Dynamic Granularity in the Wild: Differentiable Sheaf Discovery with Joint Computation Graph Pruning

Anonymous ACL submission

Abstract

In this paper, we introduce DiscoGP, a novel framework for extracting self-contained modular units, or *sheaves*, within LMs. These sheaves correspond to the models' impressive zero-shot performance across a variety of tasks. Our DiscoGP framework extends the concept of functional *circuits*, widely explored in interpretability research, by introducing *sheaves* — subsets of connection edges and weight parameters in an LM's computation graph — as interpretation units. Our framework identifies these sheaves through a differentiable pruning algorithm that operates on both the computation graph's edge connections and the model's weight parameters. This process reduces the LM to a sparse skeleton while preserving its core capabilities. Experimental results demonstrate that across a range of linguistic and reasoning tasks, DiscoGP extracts sheaves that preserve 93-100% of the model's task performance while comprising only 1-7% of the original weights and connections. Furthermore, our analysis reveals that, compared to previously identified LM circuits, the sheaves discovered by DiscoGP exhibit superior modularity and functional fidelity. Extending our method to the neuron level also unveiled novel insights into the inner workings of LLMs.[1]

1 Introduction

Transformer language models (LMs; Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; Raffel et al., 2020; OpenAI, 2023; Touvron et al., 2023) have demonstrated their incredible capabilities in solving various natural language tasks across different fields. Yet, the exact mechanisms by which these models solve tasks remain largely unknown. Researchers in the field of interpretability therefore aim to provide human-understandable explanations of the computational mechanisms of these "black-boxed" LMs. Should the interpretation of LMs become possible, it could lead to the improvement of LMs with better controllability and performance, and even germinate the next generation of explainable artificial intelligence (XAI) systems.

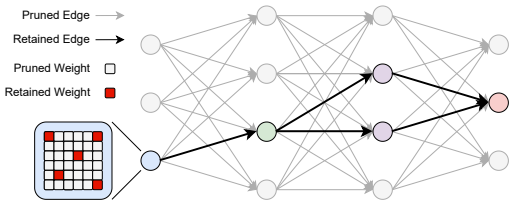


Figure 1: Illustration of DiscoGP. By combining edge and weight parameter pruning, DiscoGP enables better performance and a neuron-level granularity.

Now, a nascent "circuit"-based framework aiming at explaining this process has emerged and provides the most convincing explanation of LM behaviours to date. Several approaches within this framework decompose the computation process of an LM into a directed acyclic graph (DAG) and aim to identify the subset of model components and connections (information flow) that correspond to specific model behaviours, phenomena, and processes. Initially, these circuits were identified manually using various activation or attention patching methods (Wang et al., 2022), and ACDC (Conmy et al., 2023) automated the circuit discovery process. Since then, several follow-up attempts have been proposed to further advance the state-of-the-art circuit discovery methods.

The term *circuit*, however, is used to refer to several distinct concepts, even within the LM interpretability community. We provide a survey of the nomenclature (§2) and clarify our intention to interpret the model. With a thorough and rigorous definition of the computation graph, edge pruning, and weight pruning, we introduce the new concept of a *sheaf*: a subset of connection edges and weight parameters in the computation graph that, when executed in isolation, can preserve the original model's behaviour. Simply put, we seek to

[1] Code for DiscoGP will be made publicly available online. *GitHub URL withdrawn for submission.*

identify the interpretable sheaf of model components within the LM "haystack." This novel sheaf discovery task fills a gap in current mechanistic interpretability research and complements ongoing automatic circuit discovery efforts. It enables us to identify a self-contained collection of model units that can perform a particular LM function in isolation. These sheaves offer a unique opportunity to manipulate self-contained units and gain novel insights into LMs' internal workings.

Moreover, prior automatic sheaf and circuit discovery methods share a crucial limitation: the computational power required is prohibitively large because the number of edges in the computation graph grows quadratically ($O(n^2)$) with the number of model components. This prohibits researchers to conduct sheaf and circuit discovery at the neuron level. Nonetheless, extending circuit analysis to the neuron level is particularly valuable, as recent investigations into the properties of MLP neurons (Geva et al., 2021; Dai et al., 2022; Meng et al., 2022; Niu et al., 2024; Hong et al., 2024) indicate a high degree of idiosyncrasy in the type of information and function associated with each neuron. For instance, Niu et al. (2024) and Dai et al. (2022) showed that modifying just a few neurons, or even a single neuron, can lead to substantial changes in the model's behaviour. Therefore, refining the granularity of interpretation to the neuron level could reveal important novel insights.

