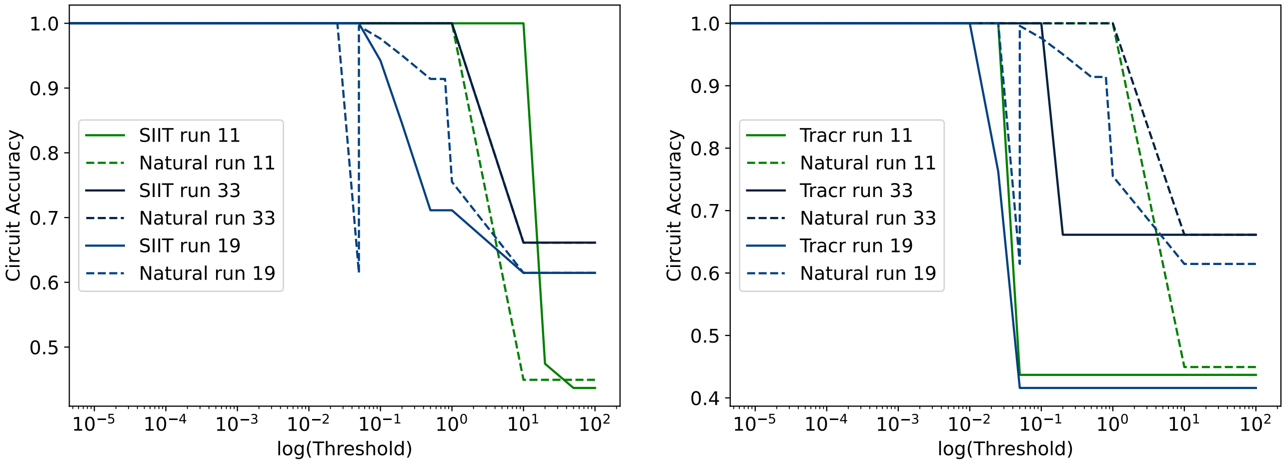

Figure 12: Accuracy of the SIIT, Tracr and "natural" models for 3 randomly selected cases after mean ablating the nodes rejected by ACDC over different thresholds. On the left, we have only SIIT and "natural" models, and on the right, we have only Tracr and "natural" models. The lines in this figure show that the SIIT models have a closer behavior to the "natural" models than the Tracr ones.

In order for this comparison to be sound we need to be more specific on the granularity at which we perform the evaluation. All of the techniques mentioned above work at the QKV granularity level, and thus they consider the outputs of the Q, K,

Case	Weight$_{SIIT}$	Nodes in circuit		Nodes not in circuit	
		Quartiles	Range	Quartiles	Range
3	10110	0.782 - 0.844 - 0.906	0.720 - 0.968	0.000 - 0.000 - 0.000	0.000 - 0.428
4	510	0.874 - 0.934 - 0.977	0.821 - 0.978	0.169 - 0.750 - 0.960	0.000 - 1.000
8	510	0.346 - 0.346 - 0.346	0.346 - 0.346	0.000 - 0.000 - 0.000	0.000 - 0.000
11	510	0.781 - 0.783 - 0.786	0.779 - 0.788	0.000 - 0.000 - 0.000	0.000 - 0.113
13	510	0.245 - 0.471 - 0.705	0.174 - 0.799	0.000 - 0.000 - 0.000	0.000 - 0.000
18	1110	0.091 - 0.256 - 0.440	0.043 - 0.545	0.000 - 0.000 - 0.071	0.000 - 0.112
19	510	0.313 - 0.326 - 0.339	0.301 - 0.351	0.000 - 0.000 - 0.067	0.000 - 0.067
20	510	0.000 - 0.000 - 0.000	0.000 - 0.000	0.000 - 0.000 - 0.000	0.000 - 0.100
21	610	0.121 - 0.146 - 0.155	0.000 - 0.824	0.000 - 0.000 - 0.000	0.000 - 0.107
26	510	0.152 - 0.152 - 0.152	0.152 - 0.152	0.000 - 0.000 - 0.000	0.000 - 0.013
29	510	0.617 - 0.617 - 0.617	0.617 - 0.617	0.000 - 0.000 - 0.000	0.000 - 0.000
33	510	0.300 - 0.300 - 0.300	0.300 - 0.300	0.000 - 0.000 - 0.000	0.000 - 0.000
34	1110	0.436 - 0.436 - 0.436	0.436 - 0.436	0.000 - 0.000 - 0.000	0.000 - 0.000
35	1110	0.493 - 0.493 - 0.493	0.493 - 0.493	0.000 - 0.000 - 0.000	0.000 - 0.000
36	1110	0.290 - 0.290 - 0.290	0.290 - 0.290	0.000 - 0.000 - 0.000	0.000 - 0.000
37	1110	0.541 - 0.541 - 0.541	0.541 - 0.541	0.000 - 0.000 - 0.000	0.000 - 0.000

