CIRCUIT COMPOSITIONS: EXPLORING MODULAR STRUCTURES IN TRANSFORMER-BASED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A fundamental question in interpretability research is to what extent neural networks, particularly language models, implement reusable functions via subnetworks that can be composed to perform more complex tasks. Recent developments in mechanistic interpretability have made progress in identifying subnetworks, often referred to as circuits, which represent the minimal computational subgraph responsible for a model's behavior on specific tasks. However, most studies focus on identifying circuits for individual tasks without investigating how functionally similar circuits relate to each other. To address this gap, we examine the modularity of neural networks by analyzing circuits for highly compositional subtasks within a transformer-based language model. Specifically, given a probabilistic context-free grammar, we identify and compare circuits responsible for ten modular string-edit operations. Our results indicate that functionally similar circuits exhibit both notable node overlap and cross-task faithfulness. Moreover, we demonstrate that the circuits identified can be reused and combined through subnetwork set operations to represent more complex functional capabilities of the model.

1 INTRODUCTION

Neural networks can be effectively modeled as causal graphs that illustrate how inputs are mapped to the output space (Mueller et al., 2024). For instance, the feed-forward and attention modules within the Transformer architecture (Vaswani et al., 2017) can be interpreted as a series of causal nodes that guide the transformation from input to output via the residual stream (Ferrando et al., 2024). This abstraction is commonly used in mechanistic interpretability to identify computational subgraphs, or *circuits*, responsible for the network's behavior on specific tasks (Wang et al., 2023). Circuits are typically identified through causal mediation analysis, which quantifies the causal influence of model components on the network's predictions (Mueller et al., 2024). Techniques such as activation patching (Meng et al., 2022), attribution patching (Nanda, 2023; Syed et al., 2023), and their variants (Hanna et al., 2024b) have been successfully applied to identify circuits in language models for tasks such as indirect object recognition (Wang et al., 2023; Merullo et al., 2024), entity tracking (Prakash et al., 2024), and factual recall (Meng et al., 2022).

However, a notable limitation of existing studies is their focus on identifying circuits for isolated, individual tasks. Few studies compare circuits responsible for different functional behaviors of the model, and those that do primarily focus on tasks with limited cross-functional similarity (Hanna et al., 2024b). This limitation may be due to the rigid task structures imposed by common circuit identification methods, which typically analyze the causal effect of individual model components on predictions related to a *specific token* of interest (Meng et al., 2022; Nanda, 2023; Syed et al., 2023). For instance, Meng et al. (2022) identify circuits responsible for recalling factual associations, such as "The Eiffel Tower is located in", by analyzing the causal effect of model components on predicting a single token representing the association, e.g., "Paris". As a result, more complex functional capacities, which are difficult to capture through the causal effect on a single token prediction, are often overlooked.

In this study, we explore the modularity of neural networks by comparing circuits responsible for highly compositional subtasks within a transformer-based sequence-to-sequence model. Specifically, we identify circuits associated with ten modular string-edit operations on a probabilistic context-free grammar, introduced by PCFG SET (Hupkes et al., 2020). We analyze the circuits discovered in terms of both node overlap and cross-task faithfulness, assessing their performance for functionally related tasks. To facilitate the study of circuits related to sequence-prediction tasks that extend beyond single-token predictions, we propose an automatic circuit identification method that builds upon continuous sparsification (Savarese et al., 2020), jointly optimizing for *faithfulness* and *minimality*—two key objectives in circuit discovery (for further details, see Section 2.2). Moreover, we demonstrate that the circuits identified can be reused and combined through subnetwork set operations to explain more complex functional capabilities of the model.

In summary, our contributions are as follows: *1)* we demonstrate the application of continuous sparsification to automatically discover both faithful and minimal circuits for sequence-to-sequence tasks from PCFG SET; *2)* we analyze the relationships between functionally related circuits by examining node overlap and cross-task faithfulness, providing insights into the model's modular structure; and *3)* we show that computational subgraphs from functionally related circuits can be combined using subnetwork set operations, resulting in novel subnetworks that explain model behavior on tasks beyond the scope of the initial circuits.

2 BACKGROUND

In this section, we provide a brief overview of key concepts related to circuit discovery and continuous sparsification. For a more comprehensive perspective of interpretability research rooted in causal mediation analysis, including the circuits framework, we direct interested readers to the work of Mueller et al. (2024).

2.1 ACTIVATION PATCHING

A widely used approach for identifying circuits within neural networks, particularly language models, is activation patching (Vig et al., 2020; Meng et al., 2022; Wang et al., 2023; Zhang & Nanda, 2024). Grounded in causal mediation analysis, activation patching assesses the causal influence of nodes within the model's computation graph by quantifying their *indirect effect* (IE) (Pearl, 2001) on downstream nodes, typically the model's final output (Mueller et al., 2024). Given a counterfactual intervention $\tilde{\boldsymbol{z}}$ applied to a mediator $\boldsymbol{z}$—commonly another node within the model's computation graph—, the indirect effect of a node $\boldsymbol{x}$ on some downstream node $\boldsymbol{y}$ through $\boldsymbol{z}$ can be calculated as the difference in a metric $\mathbb{P}$ that captures the state of $\boldsymbol{y}$ before and after the intervention:

$$\text{IE}\left(\mathbb{P}, \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{z}_x, \tilde{\boldsymbol{z}}\right) = \mathbb{P}\left(\boldsymbol{y}(\boldsymbol{x}) \middle| \, \boldsymbol{z} = \boldsymbol{z}_x\right) - \mathbb{P}\left(\boldsymbol{y}(\boldsymbol{x}) \middle| \, \text{do}\left(\boldsymbol{z} = \tilde{\boldsymbol{z}}\right)\right) \tag{1}$$

In this context, $\boldsymbol{x}$ typically represents the model's input for the task at hand, while $\boldsymbol{y}$ denotes the model's output. The variable $\boldsymbol{z}_x$ corresponds to the mediator's natural value for $\boldsymbol{x}$ without intervention, and $\tilde{\boldsymbol{z}}$ represents its counterfactual value. Various types of counterfactual interventions are explored in the literature. For example, zero and mean ablations modify the mediator's value by setting it to either zero or its mean value across a reference distribution, respectively (Wang et al., 2023). Another approach, *symmetric token replacement*, involves substituting the mediator's value with the value obtained when the model is exposed to a slightly altered input, where key tokens related to the task are replaced with similar tokens of the same sequence length (Zhang & Nanda, 2024). For instance, when exploring model components responsible for predicting factual information, Meng et al. (2022) use the mediator's value with respect to a *clean* input like "The Eiffel Tower is in" as the counterfactual intervention $\tilde{\boldsymbol{z}}$, while assessing its indirect effect on the model output $\boldsymbol{y}$ over a similar but slightly *corrupted* input $\boldsymbol{x}$, such as "The Colosseum is in," for which the mediator takes the value $\boldsymbol{z}_x$.

In its original form, activation patching iteratively assesses the indirect effect across a set of mediators, assigning causal significance to a mediator's influence on the model's task behavior when the observed indirect effect exceeds a defined threshold (Vig et al., 2020; Meng et al., 2022; Wang et al., 2023). The choice of mediator $\boldsymbol{z}$ varies between studies, ranging from the entire output activation vector $\boldsymbol{a} \in \mathbb{R}^d$ of a module (e.g., the multi-head attention module) or a submodule (e.g., the linear layer within a module), to individual neurons $a_i \in \boldsymbol{a}$ (Mueller et al., 2024). Similarly, the intervention's downstream targets can be configured. For example, node patching directs the intervention to the residual stream, while edge patching affects the input to a specific model component (Miller et al., 2024). Overall, a high level of granularity with respect to the intervention can lead to a combinatorial explosion of the search space when exhaustively exploring all mediators (Mueller et al., 2024). To address this, several variations of activation patching have been developed—most notably, attribution patching and its derivatives (Nanda, 2023; Syed et al., 2023; Hanna et al., 2024b)—which balance accuracy and causal guarantees with improved search efficiency. In this study, we overcome the combinatorial problem by optimizing over a continuous approximation of the discrete search space (see Section 3 for further details).

2.2 CIRCUIT PROPERTIES

Once a circuit has been successfully identified for a given model $\mathcal{M}$ and task T, it can be represented as a binary mask that signifies whether a specific model component is causally relevant to the model's behavior for the task at hand. Let $\boldsymbol{z}_i \in \mathcal{Z}$ represent a single node[1] in the model's causal graph, denoted as $\mathcal{G} = (\mathcal{Z}, \mathcal{E})$, where $\mathcal{E} \subseteq \mathcal{Z} \times \mathcal{Z}$ and $|\mathcal{Z}| = N$.

[1]Depending on the granularity of the identification method, this can range from entire modules to individual neurons within the model.

The circuit can be defined as the binary mask $\boldsymbol{m} \in \{0,1\}^N$ over the model's node space, where each entry specifies whether a node $\boldsymbol{z}_i$ is causally relevant to the model's task behavior ($m_i = 1$) or not ($m_i = 0$).

Circuits are generally evaluated based on three key criteria (Wang et al., 2023):

1. *Faithfulness*: A circuit is considered faithful to the task T if it accurately captures the full model's task behavior while ablating all nodes not identified as causally relevant ($m_i = 0$) by replacing them with some ablation value $\tilde{\boldsymbol{z}}$ (Hanna et al., 2024b; Miller et al., 2024). Given a metric P that compares the outputs of two models for task T, task faithfulness F_T is typically quantified as:
$$F_T = P\left(\mathcal{M}\left(\cdot\right), \mathcal{M}\left(\cdot \mid \text{do}\left(\boldsymbol{z} = \boldsymbol{m} \odot \boldsymbol{z}_x + (\mathbf{1} - \boldsymbol{m}) \odot \tilde{\boldsymbol{z}}\right)\right)\right) \tag{2}$$
where essentially, the causally relevant nodes identified by the circuit ($m_i = 1$) are preserved, while the remaining nodes ($m_i = 0$) are ablated via element-wise multiplication of $(\mathbf{1} - \boldsymbol{m})$ with $\tilde{\boldsymbol{z}}$.
2. *Minimality*: A circuit is deemed minimal if it avoids falsely identifying causally irrelevant nodes as causally relevant (Mueller et al., 2024). Formally, given a candidate set of circuits C, minimality is encouraged by selecting the circuit with the smallest norm: $\min_{\boldsymbol{m} \in C} \|\boldsymbol{m}\|_1$.
3. *Completeness*: A circuit is said to be complete if it captures all nodes necessary to explain the model's behavior on task T.

While faithfulness and minimality are typically straightforward to measure, assessing completeness is more challenging. For further discussion, please refer to Section 6.