Therefore, we introduce the **Di**fferentiable **S**heaf **Disco**very with Joint **Co**mputation **G**raph **P**runing (DiscoGP) framework, a novel method that addresses the granularity and scaling problem by applying joint edge and weight parameter pruning with differentiable masking. While the computation graph is still defined at the relatively coarse level of attention heads and MLPs, as in other circuit discovery methods, DiscoGP extends this approach by introducing weight pruning within each individual computation graph node to enable finer, neuron-level interpretability.

DiscoGP achieves state-of-the-art performance in sheaf detection: it identifies the sheaves for a wide range of tasks with the fewest edges and weight parameters while maintaining near-perfect performance compared to the original model's performance. By extending the granularity to the neuron level, we unveil several critical insights into the model that was previously unavailable.

Contribution: We begin with a proper definition of **sheaves**, and provide a survey that aim at clarifying the various different usage of the term *circuit* in relevant literature (§2). Then, we introduce **DiscoGP**, a novel sheaf discovery framework with joint pruning of weight parameters and computation graph edges that enables individual neuron level granularity (§3). Using DiscoGP, we can obtain sheaves across a wide range of tasks that are **sparser and more faithful** to the original model compared to common baselines.

2 Sheaves and Circuits

In this section, we present a comprehensive definition of the main task of *sheaf discovery*, and discuss its similarities and differences compared to the broad range of tasks often referred to as "circuit discovery" in the literature, as the term is used inconsistently and can sometimes cause confusion. In short, we aim to identify the self-contained set of LM components that, when executed in isolation, reproduce some behaviour or capability exhibited by the full model. We start with a survey of the different definitions of circuit discovery (§2.1), and then introduce our sheaf-based framework by framing weight pruning (§2.2) and edge pruning (§2.3).

2.1 Survey: Circuits and Circuit Discovery

Circuit The term "circuit" has various meanings within the LM interpretability community, depending on the context. Nanda (2022) described it as "a fairly fuzzy and poorly defined term" that roughly refers to "the sub part [sic] of a model that does some understandable computation to produce some interpretable features from prior interpretable features." Olah et al. (2020) considered circuits as a set of features and the weighted connections between them. Elhage et al. (2021) used the term "circuit" to refer to the separable parts of the computation process within each attention head. Because the computation of a transformer model can generally be considered linear, Elhage et al. (2021) argued that the computation of the query and key matrices and the output and value matrices can be considered as two largely independent circuits: the QK circuit and the OV circuit. More recently, work in the field typically decomposes an LM into its "building blocks" and considers the collection of these blocks and a subset of their connections as a circuit; however, what constitutes building blocks may differ from work to work. Wang et al. (2022) referred to a circuit as the collection of attention heads, while ACDC used the term "circuit" to refer

to the subset of edges between attention heads and MLPs in the computation graph.

Circuit Discovery The task of identifying the aforementioned circuits in pre-trained transformer LMs is called *circuit discovery*. Early studies typically search for circuits manually on simple tasks such as rudimentary anaphora resolution (Wang et al., 2022) or simple arithmetic reasoning (Hanna et al., 2023), using a combination of interpretability tools including causal interventions (Vig et al., 2020; Meng et al., 2022) and logit lens (Geva et al., 2022, 2023; Yu et al., 2024a). More recently, ACDC (Conmy et al., 2023) automated the circuit discovery process. Specifically, they used the activation patching technique (Goldowsky-Dill et al., 2023; Zhang and Nanda, 2023), or its approximations (Nanda, 2023), to assess a connection edge's importance by first knocking it out and observing its effect on the model's final output. Beginning at the output node and proceeding in reverse topological order, they evaluate the effect of removing each of the node's incoming edges individually. If the removal of an edge has a greater effect than a predetermined threshold (τ), the edge is included in the circuit; otherwise, it is pruned. Syed et al. (2024) extended ACDC and applied attribution patching to achieve improved results.