Table 3: Detailed boxplot values for the effect on accuracy of nodes in the circuit and nodes out of the circuit, for all the SIIT models in the benchmark, measured using *zero ablations*. The zero ablation technique differs from the interchange ablation in that it replaces the activations of the target node with zeros. Zero ablation is a robustness check for the SIIT models in INTERPBENCH, which were trained with interchange ablations, although it is a more aggressive intervention and can potentially throw off the distribution of the model's activations. We consider that the intervention has changed the output for regression models when the new output differs by 0.05 or more, and for classification models when the new output is simply different from the original output. Unlike mean and resample ablations, where we see little to no effects from nodes that are not in the circuit, significant effects can be seen when using zero ablations.

and V matrices in attention heads and the output of MLP components as nodes in the computational graph. On the other hand, SIIT models are trained at the attention head level, without putting a constraint on the head subcomponents, which means that the trained models can solve the required tasks via QK circuits, OV circuits, or a combination of both (Elhage et al., 2021). Thus, during the evaluation of the circuit discovery techniques, we promote the QKV nodes to heads on both the discovered circuits and the ground truth circuits. In other words, if for example the output of a Q matrix in an attention head is part of the circuit, we consider the whole attention head to be part of it as well.

Additionally, when calculating the edge ROC curves for SP, we consider an edge to be part of the circuit if both of its nodes are part of the circuit. This is a simplification, but it allows us to compare regular SP with the rest of the techniques, which work at the edge level.

Table 5 shows all the p-values for the Wilcoxon-Mann-Whitney U-test on each pair of circuit discovery techniques, for the comparison of the AUC of ROC curves. Table 6 shows the Vargha-Delaney $\hat{A}_{12}$ effect size values for the same comparison.

C. Benchmark and license details

The code repository for our benchmark can be found here: `https://github.com/FlyingPumba/circuits-benchmark`, and it is licensed under the MIT license. The trained models can be found here: `https://huggingface.co/cybershiptrooper/InterpBench`, and they are licensed under CC-BY. The benchmark's code is hosted on GitHub and the trained models are hosted on HuggingFace. We will ensure that both are available for a long time. For that purpose, we have minted DOIs for both the code repository and the trained models. The DOI for the code repository is 10.5281/zenodo.11518575 and the DOI for the trained models is 10.57967/hf/2451.

The intended use of this benchmark is to evaluate the effectiveness of mechanistic interpretability techniques. The training and evaluation procedures can be found in our code repository and are described in Sections 4 and 5. The code repository

Case	Task type	Mean ablation accuracy	Zero ablation accuracy
3	Regression	0.0	0.131
4	Regression	0.525	0.248
8	Classification	0.632	0.634
11	Classification	0.967	0.887
13	Classification	0.959	0.943
18	Classification	0.949	0.913
19	Classification	0.829	0.527
20	Classification	1.0	0.995
21	Classification	0.889	0.544
26	Classification	0.641	0.641
29	Classification	0.741	0.891
33	Classification	0.913	0.9
34	Classification	0.805	0.784
35	Classification	0.915	0.989
36	Classification	1.0	1.0
37	Classification	0.837	0.548

Table 4: Accuracy of the SIIT models after mean and zero ablating all the nodes that are not in the ground truth circuit. We consider that the ablation has changed the output for regression models when the new output differs by 0.05 or more, and for classification models when the new output is simply different from the original output. We can see that there is a big drop in accuracy for models performing regression tasks, while the models performing classification tasks are more robust. It is worth noting that using both mean and zero ablations on many nodes at the same time can be a very aggressive intervention and throw off the distribution of the model's activations. We expect realistic models to face similar issues.

	ACDC	Node SP	Edge SP	EAP	EAP-IG
ACDC	-	**0.000427**	**0.028417**	**0.000061**	0.099481
Node SP	-	-	**0.015503**	**0.000153**	**0.000979**
Edge SP	-	-	-	**0.000031**	0.307821
EAP	-	-	-	-	**0.000648**

Table 5: Wilcoxon-Mann-Whitney U-test p-values for the comparison of the AUC of ROC curves for the different circuit discovery techniques. We use $\alpha = 0.05$ as the significance level. The p-values below this level are marked in bold, which means that we can reject the null hypothesis that the two techniques being compared have the same distribution of AUC values. I.e., we can say that the AUC values are significantly different.