2.3 Continuous Sparsification

Continuous sparsification originates from model pruning and has been introduced to sparsify networks, specifically their weight space, without causing performance degradation (Savarese et al., 2020). Unlike other pruning approaches (Srinivas et al., 2017; Louizos et al., 2018), continuous sparsification approximates l_0 regularization by learning a deterministic mask $\boldsymbol{m} \in \{0,1\}^N$ over the network's parameters $\boldsymbol{w} \in \mathbb{R}^N$ that indicates which weights to prune. The search for such a sparse subnetwork can be represented by the following minimization problem:

$$\min_{\boldsymbol{w} \in \mathbb{R}^N,\, \boldsymbol{m} \in \{0,1\}^N} \mathcal{L}\left(\mathcal{M}\left(\,\cdot\,; \boldsymbol{m} \odot \boldsymbol{w}\right)\right) + \lambda \|\boldsymbol{m}\|_1 \tag{3}$$

which uses the fact that $\|\boldsymbol{m}\|_0 = \|\boldsymbol{m}\|_1$ for binary masks, and where $\mathcal{L}$ denotes the loss of the network $\mathcal{M}$, while λ controls the trade-off between loss and number of parameters $\|\boldsymbol{w}\|_0$. To circumvent the combinatorial constraint imposed by the discrete space of $\boldsymbol{m} \in \{0,1\}^N$, the mask is deterministically re-parameterized as a sigmoid function $\sigma\left(\cdot\right)$ of the new variable $\boldsymbol{s} \in \mathbb{R}^N$. Specifically, this yields:

$$\min_{\boldsymbol{w} \in \mathbb{R}^N,\, \boldsymbol{s} \in \mathbb{R}^N} \mathcal{L}\left(\mathcal{M}\left(\,\cdot\,; \sigma\left(\beta \cdot \boldsymbol{s}\right) \odot \boldsymbol{w}\right)\right) + \lambda \left\|\sigma\left(\beta \cdot \boldsymbol{s}\right)\right\|_1 \tag{4}$$

where $\beta \in [1, \infty]$ represents a temperature parameter for which the sigmoid function converges to the Heaviside function with $\lim_{\beta \to \infty} \sigma\left(\beta \cdot \boldsymbol{s}\right) = H(\boldsymbol{s})$. By minimizing the above loss via gradient descent while jointly annealing β, and recovering the binary mask from the re-parameterization via $\boldsymbol{m} = H(\boldsymbol{s})$, a sparse representation of the network's parameters can be learned.

3 Activation Pruning Through Continuous Sparsification

To automatically identify circuits for sequence-to-sequence prediction tasks, we adopt an approach similar to the subnetwork probing method proposed by Conmy et al. (2023) and formulate the identification process as a minimization problem. For this, we leverage techniques from both causal mediation analysis (Pearl, 2001) and model pruning, specifically *continuous sparsification* (Savarese et al., 2020). Unlike traditional pruning, which aims to reduce model complexity by creating a sparse and efficient representation of the model's weight space (see Section Section 2.3), we focus on intervening on the model's activations, thereby linking our method to the causal mediation analysis that underlies activation patching.

As outlined in Section 2.2, we represent a circuit as a binary mask $\boldsymbol{m} \in \{0,1\}^N$ over the model's mediator space—here the activation space of the model components considered—, indicating which activations $\boldsymbol{z}$ of a frozen model $\mathcal{M}$ are

responsible for its behavior on a given task T. To find a circuit, or binary mask $\boldsymbol{m}$, that is both *faithful* and *minimal* (we exclude *completeness* for now; see Section 6 for a detailed discussion on *completeness*), we aim to minimize the following loss term:

$$\min_{\boldsymbol{m}\in\{0,1\}^N} \mathcal{L}_T\left(\mathcal{M}(\cdot), \mathcal{M}\left(\cdot \mid \text{do}\left(\boldsymbol{z} = \boldsymbol{m} \odot \boldsymbol{z}_x + (1-\boldsymbol{m}) \odot \tilde{\boldsymbol{z}}\right)\right)\right) + \lambda \cdot \mathcal{L}_{reg}\left(\boldsymbol{m}\right) \tag{5}$$

where $\mathcal{L}_T$ captures the circuit's task faithfulness, with lower values indicating greater faithfulness, while $\mathcal{L}_{reg}$ assesses the size of the circuit. The hyperparameter λ controls the influence of $\mathcal{L}_{reg}$ on the total loss term. Note that the circuit's task faithfulness is computed by comparing the output of the full model with the output when activations outside the circuit ($m_i = 0$) are ablated by $\tilde{\boldsymbol{z}}$, while activations inside the circuit ($m_i = 1$) are kept unperturbed ($\boldsymbol{z} = \boldsymbol{z}_x$), as illustrated in Equation (2). Given the combinatorial complexity of optimizing binary masks over a potentially large activation space, direct optimization of $\boldsymbol{m}$ can be computationally prohibitive. However, as discussed in Section 2.3, the mask can be deterministically re-parameterized using a sigmoid function $\sigma(\cdot)$ over a new variable $\boldsymbol{s} \in \mathbb{R}^N$, following the approach of continuous sparsification (Savarese et al., 2020). Consequently, the optimization reduces to the following term:

$$\min_{\boldsymbol{s}\in\mathbb{R}^N} \mathcal{L}_T\left(\mathcal{M}(\cdot), \mathcal{M}\left(\cdot \mid \text{do}\left(\boldsymbol{z} = \sigma\left(\beta \cdot \boldsymbol{s}\right) \odot \boldsymbol{z}_x + \left(1-\sigma\left(\beta \cdot \boldsymbol{s}\right)\right) \odot \tilde{\boldsymbol{z}}\right)\right)\right) + \lambda \cdot \mathcal{L}_{reg}\left(\sigma\left(\beta \cdot \boldsymbol{s}\right)\right) \tag{6}$$

where the sigmoid function σ is applied element-wise, and β serves as a temperature parameter that increases progressively after each training epoch, following an exponential schedule until it reaches a maximum value, β_{max}, as proposed by Lepori et al. (2023). As β approaches infinity ($\beta \to \infty$), the sigmoid function converges to the Heaviside step function, i.e., $\sigma\left(\beta \cdot \boldsymbol{s}\right) \to H(\boldsymbol{s})$. To assess the circuit's task faithfulness, we use the Kullback-Leibler divergence $D_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$, which captures the difference between the circuit's predicted output distribution and that of the full model. Since the full model's predicted output distribution $\boldsymbol{y}^{\mathcal{M}}$ is independent of the variable $\boldsymbol{s}$, minimizing the KL divergence is equivalent to minimizing the cross-entropy between $\boldsymbol{y}^{\boldsymbol{m}}$ and $\boldsymbol{y}^{\mathcal{M}}$. Accordingly, we define the cross-entropy loss $\mathcal{L}_{CE}$ as our task faithfulness loss. For regularization, we apply l_1 regularization in line with continuous sparsification (see Equation (4)). Thus, the final minimization problem is formulated as follows:

$$\min_{\boldsymbol{s}\in\mathbb{R}^N} \mathcal{L}_{CE}\left(\mathcal{M}(\cdot), \mathcal{M}\left(\cdot \mid \text{do}\left(\boldsymbol{z} = \sigma\left(\beta \cdot \boldsymbol{s}\right) \odot \boldsymbol{z}_x + \left(1-\sigma\left(\beta \cdot \boldsymbol{s}\right)\right) \odot \tilde{\boldsymbol{z}}\right)\right)\right) + \lambda \cdot \left\|\sigma\left(\beta \cdot \boldsymbol{s}\right)\right\|_1 \tag{7}$$

By minimizing the expression in Equation (7), we obtain an approximation of $\boldsymbol{m}$ that strikes a balance between *faithfulness* and *minimality*, with λ governing the emphasis on the latter. It is important to note that during training, we optimize solely with respect to the mask approximation, specifically $\boldsymbol{s}$, while the original model's weights and activations remain unchanged. Once training converges, the binary mask is derived through re-parameterization via $\boldsymbol{m} = H(\boldsymbol{s})$, representing the final circuit identified.

In comparison to the subnetwork probing method proposed by Conmy et al. (2023), our approach identifies circuits through deterministic re-parameterization of the binary mask $\boldsymbol{m} \in \{0,1\}^N$ as a sigmoid function of the new variable $\boldsymbol{s} \in \mathbb{R}^N$. In model pruning, this deterministic approach has demonstrated superior performance over stochastic methods (Savarese et al., 2020; Lepori et al., 2023; Gale et al., 2019). Additionally, unlike activation patching, our approach automates the search for causally relevant nodes, enabling simultaneous assessment of the causal effects of all nodes considered, guided by the optimization process. Finally, our method supports different levels of node granularity; in this study, we focus on neuron-level interventions.

4 Experiments

This work studies the modularity of a transformer-based sequence-to-sequence model by analyzing circuits responsible for its behavior on highly compositional subtasks. Specifically, we employ *activation pruning through continuous sparsification* (as outlined in Section 3) to identify circuits associated with the ten string-edit operations introduced by PCFG SET (Hupkes et al., 2020), a dataset designed to study compositionality within neural networks. The experimental setup, including an overview of the dataset and details on the training and evaluation processes, is outlined in Section 4.1. The corresponding results are presented in Section 4.2.

Table 1: The string-edit operations in PCFG SET from Hupkes et al. (2020).

Unary operation	Input	Output
`copy`	$x_1 \dots x_n$	$x_1 \dots x_n$
`echo`	$x_1 \dots x_n$	$x_1 \dots x_n x_n$
`repeat`	$x_1 \dots x_n$	$x_1 \dots x_n x_1 \dots x_n$
`reverse`	$x_1 \dots x_n$	$x_n \dots x_1$
`swap`	$x_1 \dots x_n$	$x_n x_2 \dots x_{n-1} x_1$
`shift`	$x_1 \dots x_n$	$x_2 \dots x_n x_1$

Binary operation	Input	Output
`append`	x, y	$x\, y$
`prepend`	x, y	$y\, x$
`remove_first`	x, y	y
`remove_second`	x, y	x

4.1 Setup

4.1.1 Dataset

We use the PCFG SET dataset (Hupkes et al., 2020) to analyze and compare circuits for functionally related tasks. As illustrated in Table 1, the dataset (Hupkes et al., 2020) comprises ten different string-edit operations (SET) applied to sequences generated by a probabilistic context-free grammar (PCFG). All tasks resemble translation problems, where an input sequence is transformed into a corresponding output sequence through the recursive application of the string-edit operations specified within the input sequence. The dataset includes two types of functions: unary and binary string-edit operations. Unary functions operate on a single string sequence, whereas binary functions require two input arguments. For instance, the binary function `prepend` places the second argument before the first (e.g., `prepend` $A1\,, B1 \rightarrow B1\, A1$). The input alphabet in PCFG SET consists of three categories: i) words representing string-edit operations (e.g., `copy` or `echo`), ii) symbols forming the input sequence (e.g., `A1`, `B1`, etc.), and iii) a separator ", " that distinguishes the two arguments for binary operations. For additional information and examples, please refer to Appendix A.1.

Hupkes et al. (2020) construct PCFG SET in such a way that compositionality is a salient feature of the dataset. Notably, all operators in PCFG SET are functionally related. For example, the `repeat` operator can be replicated by applying the `copy` operation two times in succession (see Table 1). Our objective is to identify circuits for each of the ten string-edit operations in PCFG SET. To this end, we generate ten distinct data subsets, each containing examples from a specific string-edit operation. For each task, we generate 20,000 samples, with 16,000 allocated to the training set and 4,000 to the test set. Details regarding the data generation process and the individual sub-datasets can be found in Appendix A.2.