Recent work has also explored other notions of a circuit, such as formulating circuits as collections of human-interpretable neural activation features (Huben et al., 2023; Marks et al., 2024; Yu et al., 2024b), or as distributed neural representations of proposed symbolic algorithms (Geiger et al., 2021; Wu et al., 2023). Most automated circuit discovery studies evaluate their methods based on their structural overlap with previously discovered or manually hardwired circuits (Conmy et al., 2023; Syed et al., 2024). While we concur with recent critiques of this evaluation metric (Hanna et al., 2024), we argue that the functional fidelity[2] metric (measuring how well the circuit reproduces the original model's performance) is a more appropriate criterion for this task.

2.2 Weight Pruning

Weight pruning is a technique widely used in the model interpretability community to identify subnetworks (a subset of a model's weight parameters) associated with specific functions of a neural network (Cao et al., 2021; Csordás et al., 2021; Zhang et al., 2021; Guo et al., 2021; De Cao et al., 2022). More recently, Lepori et al. (2023) extended this work to transformer-based language models. Figure 2a provides an overview of weight pruning. This line of research was encouraged by Frankle and Carbin's (2018) the Lottery Ticket Hypothesis, which states that it is possible to identify a much smaller subnetwork within a model. When this subnetwork is trained from scratch with a similar computational budget, it can achieve performance comparable to that of the original model. Using the *continuous sparsification* method (Figure 2a), Savarese et al. (2020) demonstrated that a subnetwork can be directly extracted from a neural network that maintains task performance without the need for retraining, as originally suggested in the hypothesis. The method is also referred to as *differentiable masking* (De Cao et al., 2022).

2.3 Computation Graph and Edge Pruning

Computation Graph Elhage et al. (2021) introduced the concept of the *residual stream*, providing a clear and concise view of the computation within a transformer block. Each block consists of an attention module followed by an MLP module. Let $x_0^{(i)}$ be the input to the i-th transformer block, with $H^{(i)}$ representing the set of attention heads and f_i denoting the MLP module, we can write the computation of a Transformer layer (i-th layer) as:

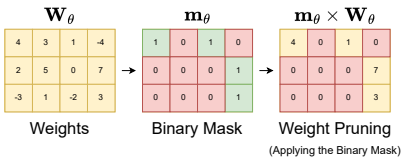
$$x_{i+1} = \overbrace{x_i + \sum_{h \in H^{(i)}} h(x_i)}^{x_i^{\text{mid}}} + f_i(x_i^{\text{mid}}). \quad (1)$$

To demonstrate the concept of a computation graph, let us consider a simple one layer transformer model with the input embedding x_0. We can "unroll" the residual stream (Nanda and Bloom, 2022) as shown in (2). From the result, we can tell that final output of the transformer block consists of 4 input sources: the original word embedding input x_0, the output of the two attention heads $h_1(x_0)$ and $h_2(x_0)$, as well as the output of the MLP module $f_1(x_0 + h_1(x_0) + h_2(x_0))$.

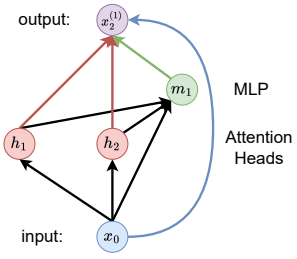
$$\begin{aligned} x_1^{(1)} &= x_0 + h_1(x_0) + h_2(x_0) \\ x_2^{(1)} &= x_1^{(1)} + f_1(x_1^{(1)}) = x_0 + h_1(x_0) + \\ &\quad h_2(x_0) + f_1(x_0 + h_1(x_0) + h_2(x_0)). \end{aligned} \quad (2)$$

From the unrolled residual stream, we can understand how information flows within the transformer block. Using (2) as an example, the output ($x_2^{(1)}$) is

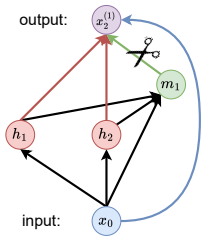
[2]The term is also referred to as functional *faithfulness*.



(a) Weight pruning is performed by directly setting the value of a weight parameter to zero. In practice, previous work has shown that differentiable masking is one of the most effective methods for weight pruning. Specifically, a differentiable binary mask is applied to each weight parameter, and the algorithm optimises the value of these masks to identify the subnetwork that achieves the best task performance.