	ACDC	Node SP	Edge SP	EAP	EAP-IG
ACDC	-	0.742	0.541	0.91	0.555
Node SP	-	-	0.355	0.844	0.316
Edge SP	-	-	-	0.887	0.486
EAP	-	-	-	-	0.111

Table 6: Vargha-Delaney $\hat{A}_{12}$ effect size values for the comparison of the AUC of ROC curves for the different circuit discovery techniques. The values are interpreted as follows: $0.56 < \hat{A}_{12} < 0.64$ is considered small, $0.64 < \hat{A}_{12} < 0.71$ is considered medium, and $\hat{A}_{12} > 0.71$ is considered large.

also contains instructions on how to replicate the empirical results presented in this work. The benchmark we provide does not contain any offensive content. We, the authors, bear all responsibility to withdraw our paper and data in case of violation of licensing or privacy rights.

We provide several structured metadata files for our benchmark, all available in HuggingFace's repository:

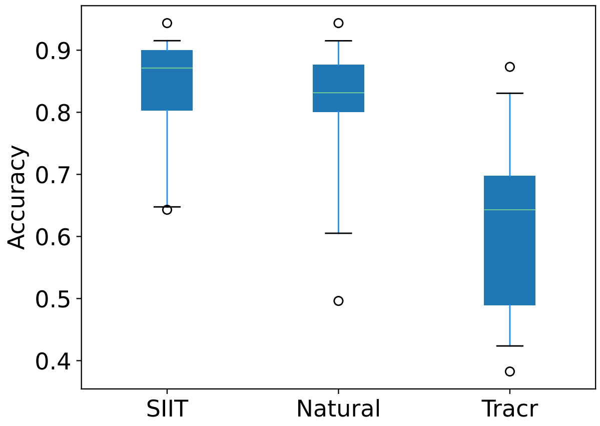

Figure 13: Average accuracy of circuit across ACDC thresholds, for Tracr, SIIT, and "naturally" trained transformers on all tasks. The scores in each boxplot show the accuracy of models after mean-ablating all the nodes that are not a part of ACDC's hypothesis, averaged across multiple thresholds, for each task. SIIT and Natural scores are clearly the most similar.

- A Croissant metadata record.
- A CSV file listing the metadata for all cases in the benchmark.
- A Parquet file listing the metadata for all cases in the benchmark.
- A JSON file listing the metadata for all cases in the benchmark.

D. Tasks in the benchmark

Table 7 displays all the tasks included in InterpBench.

E. Benchmark usage

The trained models hosted on HuggingFace are organized in directories, each one corresponding to a case in the benchmark, containing the following files:

- `ll_model.pth`: A serialized PyTorch state dictionary for the trained transformer model.
- `ll_model_cfg.pkl`: Pickle file containing the architecture config for the trained transformer model.
- `meta.json`: JSON file with hyperparameters used for training for the model.
- `edges.pkl`: Pickle file containing labels for the circuit, i.e., list of all the edges that are a part of the ground truth circuit.

These models can be loaded using TransformerLens, a popular Python library for Mechanistic Interpretability on transformers:

```
import pickle
from transformer_lens import HookedTransformer

cfg_dict = pickle.load(f"ll_model_cfg.pkl")
cfg = HookedTransformerConfig.from_dict(cfg_dict)

model = HookedTransformer(cfg)
```

Case	Tracr?	Task type	Description
11	Yes	Classification	Counts the number of words in a sequence based on their length.
13	Yes	Classification	Analyzes the trend (increasing, decreasing, constant) of numeric tokens.
18	Yes	Classification	Classify each token based on its frequency as 'rare', 'common', or 'frequent'.
19	Yes	Classification	Removes consecutive duplicate tokens from a sequence.
20	Yes	Classification	Detect spam messages based on appearance of spam keywords.
21	Yes	Classification	Extract unique tokens from a string.
26	Yes	Classification	Creates a cascading effect by repeating each token in sequence incrementally.
29	Yes	Classification	Creates abbreviations for each token in the sequence.
3	Yes	Regression	Returns the fraction of 'x' in the input up to the i-th position for all i.
33	Yes	Classification	Checks if each token's length is odd or even.
34	Yes	Classification	Calculate the ratio of vowels to consonants in each word.
35	Yes	Classification	Alternates capitalization of each character in words.
36	Yes	Classification	Classifies each token as 'positive', 'negative', or 'neutral' based on emojis.
37	Yes	Classification	Reverses each word in the sequence except for specified exclusions.
4	Yes	Regression	Return fraction of previous open tokens minus the fraction of close tokens.
8	Yes	Classification	Fills gaps between tokens with a specified filler.
IOI	No	Classification	Indirect Object Identification.
IOI Next token	No	Classification	Indirect Object Identification, trained also with next-token prediction.

Table 7: A description of the tasks included in InterpBench.

```
weights = torch.load(f"ll_model.pth")
model.load_state_dict(weights)
```

More details of usage are provided in the GitHub repository.