4.1.2 Training

Base Model Training. As a first step, we train a base model $\mathcal{M}$ to perform all ten string-edit operations in PCFG SET. Specifically, we use an encoder-decoder model based on the Transformer architecture (Vaswani et al., 2017), comprising six encoder and decoder layers with a hidden state dimension of 512, resulting in approximately 58 million parameters. The model is trained on the official data splits of PCFG SET Hupkes et al. (2020), which include around 83,000 training samples covering all string-edit operations and their compositions. Additional details on the training procedure and corresponding hyperparameters can be found in Appendix B.3.

Mask Training. Next, we aim to identify circuits corresponding to each subtask. This is accomplished by training a binary mask $\boldsymbol{m} \in \{0, 1\}^N$ over the base model's causal node space $\mathcal{Z}$, as described in Section 3. For each subtask, we minimize the loss as described in Equation (7) by training the mask on the respective subtask dataset (see Section 4.1.1). We consider individual output activations from the feed-forward and multi-head attention modules as mediators $\boldsymbol{z} = a_i \in \boldsymbol{a}$. For the ablation value $\tilde{\boldsymbol{z}}$, we conduct experiments with both zero and mean ablations. We use the same ablation value across all token positions and adopt an approach similar to node patching, where interventions target the model's residual stream. When employing mean ablations, we follow the approach proposed by Wang et al. (2023) and use the mediator's mean value across a reference distribution, specifically the subtask dataset. This removes variation-specific information while preserving constant information from the reference distribution. Further details on the mask training procedure and associated hyperparameters can be found in Appendix B.4.

It is worth noting that while we choose to apply neuron-level ablations for the feed-forward and multi-head attention modules, activation pruning as introduced in Section 3 can be extended to other levels of granularity. For instance, one could adopt an approach similar to edge attribution patching (Syed et al., 2023) by applying a mask to individual edges representing inputs to other layers of interest. Alternatively, entire modules could be ablated rather than individual neurons, thereby reducing the dimension of the mask.

(a) F_T across all token positions (b) F_T for selected token positions (c) Accuracy

Figure 1: Task faithfulness performance F_T and accuracy for the **unary** tasks. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.

4.1.3 Evaluation

We evaluate circuits according to two primary criteria: performance and node overlap. Regarding performance, we assess each circuit on each string-edit operation in PCFG SET (see Section 4.1.1 for details). Specifically, we measure both the circuit's faithfulness to the original model's behavior, as defined by Equation (2), and its accuracy on the assigned task. As outlined in Section 3, the circuit's faithfulness is quantified via the KL-divergence $D_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$, which captures the difference between the circuit's predicted output distribution and that of the full model. However, since the KL-divergence is unbounded and therefore unsuitable for cross-task comparisons, we also use a normalized version of the Jensen–Shannon divergence JSD_{norm} to evaluate the similarity between the circuit's predictions and those of the model (see Equation (8) in Appendix B.5). Note that $0 \leq \text{JSD}_{\text{norm}} \leq 1$. For visualization purposes, and to convert the loss into a performance metric, we define the circuit's *faithfulness performance* on task T as $F_T = 1 - \text{JSD}_{\text{norm}}$, where values approaching 1 indicate higher task faithfulness. Moreover, the circuit's accuracy is assessed by comparing the exact match between the circuit's predicted and the ground truth output sequence. When evaluating a circuit identified for task T on a different task $\hat{T}$, we retain the mean values from task T as ablation values for nodes with $m_i = 0$ when using mean ablations. For a discussion on this approach, please refer to Section 6.

In addition to performance, circuits are compared based on their node overlap. For this, we use two metrics: Intersection over Union (IoU) and Intersection over Minimum (IoM), as defined in Equation (9) in Appendix B.5. Note that the IoU captures the overall overlap of the circuits relative to their combined size, whereas the IoM measures how much of the smaller circuit is contained within the larger one.

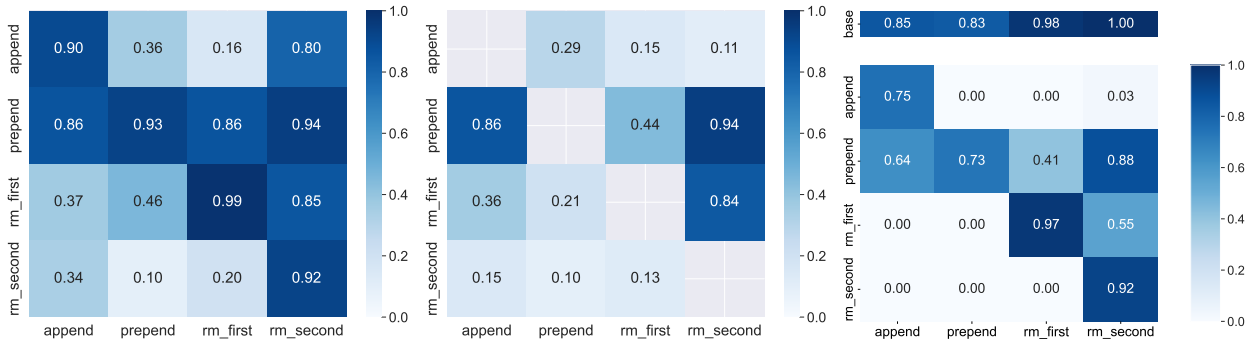
4.2 Results

We begin by presenting the performance results of the base model on PCFG SET. After training, the encoder-decoder model exhibits strong performance across all ten string-edit operations, achieving accuracy rates exceeding 95% on unary functions and ranging between 83% and 99% on binary tasks. A detailed breakdown of the base model's task-specific accuracy is provided in Appendix B.3. As shown in Table 3 in the appendix, the most challenging operations for the model are `prepend` and `append`.

In the following sections, we report results for the circuits identified in this study. Due to space constraints, we focus primarily on the circuits discovered via mean ablation. For additional results concerning circuits identified through zero ablation, please refer to Appendix C.2.

4.2.1 Circuit Performance

We first analyze the circuits' performance across different string-edit operations. Figure 1 illustrates both the task faithfulness performance, F_T, as defined in Section 4.1.3, as well as the accuracy of each circuit with respect to the six unary operations. Due to the functional similarities between the string-edit operations, multiple operators may produce the same output tokens at various token positions. Therefore, we assess faithfulness performance under two configurations: one where F_T is averaged across all output tokens (Figure 1a) and another where faithfulness is calculated only at positions where the ground truth output sequences of the circuit-task and evaluation-task diverge



(a) All token positions. (b) Differing token positions. (c) Accuracy.

Figure 2: Task faithfulness performance F_T and accuracy for the **binary** tasks. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.

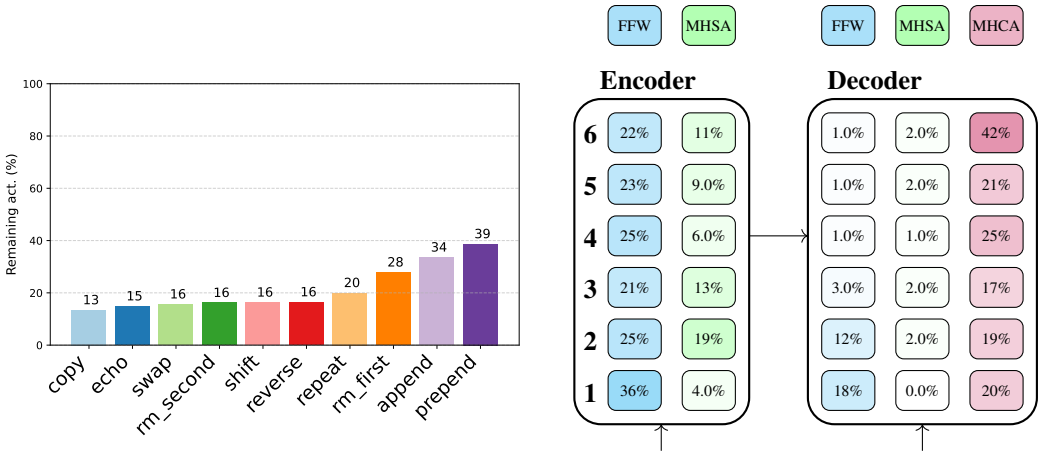
(Figure 1b). For example, when evaluating the `copy` circuit on the `echo` task, the evaluation focuses on measuring task faithfulness at the final token of the target output sequence —specifically, the additional x_n in `echo` which differs from the end-of-sequence token in `copy` (see Table 1).

Unary Circuits. Analyzing the diagonal in Figure 1a, it is evident that all unary circuits demonstrate a high level of faithfulness performance, with rates exceeding 0.94 on their respective tasks. Additionally, several circuits, such as `echo`, `repeat`, and `swap`, exhibit strong cross-task faithfulness and accuracy on the `copy` task. This observation aligns with human intuition, as the `copy` operation is either a significant component of these string-edit operations or can be effectively performed by these functions. In contrast, operators like `reverse` and `shift`, which substantially alter the input sequence, demonstrate lower functional similarity to the `copy` operation. This is reflected in both the reduced cross-task faithfulness (see Figure 1a; 1b) and the lower accuracy (Figure 1c) of the `reverse` and `shift` circuits on the `copy` task. Interestingly, while the `repeat` and `swap` circuits perform well on the `copy` task, the `copy` circuit does not perform as effectively on these tasks, despite the theoretical possibility of completing the repeat operation through two consecutive applications of the `copy` operator.

For many circuits, we observe high cross-task faithfulness performance when considering all tokens, as illustrated in Figure 1a. However, their faithfulness scores on tokens that differ between the circuit and the evaluation task, as well as their cross-task accuracy, tend to be low. For example, the `copy` circuit demonstrates a notable cross-task faithfulness score of 0.64 on the `swap` task when evaluated across the entire output sequence, as shown in Figure 1a. When focusing solely on the tokens that differ between the ground truth output sequences of `copy` and `swap`, the faithfulness score drops significantly to 0.01 (see Figure 1b). This suggests that the `copy` circuit predominantly adheres to its designated `copy` function, even when the input sequence demands a `swap` operation. The instances where the `copy` circuit generates the correct output for the `swap` task —reflected in its cross-task accuracy of 0.16 —may be attributed to the fact that applying the `swap` operation twice yields the same result as `copy`.

Binary Circuits. Figure 2 illustrates the performance of circuits associated with the binary string-edit operations. Similar to the unary circuits, these binary circuits exhibit a high level of faithfulness for their respective tasks, with performance values ranging from 0.90 to 0.99, as illustrated in Figure 2a. Notably, the `remove_first` circuit demonstrates strong cross-task faithfulness and accuracy when applied to the `remove_second` task, although the reverse is not true. Additionally, the `prepend` circuit shows consistently strong performance across all binary tasks. When analyzing the size of each circuit —i.e., the fraction of nodes with causal relevance to the model's task behavior, expressed as a fraction of remaining activations, as displayed in Figure 3a —we observe that the `prepend` circuit is the largest among all circuits, retaining 39% of its activations. Similarly, the `remove_first` circuit is notably larger than the `remove_second` circuit. A potential hypothesis for this pattern is that more complex operations, such as `prepend`, `append`, or `remove_first` engage a larger portion of the model's activation space compared to simpler functions.[2] For example, the `remove_second` operation may be accomplished through a straightforward `copy` operation while producing an end-of-sequence token at the position of the separator ",". In contrast, the `remove_first` operation might not be executed as easily.