(b) The computation graph of the single-layer transformer block example. This graphical representation corresponds to the unrolling of the residual stream in (2). The top-level terms are colour-coded to match those in (2).



$$x_2^{(1)} = x_0 + h_1(x_0) + h_2(x_0) + \cancel{m_1(x_0 + h_1(x_0) + h_2(x_0))}$$

(c) "Cutting off" an edge is equivalent to removing a term from the residual stream in the zero-ablation setting. In other settings (mean or interchange), the term is replaced with a new value.

Figure 2: Illustration of the formulation of sheaf discovery: computation graph, weight pruning and edge pruning.

derived from the outputs of the two attention heads (h_1, h_1), the MLP (f_1) output, and the original input embedding (x_0). The attention heads only take x_0 as input, while f_1 receives both the outputs of the attention heads and x_0. Based on this information flow, we can construct a computation graph as shown in Figure 2b.

Edge Pruning The introduction of computation graphs allows us to analyse the impact of information flow between model components. By pruning[3] a connection edge, we can examine how this modification affects the model's final output, revealing the importance of that specific information flow. As shown in Figure 2c, the pruning of an edge is equivalent to the removal of a term in the unrolled residual stream, which can be achieved either through greedily applying causal mediation methods (Vig et al., 2020; Finlayson et al., 2021; Meng et al., 2022) to identify important edges (Conmy et al., 2023), or by leveraging gradient-based techniques to mask out unessential component connections (Bhaskar et al., 2024).

3 DiscoGP: Sheaf Discovery

Weight pruning and circuit pruning are not mutually exclusive, so why not apply both? Here, we introduce the term "sheaf" to describe the intersection of circuit pruning (edge pruning) and subnetwork pruning (weight pruning). Let $G = \{E, V\}$ represent the computation graph, and let $\boldsymbol{\Theta}$ denote the set of all parameters of the language model. The task of identifying a sheaf involves searching for two binary masks, $\boldsymbol{m} = (\boldsymbol{m_\theta}, \boldsymbol{m}_E) \in \{0,1\}^{|\boldsymbol{\theta}|+|E|}$, which correspond to the pruned weights and edges, respectively. Similar to prior weight pruning approaches, DiscoGP makes both binary masks differentiable, enabling the search for a globally optimal solution across weight and edge pruning. This section outlines the sheaf discovery task and the DiscoGP joint edge and weight pruning algorithm.

In summary, sheaf discovery has three steps:

1. For an LM capability, define a task corresponding to the capability by constructing a dataset;
2. Search for a sheaf (a collection of edges and weight parameters) corresponding to the dataset;
3. Evaluate the sheaf to determine whether the model can still perform the task using only the sheaf, i.e., with all other components or connections turned off.

These three steps of sheaf discovery share some superficial resemblance to the three steps of the "automatic circuit discovery workflow" proposed by Conmy et al. (2023). They argue that researchers should "perform an extensive and iterative series of patching experiments with the goal of removing as many unnecessary components and connections from the model as possible" (Conmy et al., 2023). Our framework differs in two key aspects: (1) we do not impose any restrictions on patching-based approaches; and (2) we aim to identify sheaves that are self-contained — in other words, while ACDC's goal is to identify the most salient components and edges, it does not consider whether the resulting "circuits" can perform the task by itself.

3.1 Joint Weight and Edge Pruning

Similar to previous work on differentiable mask learning (Louizos et al., 2018; Csordás et al., 2021;

[3] Also referred to as *knockout*, *cut-off*, or *ablation*.

Cao et al., 2021; De Cao et al., 2022; Bayazit et al., 2023), DiscoGP models each mask $m_i \in \boldsymbol{m}$ as a random variable, parameterised by a hard-concrete or gumbel-sigmoid distribution. We first compute a continuous score $s_i \in [0, 1]$:

$$s_i = \sigma\Big(\frac{l_i - \log \frac{\log \mathcal{U}_1}{\log \mathcal{U}_2}}{\tau}\Big), \tag{3}$$

where $\tau \in (0, \inf)$ is a temperature hyperparameter, l_i is a learnable logit parameter of a sigmoid distribution $\sigma(\cdot)$, and $\mathcal{U}_1, \mathcal{U}_2 \sim \text{Uniform}(0, 1)$ are random variables drawn from a uniform distribution. We then use the straight-through estimator (Bengio et al., 2013) to convert the sampled s_i into a binary mask variable:

$$m_i = [\mathbb{1}_{s_i > 0.5} - s_i]_{\text{detach}} + s_i, \tag{4}$$

where $\mathbb{1}$ represents the indicator function, and $[\cdot]_{\text{detach}}$ is an operator that blocks gradient flow during backpropagation. This approach makes the binary mask m_i a differentiable function of the logit l_i, allowing it to be optimised through backpropagation for specific objectives.

Sheaf Searching Objectives Given a task dataset $\mathcal{D} = \{\mathbf{x}, \hat{\mathbf{y}}\}$, where $\mathbf{x}$ represents the input and $\hat{\boldsymbol{y}}$ is the output of the original model, our aim is to identify a set of masks $\boldsymbol{m}$ on weights and edges, such that the pruned sheaf produces results as close to the original model as possible. To achieve this, we define the **functional fidelity loss** as the negative log-likelihood of the original model's predicted label in the output distribution of the pruned circuit:

$$\mathcal{L}_{\text{fidelity}} = -\sum_i \log p_{\boldsymbol{m}}(\hat{y}_i | x_i). \tag{5}$$

Moreover, we want the sheaf to contain as much function-specific weight and edges as possible. In other words, when the detected sheaf is removed from the original model, the remaining computational graph should perform at near-random levels on D. Let $\tilde{\boldsymbol{m}} = 1 - \boldsymbol{m}$ denote the reverse mask of $\boldsymbol{m}$, we define the **completeness loss** as the cross-entropy between the output distribution of the complementary sheaf and a uniform distribution over the label space $\{y_k\}_{k=1}^K$:

$$\mathcal{L}_{\text{complete}} = -\sum_i \sum_{k=1}^{K} \frac{1}{K} \log p_{\boldsymbol{m}}(y_k | x_i). \tag{6}$$

Dataset	Example Prompt	Correct	Incorrect
BLiMP	`Raymond is selling this __`	`sketch`	`sketches`
IOI	`When Mary and John went to the store, John gave a drink to __`	`Mary`	`John`
OQA	`The capital city of Canada is __`	`Ottawa`	*not unique

Table 1: An overview of the tasks and datasets.

Lastly, we want the sheaf to be as sparse as possible. Therefore, we minimize the **sparsity loss**:

$$\begin{aligned}\mathcal{L}_{\text{sparse}} &= \mathcal{L}_{\text{sparse}-\boldsymbol{\theta}} + \mathcal{L}_{\text{sparse}-E} \\ &= \frac{1}{|\boldsymbol{m}_{\boldsymbol{\theta}}|} \sum_{i=1}^{|\boldsymbol{m}_{\boldsymbol{\theta}}|} \sigma(l_i) + \frac{1}{|\boldsymbol{m}_E|} \sum_{i=1}^{|\boldsymbol{m}_E|} \sigma(l_i).\end{aligned} \tag{7}$$

The final objective function is then comprised of a weighted mixture of the three loss terms:

$$\mathcal{L}_{\text{GP}} = \mathcal{L}_{\text{fidelity}} + \lambda_c \mathcal{L}_{\text{complete}} + \lambda_s \mathcal{L}_{\text{sparse}}, \tag{8}$$

where λ_c, λ_s are hyperparameters that regulate relative loss importance.

DiscoGP Implementation Details Due to page limitations, other optimisation techniques we implemented, including post-hoc sheaf pruning and split QKV pruning, are introduced in Appendix A.

4 Experimental Setup

Evaluation: We evaluate DiscoGP and the baselines across three tasks (Table 1): syntactic agreement from the **BLiMP** corpus (Warstadt et al., 2020), the indirect object identification (**IOI**) task introduced by Wang et al. (2022), and factual information from open-domain question answering (OQA) with the **PARAREL** (Elazar et al., 2021) dataset. These three tasks provide a comprehensive coverage of syntactic, semantic and factual information. See Appendix B for more information.