[2]Although there is currently insufficient evidence to confirm this hypothesis, we consider it a valuable perspective worthy of inclusion. For a more detailed discussion, we refer to Section 6.



(a) The global sparsity of the mean ablated circuits.

(b) The local sparsity of the `copy` circuit.

Figure 3: Node overlap for circuits identified via activation pruning with mean ablation.

Local sparsity. In Figure 3b, we visualize the local sparsity of the `copy` circuit, which is the smallest among all circuits. Specifically, we depict the percentage of remaining activations of the feed-forward (FFW), multi-head self-attention (MHSA), and multi-head cross-attention (MHCA) modules for each layer. Interestingly, in the decoder, we observe that the MHCA modules are predominantly active, whereas the activations in the FFW and MHSA modules are nearly entirely pruned. This might be expected, as the copy task primarily involves directing the input data processed by the encoder to the output. Additional visualizations for other circuits can be found in Appendix C.1.2.

Zero vs. mean ablations. The results so far relate to circuits identified via mean ablation. It is also possible to zero-ablate nodes, respectively. However, similar to prior work by Miller et al. (2024), we find circuits identified via zero ablations to be less faithful. For a detailed overview of respective results, please refer to Appendix C.2.

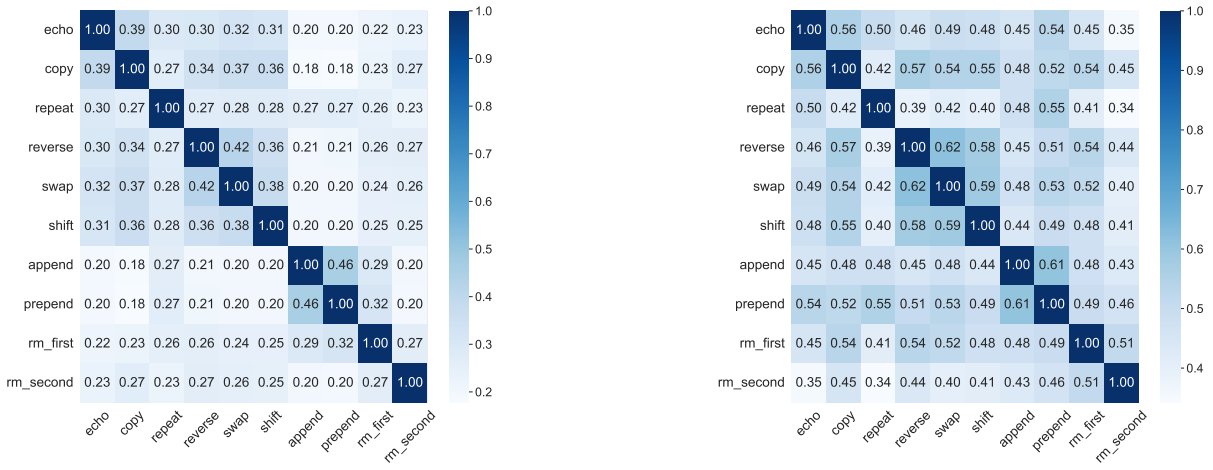
4.2.2 Circuit Overlap

We compare the circuits identified for all ten string-edit operations by illustrating their respective node overlap in Figure 4. Specifically, we assess the IoU and IoM between all circuit pairs, as described in Section 4.1.3. Upon analyzing the overall IoU among circuits, we observe that most circuit pairs exhibit values of around 0.20 to 0.30. However, certain clusters of circuits show a higher degree of node overlap. For example, the circuits for `reverse`, `swap`, and `shift` demonstrate IoU values ranging from 0.36 to 0.42. This is further reflected in their IoM, which is with values around 0.60 notably higher than for most other circuit pairs. This suggests that the base model $\mathcal{M}$ may reuse a subset of the shared activations for all three tasks. Given the functional similarities of these operations, this finding aligns with intuitive expectations. However, it is worth noting that the cross-task performance of `reverse`, `swap`, and `shift` on their respective tasks, as depicted in Figure 1, is lower than that of other circuits. A similarly high node overlap is observed for the `append` and `prepend` circuits, both associated with tasks that the base model finds particularly challenging.

In Section 4.2.1, we demonstrated that the `echo`, `repeat` and `swap` circuits exhibit high cross-task performance on the `copy` task. This raises the question to what extent the `copy` circuit is embedded within these circuits. Upon examining the node overlap between the `copy` circuit and the aforementioned circuits, we find their IoM values to range from 0.42 to 0.56, indicating that approximately half of the nodes in the `copy` circuit are contained within these other circuits. For a more detailed discussion of the implications of these findings, please refer to Section 6.

4.2.3 Circuit Compositions: Subnetwork Set Operations

Given that we can define a circuit as binary mask $\boldsymbol{m} \in \{0,1\}^N$ over the model's mediator space, we extend our study on *cross-circuit relationships* by creating compositions through basic set operations on these masks. Specifically, for a circuit-pair $\left(\boldsymbol{m}^{T_1}, \boldsymbol{m}^{T_2}\right)$, we define their composite as the union $\boldsymbol{m}^{T_1,T_2} = \boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$. Importantly, this union is not symmetric: we use the ablation values $\tilde{\boldsymbol{z}}^{T_1}$ of the first circuit for nodes deemed causally irrelevant after the union ($m_i^{T_1,T_2} = 0$). For additional details on how this union is applied, we refer to Appendix B.6.



(a) Intersection over Union (IoU).

(b) Intersection over Minimum (IoM).

Figure 4: Node overlap for circuits identified via activation pruning with mean ablation.

Figure 5 illustrates the cross-task accuracy of various circuits and their composites, derived from the union operation previously described. The results demonstrate the existence of composite circuits which acquire functional capacities for subtasks that the original base circuits could not handle individually. For example, while neither the `repeat` nor the `reverse` circuit can solve the `echo` task alone, their union achieves a notable accuracy of 78%. Similarly, the composite of `swap` and `reverse` shows a performance improvement on the `shift` task, reaching an accuracy of 33%. For tasks that can be solved by at least one of the original circuits, the performance of their composite generally reflects a combination of their individual accuracies. For instance, the `repeat` circuit achieves 90% accuracy on the `copy` task, while the `reverse` circuit yields 0.0% accuracy. Their union achieves an intermediate accuracy of 80% on the task. Similarly, the `swap` circuit achieves 27% accuracy on the `reverse` task, whereas the `reverse` circuit reaches up to 96%. Their composite circuit achieves a balanced performance of 45%. This implies that improvements in accuracy for novel tasks may be accompanied by a decline in performance on previously mastered skills. Further experiments and results on subnetwork set operations can be found in Appendix C.1.4.

A possible explanation for these observations is that the union operation $\boldsymbol{m}^{T_1,T_2} = \boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$ shifts the activation space of $\boldsymbol{m}^{T_1}$ towards $\boldsymbol{m}^{T_2}$, effectively transforming the input into a feature space that aligns with one of the subtasks the model $\mathcal{M}$ can solve. For instance, the union $\boldsymbol{m}^{\texttt{swap}} \cup \boldsymbol{m}^{\texttt{reverse}}$ might shift the activation space of $\boldsymbol{m}^{\texttt{swap}}$ towards $\boldsymbol{m}^{\texttt{reverse}}$, essentially allowing the new circuit $\boldsymbol{m}^{\texttt{swap, reverse}}$ to to perform the `shift` operation, while sacrificing some performance on the `swap` task.

5 Related Work

Techniques that discover circuits through causal mediation analysis have been successfully applied across various domains and tasks, including the study of gender bias in language models (Jeoung & Diesner, 2022; Chintam et al., 2023), different forms of factual recall (Meng et al., 2022; Geva et al., 2023), subject-verb agreement (Chintam et al., 2023), and arithmetic operations (Nanda et al., 2023; Hanna et al., 2024a). Finding subnetworks has also been of interest outside of the mechanistic interpretability literature. Previous work has studied task-specific subnetworks through clustering approaches (Casper et al., 2022; Watanabe, 2019), pruning Csordás et al. (2020); Cao et al. (2021); Lepori et al. (2023), sparse fine-tuning (Ansell et al., 2022), and adapters (Pfeiffer et al., 2021; Rücklé et al., 2021).

6 Discussion

In this section, we discuss the implications of our findings, particularly focusing on the nature of the identified circuits and their significance for the model's modularity.

What constitutes a circuit? We introduce *activation pruning through continuous sparsification*, a method that formulates the circuit discovery process as an optimization problem balancing *task faithfulness* and *minimality*. Our

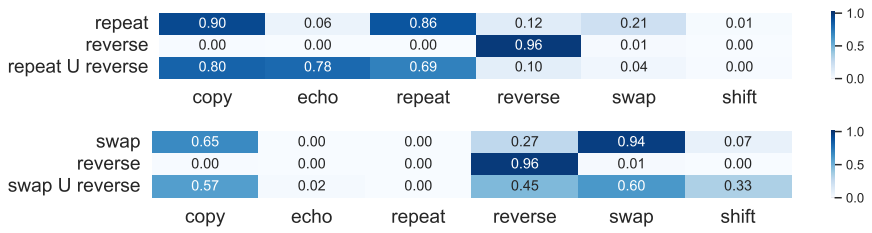


Figure 5: The results of combining circuits through a union operation on their respective binary masks.

results demonstrate that this method successfully identifies circuits that are both faithful and sparse for each subtask within PCFG SET. However, as discussed in Section 2.2, circuits are further expected to be *complete*, meaning they should include all nodes involved in the model's computations for a given task. We emphasize that our current approach does not guarantee *completeness*. While some methods attempt to evaluate completeness by analyzing circuit behavior under ablations (Wang et al., 2023), these approaches are computationally expensive, especially when dealing with large circuit sizes. Future work is needed to develop more efficient ways to quantify completeness and incorporate this objective into the optimization framework described in Equation (7). In addition, we observe that the circuits identified are influenced by the ablation value $\tilde{z}$ used during the identification process. For example, when comparing circuits based on mean versus zero ablation, we find notable differences in the nodes identified as causally relevant, even for zero-ablated circuits that demonstrate high task faithfulness (see results in Appendix C.2, specifically Figure 14). This raises a broader question about what truly defines a circuit. In our setup, a circuit's behavior in response to an input is influenced not only by the nodes deemed causally significant ($m_i = 1$) but also by the values $\tilde{z}$ of the nodes where $m_i = 0$. Essentially, the behavior of a circuit is determined not merely by what it includes but also by what it excludes, as similarly noted by Miller et al. (2024). We believe this is a general characteristic of methods that rely on constant perturbations, such as zero and mean ablations (Olsson et al., 2022; Wang et al., 2023).

Modularity in Neural Networks. Our results presented in Section 4.2 demonstrate that circuits responsible for tasks such as `echo`, `repeat` and `swap` are capable of successfully completing the `copy` task, whereas the `copy` circuit does not perform effectively on these other tasks. This raises the question: do the `echo`, `repeat`, and `swap` circuits leverage the `copy` operation for their respective functions? Our analysis on node overlap suggests that the `copy` circuit —which is notably the smallest among those identified, as shown in Figure 3a—is largely embedded within these other circuits. Moreover, circuits that share functional similarities exhibit greater node overlap compared to those that are functionally distinct. While it is premature to draw definitive conclusions based on this experimental evidence, we underscore the general observation that more complex operations tend to engage a larger portion of the model's activations and are capable of solving multiple tasks. Furthermore, we find that circuits combined via the union operator can represent novel functional capabilities. This might suggest some level of modularity within the network. One might argue that our identification process has not produced truly minimal circuits. While it is inherently challenging to determine whether a circuit is truly minimal, we have mitigated this concern by conducting an extensive hyperparameter search, ensuring a fair balance between faithfulness and minimality. Lastly, it is important to note that our study focuses on a small encoder-decoder model trained on a single dataset like PCFG SET. Naturally, the question arises as to whether these findings will generalize to larger models commonly used in practice. This touches on a broader concern within the field of mechanistic interpretability, where many studies are conducted on smaller models and datasets (Elhage et al., 2022). While scaling our findings to larger models is an area for future research, we believe the methodology used in this study offers valuable insights for future investigations into the compositionality of larger, more complex models.

7 Conclusion

This study explores modularity in neural networks by analyzing circuits in a transformer-based language model for highly compositional subtasks. Specifically, we train an encoder-decoder model using the PCFG SET dataset and identify circuits responsible for ten modular string-edit operations. To achieve this, we introduce *activation pruning through continuous sparsification*, a method that allows us to formulate circuit identification as minimization problem. Our results demonstrate that this approach successfully identifies both faithful and sparse circuits for each subtask within PCFG SET. Additionally, we assess all circuits by examining their cross-task performance and node overlap. Finally, we show that new functional circuits can be composed through set operations, such as the union of two circuits.

8 REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our experiments, we make all code publicly available as supplementary material and provide comprehensive instructions along with user-friendly bash scripts for replicating our results. Details of the training process, including the computational setup, model implementation, and hyperparameter selection, are thoroughly documented in Section 4.1.2 and Appendix B. Similarly, a detailed account of the evaluation procedure can be found in Section 4.1.3 and Appendix B.5. Furthermore, all data used in this work is either publicly available or accompanied by a detailed description of the data generation process in Appendix A, supported by scripts that facilitate its reproduction.

REFERENCES

Alan Ansell, Edoardo Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1778–1796, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.125. URL https://aclanthology.org/2022.acl-long.125.

Steven Cao, Victor Sanh, and Alexander Rush. Low-complexity probing via finding subnetworks. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 960–966, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.74. URL https://aclanthology.org/2021.naacl-main.74.

Stephen Casper, Shlomi Hod, Daniel Filan, Cody Wild, Andrew Critch, and Stuart Russell. Graphical clusterability and local specialization in deep neural networks. In *ICLR 2022 Workshop on PAIR2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

Abhijith Chintam, Rahel Beloch, Willem Zuidema, Michael Hanna, and Oskar van der Wal. Identifying and adapting transformer-components responsible for gender bias in an English language model. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 379–394, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.29. URL https://aclanthology.org/2023.blackboxnlp-1.29.

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

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*, 2020.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv e-prints*, arXiv:1902.09574, 2019. URL https://arxiv.org/abs/1902.09574.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12216–12235, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.751. URL https://aclanthology.org/2023.emnlp-main.751.

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

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

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Sullam Jeoung and Jana Diesner. What changed? investigating debiasing methods using causal mediation analysis. In Christian Hardmeier, Christine Basta, Marta R. Costa-jussà, Gabriel Stanovsky, and Hila Gonen (eds.), *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*, pp. 255–265, Seattle, Washington, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.gebnlp-1.26. URL https://aclanthology.org/2022.gebnlp-1.26.

Michael Lepori, Thomas Serre, and Ellie Pavlick. Break it down: Evidence for structural compositionality in neural networks. *Advances in Neural Information Processing Systems*, 36:42623–42660, 2023.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

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. URL https://openreview.net/forum?id=fpoAYV6Wsk.

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.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability, 2024. URL https://arxiv.org/abs/2408.01416.

Neel Nanda. Attribution patching: Activation patching at industrial scale, 2023. URL https://www.neelnanda.io/mechanistic-interpretability/attribution-patching. Accessed: 2024-09-19.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Judea Pearl. Direct and indirect effects. In *Proceedings of the Seventeenth Conference on Uncertainty and Artificial Intelligence, 2001*, pp. 411–420. Morgan Kaufman, 2001.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, 2021.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *ArXiv*, abs/2402.14811, 2024. URL https://api.semanticscholar.org/CorpusID:267783084.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7930–7946, 2021.

Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. *Advances in neural information processing systems*, 33:11380–11390, 2020.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training sparse neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 455–462, 2017. doi: 10.1109/CVPRW.2017.61.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

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.

Chihiro Watanabe. Interpreting layered neural networks via hierarchical modular representation. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26*, pp. 376–388. Springer, 2019.

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://openreview.net/forum?id=Hf17y6u9BC.

Table 2: The string-edit operations in PCFG SET from Hupkes et al. (2020).

Unary operation	Input	Output	Example
copy	$x_1 \dots x_n$	$x_1 \dots x_n$	copy K1 Y1 W1 K1 → K1 Y1 W1 K1
echo	$x_1 \dots x_n$	$x_1 \dots x_n x_n$	echo E1 K1 A1 X1 J → E1 K1 A1 X1 J1 J1
repeat	$x_1 \dots x_n$	$x_1 \dots x_n x_1 \dots x_n$	repeat J1 F1 S1 → J1 F1 S1 J1 F1 S1
reverse	$x_1 \dots x_n$	$x_n \dots x_1$	reverse G1 T1 X1 J1 → J1 X1 T1 G1
swap	$x_1 \dots x_n$	$x_n x_2 \dots x_{n-1} x_1$	swap B1 Z1 V1 I1 W1 → W1 Z1 V1 I1 B1
shift	$x_1 \dots x_n$	$x_2 \dots x_n x_1$	shift Y1 I1 D1 H1 K1 → I1 D1 H1 K1 Y1
Binary operation	**Input**	**Output**	**Example**
append	x, y	$x\, y$	append F1 B1, U1 A1 G1 → F1 B1 U1 A1 G1
prepend	x, y	$y\, x$	prepend F1 B1, U1 A1 G1 → U1 A1 G1 F1 B1
remove_first	x, y	y	remove_first Z1 P1 N1, A1 D1 → A1 D1
remove_second	x, y	x	remove_second F1 B1, U1 A1 G1 → F1 B1

A Dataset

A.1 PCFG SET

Hupkes et al. (2020) construct PCFG SET in such a way that compositionality is a salient feature of the dataset, while aligning its statistical properties with those of natural language corpora, specifically English. We present examples for each string-edit operation in Table 2 to help the reader get more familiar with the functions employed in PCFG SET Hupkes et al. (2020).

A.2 Generating isolated function data

We use the probabilistic context-free grammar proposed by Hupkes et al. (2020) to generate datasets for each of the string-edit operations originally used to construct PCFG SET. Using the same vocabulary, consisting of letters $[A-Z]$ combined with numbers in $(0, \infty] \in \mathcal{Z}^+$, we restrict the grammar to produce samples that use only specific string-edit operation for each subtask dataset, e.g, copy W1 O1 Z5 G1. For each SET operation, we generate 16 0000 samples for training and 4 000 for validation. Each subtask dataset includes samples of both single-use and composed applications of that function, such as remove_first remove_first $A1$, $B1$, $B1\, A1 \rightarrow B1\, A1$.

B Experimental details

We provide all code for our experiments through the supplementary material. In this section, we outline key details regarding our experiments, such as our computational setup and the hyperparameters selected.

B.1 Software and computing

All experiments are implemented using PyTorch (Paszke et al., 2019) as the primary framework. Detailed information about supporting software and specific versions can be found in our code repository. All experiments have been conducted using AMD Instinct MI250X accelerators.

B.2 Base model architecture

The base model $\mathcal{M}$ is a transformer-based encoder-decoder, as proposed by Vaswani et al. (2017), but with the Gated Linear Unit (GLU) variant from Shazeer (2020). We use sinusoidal positional encoding, delimit tokens on white-space, and disjoint embedding matrices for the input and output sequences. The base model $\mathcal{M}$ consists of six encoder and six decoder layers, with a hidden size of dimension 512, and eight attention heads per layer.

B.3 Base model training

We train the base model $\mathcal{M}$ using a learning rate of $5 \cdot 10^{-7}$, a batch size of 64 without gradient accumulation, a gradient clipping value of 15, and a dropout rate of 0.2. Furthermore, the model is trained using the original datasets from Hupkes et al. (2020). In Table 3, we report the accuracy of the base model on each of the isolated subtask

Table 3: The generation accuracy of the base model on the different subtasks(Hupkes et al., 2020).

Unary operation	Accuracy
copy	1.0
echo	0.999
repeat	0.943
reverse	0.989
swap	0.996
shift	0.992

Binary operation	Accuracy
append	0.848
prepend	0.832
remove_first	0.975
remove_second	0.999

datasets. The base model $\mathcal{M}$ achieves final accuracy of 87.8% on the official PCFG test set, which is slightly lower than the performance of 92.0% reported by Hupkes et al. (2020) .

B.4 Mask training

In this study, we conduct experiments for both mean and zero ablation. Following the approach of Wang et al. (2023), the mean ablation value $\tilde{\boldsymbol{z}}$ is derived from a reference distribution. Specifically, when training a mask for task T_l on dataset D_l, we assign the ablated value $\tilde{\boldsymbol{z}}_i$ for each potential mediator $\boldsymbol{z}_i$ in $\mathcal{M}$ as the average activation of $\boldsymbol{z}_i$ across samples in D_l, which only includes samples corresponding to task T_l. For instance, when training a mask for the copy task, the ablated value $\tilde{\boldsymbol{z}}_i$ for each mediator $\boldsymbol{z}_i$ is set to the average activation of $\boldsymbol{z}_i$ over samples exclusively containing the copy function, i.e., $D^{\text{train}}_{\texttt{copy}}$, averaged across all token positions.

The final optimization problem, described in Equation (7), requires the predicted output probabilities $\boldsymbol{y}^{\mathcal{M}}$ of the base model $\mathcal{M}$ for each dataset sample across all of the subtask datasets. In practice, this is achieved through caching: a forward pass is performed on all datasets using $\mathcal{M}$, and $\boldsymbol{y}^{\mathcal{M}}$ is stored for later use during mask training.

B.4.1 Hyperparameters

In Table 4 we report the final hyperparameters for the mask training for each subtask, using the same notation as in Section 3. These are selected based on a random search over the following hyperparameters: learning rate: $\{1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$, λ: $\{1 \cdot 10^{-3}, 1 \cdot 10^{-4}, 1 \cdot 10^{-5}\}$, $\boldsymbol{s}_{\text{initial}}$: $\{0.2, 0.05, 0\}$, β_{max}: $\{100, 200, 300\}$.

Table 4: Final hyperparameters used for mask training.

Circuit	lr	λ	$\boldsymbol{s}_{\text{initial}}$	β_{max}	Epochs
echo	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
copy	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
repeat	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
reverse	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
swap	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
shift	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	0.05	200	500
append	$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	0	200	500
prepend	$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	0	100	500
remove_first	$1 \cdot 10^{-4}$	$1 \cdot 10^{-5}$	0.05	300	500
remove_second	$1 \cdot 10^{-3}$	$1 \cdot 10^{-5}$	0.05	100	300

B.5 Evaluation

We evaluate circuits according to two primary criteria, performance and node overlap, as described in Section 4.1.3. Specifically, we quantify the performance of a circuit by its *faithfulness* score, which is calculated via the KL divergence, D_{KL}, between the output distribution of the circuit $\boldsymbol{y}^{\boldsymbol{m}}$ and the base model $\boldsymbol{y}^{\mathcal{M}}$. However, to get a bounded metric, we instead use the normalized version of the Jensen-Shannon divergence between the two output distributions:

$$\text{JSD}_{\text{norm}}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right) = \frac{1}{2\log(2)}\left(D_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\boldsymbol{m}\mathcal{M}}\right) + D_{KL}\left(\boldsymbol{y}^{\mathcal{M}} \parallel \boldsymbol{y}^{\boldsymbol{m}\mathcal{M}}\right)\right), \quad \text{where} \quad \boldsymbol{y}^{\boldsymbol{m}\mathcal{M}} = \frac{\boldsymbol{y}^{\boldsymbol{m}} + \boldsymbol{y}^{\mathcal{M}}}{2} \tag{8}$$

where $\boldsymbol{y}^{\boldsymbol{m}\mathcal{M}}$ is a mixture distribution of $\boldsymbol{y}^{\boldsymbol{m}}$ and $\boldsymbol{y}^{\mathcal{M}}$. Note that the Jensen-Shannon divergence is symmetric. Furthermore, it is bounded, i.e., $0 \leq \text{JSD}_{\text{norm}} \leq 1$.

To measure the circuits' node overlap we compute their Intersection over Union (IoU) and Intersection over Minimum (IoM). Given two circuits $\boldsymbol{m}^{T_1} \in \{0,1\}^N$ and $\boldsymbol{m}^{T_2} \in \{0,1\}^N$ defined over the same node space, the IoU and IoM are computed as follows:

$$\text{IoU} = \frac{\|\boldsymbol{m}^{T_1} \cap \boldsymbol{m}^{T_2}\|_1}{\|\boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}\|_1} \qquad \text{IoM} = \frac{\|\boldsymbol{m}^{T_1} \cap \boldsymbol{m}^{T_2}\|_1}{\min\left(\|\boldsymbol{m}^{T_1}\|_1, \|\boldsymbol{m}^{T_2}\|_1\right)} \tag{9}$$

where $\boldsymbol{m}^{T_1} \cap \boldsymbol{m}^{T_2}$ represents the intersection of the two binary masks, while $\boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$ denotes their union. Specifically, the intersection between two circuits $\boldsymbol{m}^{T_1,T_2} = \boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$ is defined as:

$$m_i^{T_1,T_2} = m_i^{T_1} \cap m_i^{T_2} = m_i^{T_1} \wedge m_i^{T_2} = \begin{cases} 1 & \text{if} \quad m_i^{T_1} = 1 \text{ and } m_i^{T_2} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Similarly, the union between two circuits $\boldsymbol{m}^{T_1,T_2} = \boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$ can be computed as:

$$m_i^{T_1,T_2} = m_i^{T_1} \cup m_i^{T_2} = m_i^{T_1} \vee m_i^{T_2} = \begin{cases} 0 & \text{if} \quad m_i^{T_1} = 0 \text{ and } m_i^{T_2} = 0, \\ 1 & \text{otherwise.} \end{cases} \tag{11}$$

B.6 Subnetwork compositions

When creating circuit compositions, we apply the union operator between the binary masks of two circuits, as illustrated in Equation (11). For zero-ablated circuits, this operation remains symmetric. However, in the case of mean-ablated circuits, symmetry is not preserved. This is primarily due to the fact that the mediator value, $\boldsymbol{z}_i$, for which $m_i = 0$ after the union, is ablated by the mean value across the reference distribution associated with the first circuit's task, $\tilde{\boldsymbol{z}}_i^{T_1}$. Specifically, for the union $\boldsymbol{m}^{T_1,T_2} = \boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2}$, the mediator value $\boldsymbol{z}_i$ is determined as follows:

$$\boldsymbol{z}_i = \begin{cases} \tilde{\boldsymbol{z}}_i^{T_1} & \text{if} \quad m_i^{T_1} = 0 \text{ and } m_i^{T_2} = 0, \\ \boldsymbol{z}_{x_i} & \text{otherwise,} \end{cases} \tag{12}$$

This means that $\boldsymbol{m}^{T_1} \cup \boldsymbol{m}^{T_2} \neq \boldsymbol{m}^{T_2} \cup \boldsymbol{m}^{T_1}$ for mean ablations if $\tilde{\boldsymbol{z}}_i^{T_1} \neq \tilde{\boldsymbol{z}}_i^{T_2}$. For zero ablation, the ablation value remains the same for both cases $\tilde{\boldsymbol{z}}_i^{T_1} = \tilde{\boldsymbol{z}}_i^{T_2} = 0$, thus yielding symmetry of the operator. Please refer to Section 6 for further discussion.

C Supplemental results

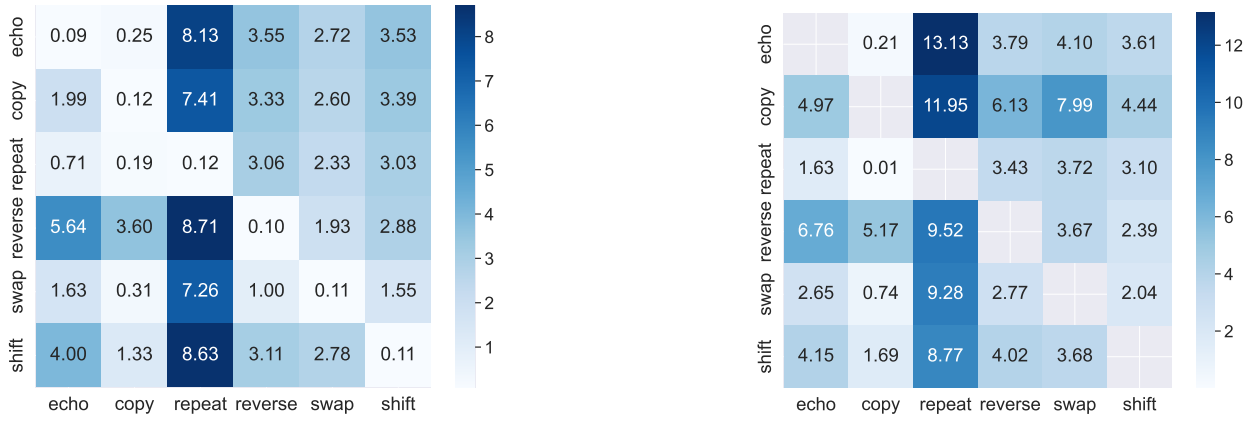
In this section, we report supplemental results that complement the paper's key findings.

C.1 Mean ablation

We first report additional results related to circuits identified via mean ablation.

C.1.1 Circuit Performance

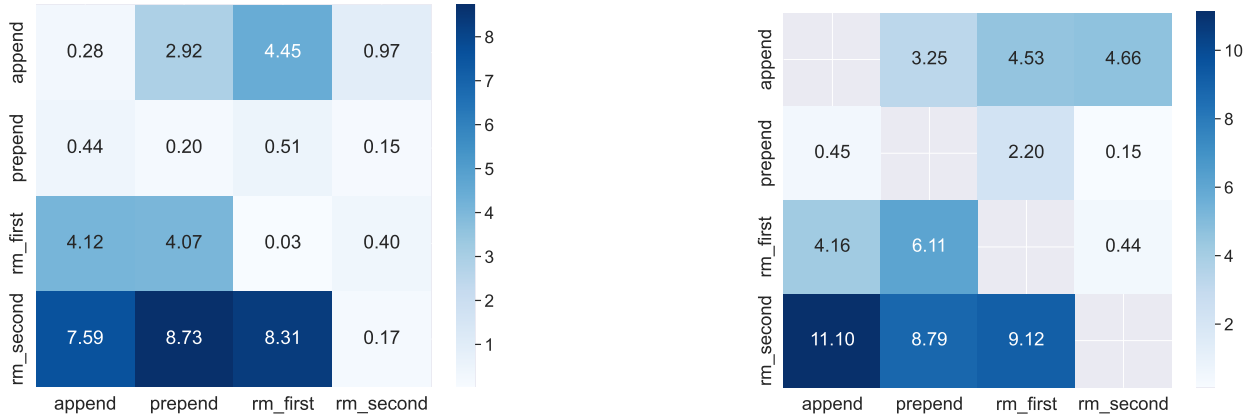
In Figure 6 and Figure 7 we report the KL divergence between the predicted output distributions of the mean-ablated circuits, $\boldsymbol{m}$, and the base model, $\mathcal{M}$, for both unary and binary string-edit operations. The results exhibit a trend consistent with the observations discussed in Section 4.2.1.



(a) All token positions.

(b) Differing token positions.

Figure 6: Task faithfulness measured via $\mathrm{D}_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$ for the **mean-ablated unary** circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.



(a) All token positions.

(b) Differing token positions.

Figure 7: Task faithfulness measured via $\mathrm{D}_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$ for the **mean-ablated binary** circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.

C.1.2 Local Sparsity

Figure 16 illustrates the local sparsity of each unary circuit identified via mean-ablations, while 17 presents the local sparsity of each binary circuit, respectively. It is evident that unary circuits are highly sparse, with no module retaining more than 45% of activations. Notably, the feed-forward (FFW) and multi-head self-attention (MHSA) modules within the decoder demonstrate pronounced sparsity across all layers, typically retaining only 0.0% to 10% of activations. In contrast, the multi-head cross-attention (MHCA) modules retain a higher proportion of activations. As shown in Figure Figure 17, binary circuits generally retain a greater number of activations. Specifically, the `append` and `prepend` circuits show substantial remaining activations in both the encoder and decoder.

C.1.3 Deterministic mask approximation.

As discussed in Section 3, a key advantage of learning a binary mask through continuous sparsification is the deterministic nature of the approach. To assess whether our method consistently converges to the same circuits despite the stochastic elements of the training process (e.g., dataset shuffling), we examine the node overlap of circuits trained using different random seeds. Our findings demonstrate that the method reliably converges to the same mask for iden-

tical subtasks, regardless of the random seed. This holds true for both mean and zero ablation. For example, we report an IoU and IoM of 1.0 when comparing five `prepend` circuits trained with varying random seeds.

C.1.4 SET OPERATIONS

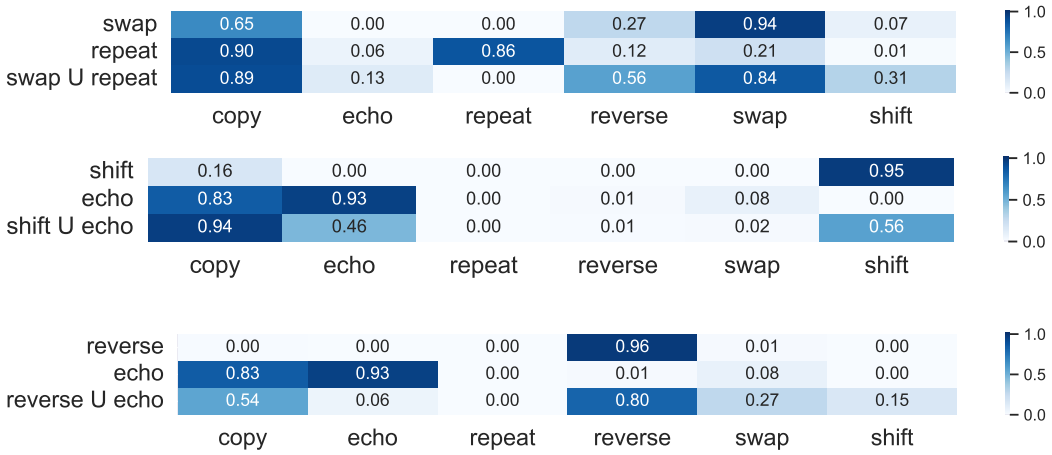


Figure 8: The results of combining circuits through a union operation on their respective binary masks.

In Figure 8 we present the cross-task accuracy of additional composite circuits as supplementary results to Section 4.2.3. Consistent with previous observations, the union of two circuits acquires functional capacities for subtasks that the original base circuits cannot perform independently. For example, the union $\boldsymbol{m}^{\texttt{swap}} \cup \boldsymbol{m}^{\texttt{repeat}}$ achieves a 56% accuracy on the `reverse` task and 31% on `shift`, representing significant improvements over the performance of these circuits in isolation. Similarly, the union $\boldsymbol{m}^{\texttt{reverse}} \cup \boldsymbol{m}^{\texttt{echo}}$ reaches 31% accuracy on `swap`. In the case of $\boldsymbol{m}^{\texttt{shift}} \cup \boldsymbol{m}^{\texttt{echo}}$, we observe that while the composite retains approximately half of the performance on the individual tasks, it also enhances performance on the `copy` task.

C.2 ZERO ABLATION

While the main paper focuses on circuits obtained through mean ablation, this section presents the results from the same set of experiments conducted using zero ablation.

C.2.1 CIRCUIT PERFORMANCE

In Figure 9 and Figure 10, we present the task faithfulness performance F_T and generation accuracy for the zero-ablated circuits. Similarly, Figure 11 and Figure 12 illustrate task faithfulness in terms of the KL divergence between the predicted output distributions of the zero-ablated circuits, $\boldsymbol{m}$, and the base model, $\mathcal{M}$, for both unary and binary string-edit operations. The key finding from these experiments is that zero ablation yields results closely aligned with those of mean ablation for binary operations, but not for unary operations. Similarly, patterns observed in mean-ablated unary circuits are not evident here. As noted in the literature, zero ablation can significantly shift the distribution, leading to degraded performance, which is one of the reasons for adopting mean ablation (Miller et al., 2024).

C.2.2 NODE OVERLAP

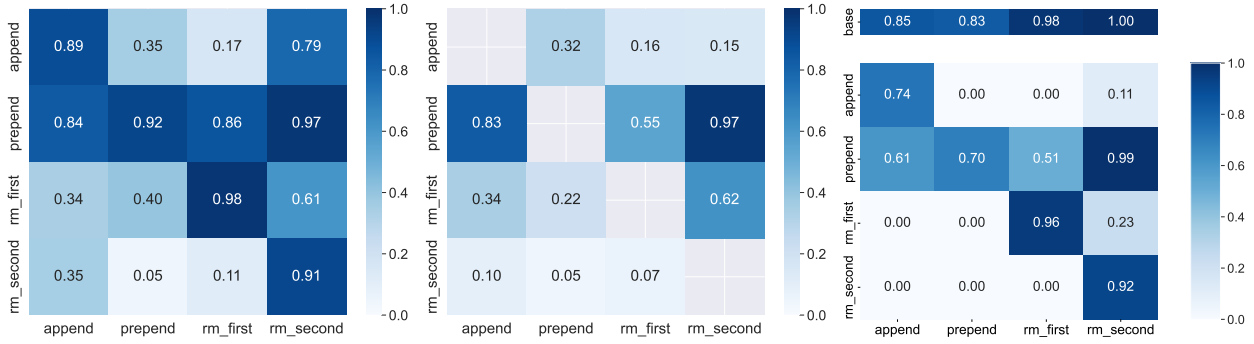
Figure 13 illustrates the IoU and IoM between all circuit-pairs under zero ablation. The observed patterns are consistent with those identified under mean ablation (Figure 4), where unary circuits tend to exhibit higher node overlap compared to binary circuits.

C.3 COMPARISON BETWEEN MEAN AND ZERO ABLATION

Figure 14 illustrates the overlap between the mean and zero ablated circuits for identical subtasks. As discussed in Section 6, we find significant differences in the circuits identified, specifically for unary circuits that tend to engage a smaller portion of the base model's activation space (refer to Figure 15 for comparison).

(a) F_T across all token positions. (b) F_T for selected token positions. (c) Accuracy.

Figure 9: Task faithfulness performance F_T and accuracy for the **unary** tasks for the **zero** ablated circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. For task faithfulness with respect to positions where the ground truth tokens differ between the circuit and the evaluation task, the diagonal is omitted, as there are no applicable token positions for comparison.

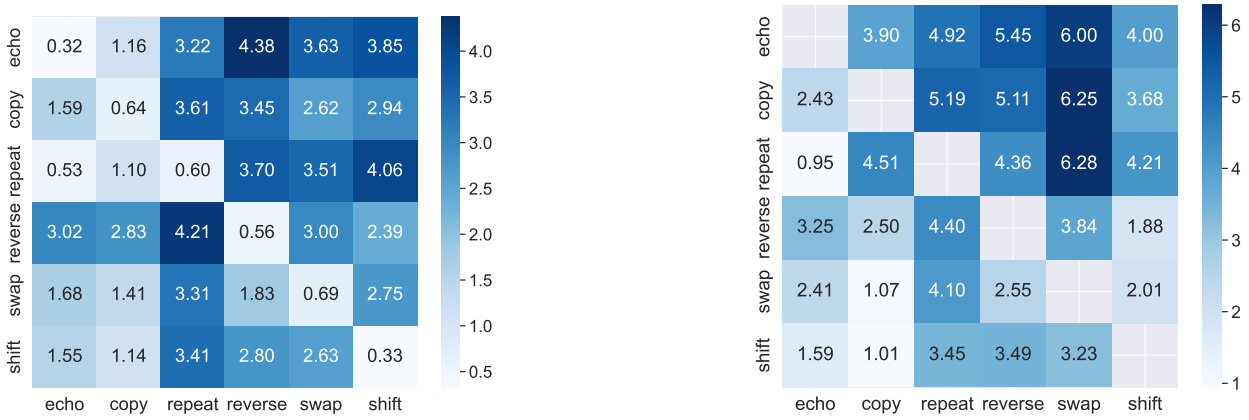


(a) F_T for all token positions. (b) F_T for selected token positions. (c) Accuracy.

Figure 10: Task faithfulness performance F_T and accuracy for the **binary** tasks for the **zero** ablated circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. For task faithfulness with respect to positions where the ground truth tokens differ between the circuit and the evaluation task, the diagonal is omitted, as there are no applicable token positions for comparison.

C.4 Global Sparsity

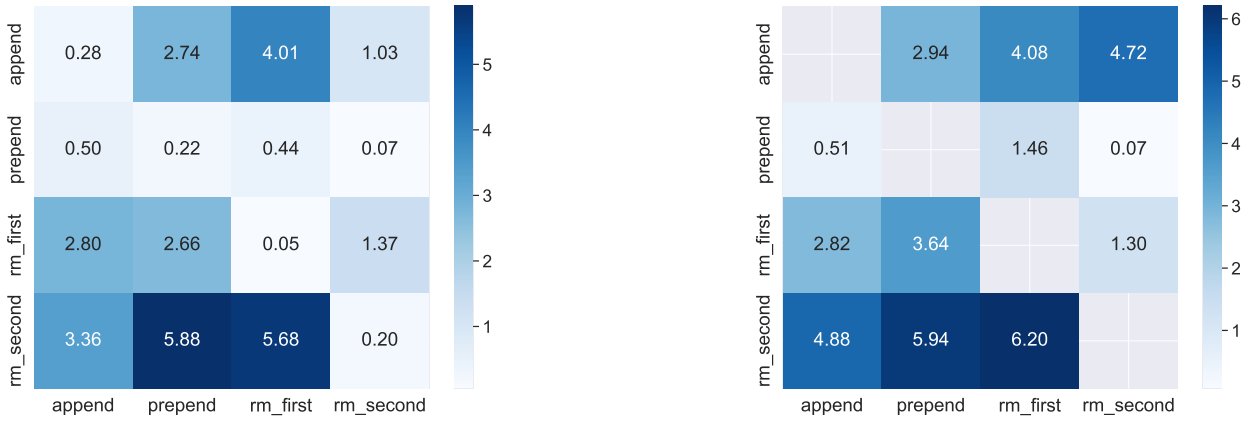
In Figure 15 we present the global sparsity of the circuits for both mean and zero ablation. As touched upon in Section 6, we observe that while the two ablation strategies identify circuits of comparable sizes, they do not necessarily yield identical circuits. Additionally, the ordering of circuits in terms of remaining activations is nearly the same for both methods, with the only notable difference being the internal ordering of `swap` and `reverse`.



(a) All token positions.

(b) Differing token positions.

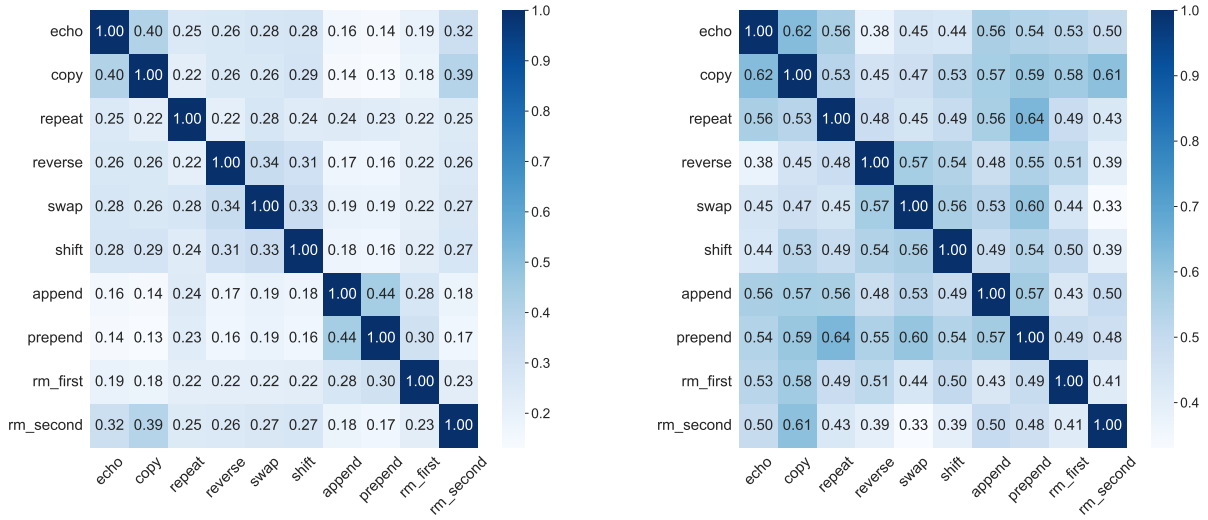
Figure 11: Task faithfulness measured via $\mathrm{D}_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$ for the **zero-ablated unary** circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.



(a) All token positions.

(b) Differing token positions.

Figure 12: Task faithfulness measured via $\mathrm{D}_{KL}\left(\boldsymbol{y}^{\boldsymbol{m}} \parallel \boldsymbol{y}^{\mathcal{M}}\right)$ for the **zero-ablated binary** circuits. The y-axis corresponds to the circuit, while the x-axis represents the evaluation task. When we only evaluate selected positions, we omit the diagonal, as there are no applicable tokens for comparison.



(a) Intersection over Union (IoU). (b) Intersection over Minimum (IoM).

Figure 13: Node overlap for circuits identified via zero ablation.

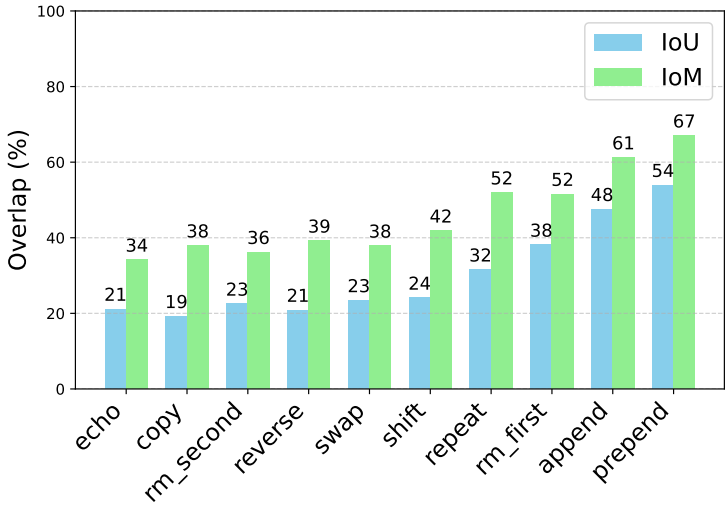
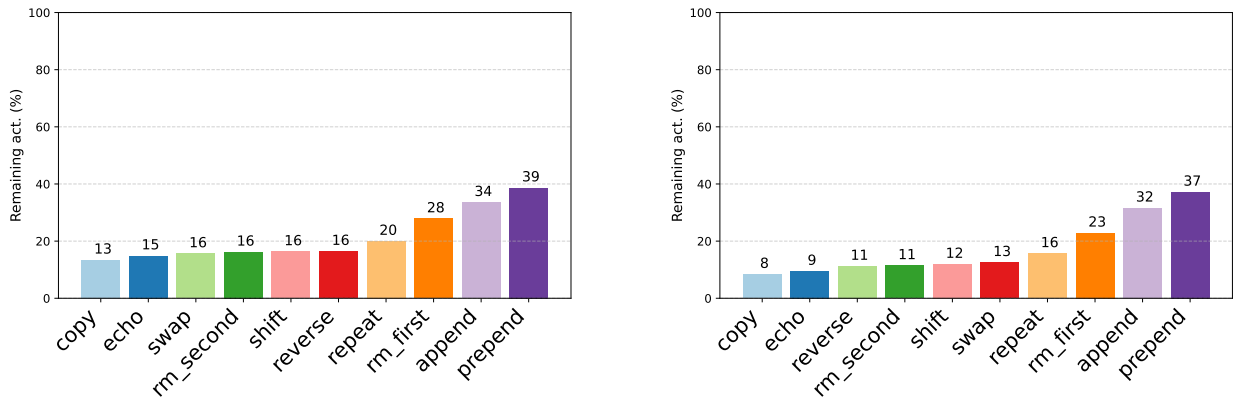


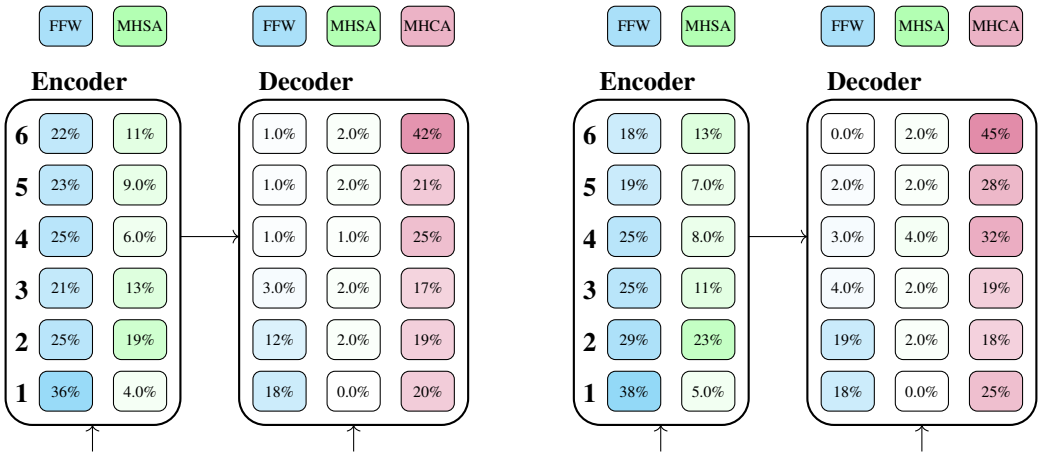
Figure 14: The overlap between mean and zero ablated circuits for all subtasks.



(a) Mean ablation.

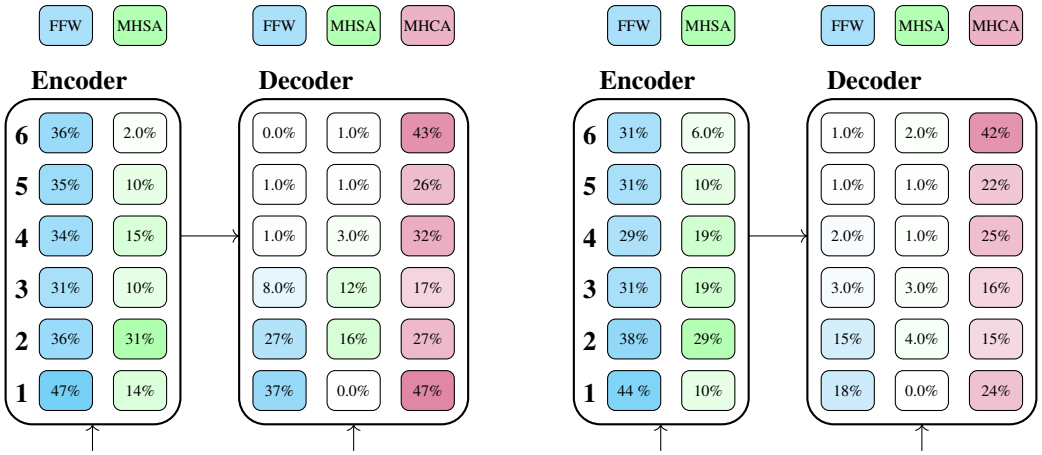
(b) Zero ablation.

Figure 15: Overall remaining nodes after activation pruning.



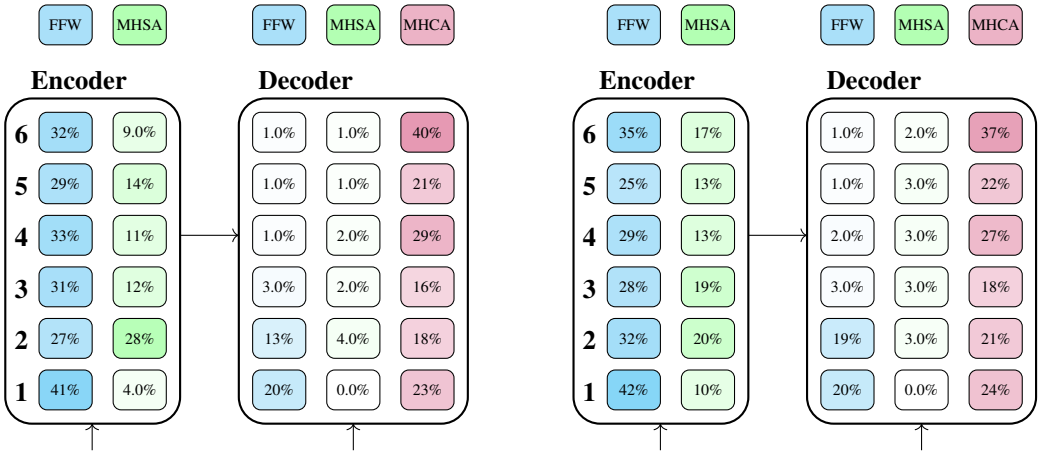
(a) The local sparsity of the `copy` circuit.

(b) The local sparsity of the `echo` circuit.



(c) The local sparsity of the `repeat` circuit.

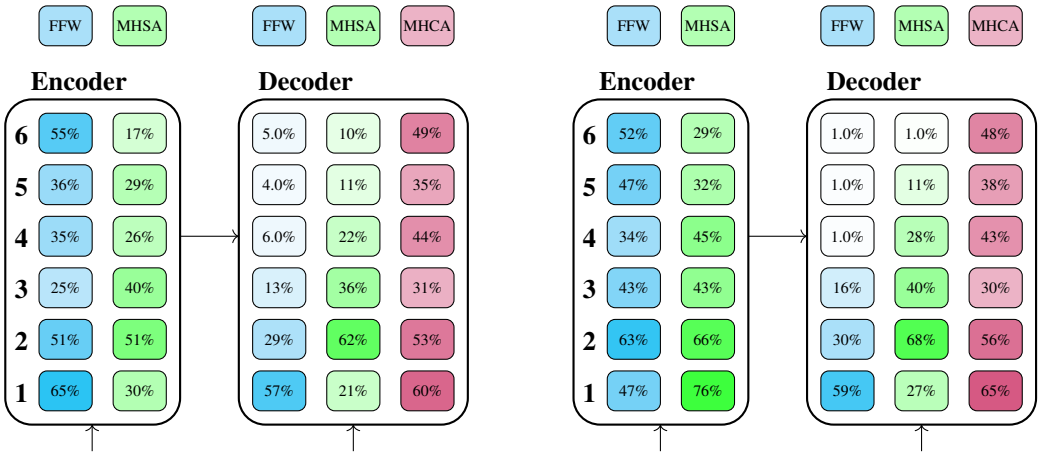
(d) The local sparsity of the `reverse` circuit.



(e) The local sparsity of the `swap` circuit.

(f) The local sparsity of the `shift` circuit.

Figure 16: The local sparsity of the **unary** circuits achieved via mean ablation. Considered are feed-forward (FFW), multi-head self-attention (MHSA), and multi-head cross-attention (MHCA) modules for each layer.



(a) The local sparsity of the `append` circuit.

(b) The local sparsity of the `prepend` circuit.

(c) The local sparsity of the `remove_first` circuit.

(d) The local sparsity of the `remove_second` circuit.

Figure 17: The local sparsity of the **binary** circuits achieved via mean ablation. Considered are feed-forward (FFW), multi-head self-attention (MHSA), and multi-head cross-attention (MHCA) modules for each layer.