Metric-wise, we report the **functional fidelity**: this includes the sheaf's accuracy and the KL divergence of the sheaf's output (sheaf accuracy refers to the task accuracy when all pruned components are "turned off" and sheaf KL divergence is measured between the sheaf's output and that of the original model). We also report completeness or the **complement sheaf accuracy** (i.e., the accuracy when the sheaf is turned off and all other model components are kept on), as well as **sparsity** (both edge and weight sparsity). These evaluation metrics follows the typical fidelity, completeness and sparisty scheme used by other mechanistic interpretability work (Wang et al., 2022; Conmy et al., 2023; Syed et al., 2024; Bhaskar et al., 2024).

Task	Discovery Method	Sheaf Acc. (%) (higher is better)	KL Div. (lower is better)	Comp. Acc. (%) (random* is better)	Weight Density (%) (lower is better)	Edge Density (%) (lower is better)
anaphor gender agr. (AGA)	ACDC	83.3	0.121	42.7	100	6.48
	EAP	89.3	0.091	53.9	100	4.88
	Edge Pruning	88.4	0.137	49.7	100	6.62
	Weight Pruning	97.1	0.078	50.2	3.01	100
	DiscoGP (Ours)	**98.5**	**0.074**	49.9	**1.58**	**3.88**
anaphor number agr. (ANA)	ACDC	81.0	0.250	67.0	100	6.26
	EAP	95.3	0.049	56.3	100	8.66
	Edge Pruning	87.9	0.178	39.3	100	2.78
	Weight Pruning	97.7	0.076	40.3	2.79	100
	DiscoGP (Ours)	**99.7**	**0.043**	39.2	**1.36**	**1.94**
det. noun agr. 1 (DNA)	ACDC	85.3	0.129	46.3	100	7.35
	EAP	85.7	0.138	40.6	100	9.83
	Edge Pruning	83.7	0.114	59.3	100	2.27
	Weight Pruning	95.3	0.099	53.0	0.280	100
	DiscoGP (Ours)	**95.3**	**0.098**	51.7	**0.187**	**1.92**
det. noun irr. 1 (DNA i)	ACDC	62.7	0.419	39.3	100	6.61
	EAP	60.0	0.434	38.3	100	8.92
	Edge Pruning	67.1	0.374	48.0	100	2.46
	Weight Pruning	94.3	0.103	53.6	0.263	100
	DiscoGP (Ours)	**95.8**	**0.102**	47.2	**0.244**	**1.68**
det. noun adj. 1 (DNA a)	ACDC	82.4	0.169	52.3	100	7.04
	EAP	83.5	0.153	45.7	100	9.90
	Edge Pruning	50.3	0.412	47.6	100	7.14
	Weight Pruning	94.7	0.136	49.9	0.565	100
	DiscoGP (Ours)	**95.5**	**0.118**	45.3	**0.520**	**5.71**
det. noun adj. irr. 1 (DNA ai)	ACDC	50.2	0.120	41.4	100	9.46
	EAP	60.7	0.128	44.7	100	6.89
	Edge Pruning	56.3	0.348	47.8	100	12.9
	Weight Pruning	94.6	0.127	49.9	0.569	100
	DiscoGP (Ours)	**95.1**	**0.118**	45.3	**0.496**	**6.22**
IOI	ACDC	51.6	0.730	50.6	100	2.45
	EAP	58.3	0.756	55.2	100	3.48
	Edge Pruning	100	0.032	49.9	100	2.97
	Weight Pruning	98.4	0.043	57.5	1.87	100
	DiscoGP (Ours)	**100**	**0.020**	49.2	**1.79**	**2.03**
PARAREL Average†	ACDC	1.0	0.379	0.6	100	5.35
	EAP	0.9	0.341	0.6	100	5.92
	Edge Pruning	90.4	0.039	0.7	100	2.97
	Weight Pruning	91.8	0.032	0.8	2.83	100
	DiscoGP (Ours)	**93.1**	**0.023**	0.62	**2.77**	**2.91**

Table 2: Sheaf Discovery Performance Comparison. DiscoGP achieves the best performance across all tasks, using the fewest weight parameters and edges. The best-performing methods are highlighted in **bold**. *: For complement sheaf accuracy, successful searches are expected to yield random performance. Therefore, scores close to random indicate good performance, and direct comparison of complement scores is not meaningful. BLiMP and IOI's expected random performance is 50%, and PARAREL's expected random performance is 0%. †: Due to page limits, the full PARAREL results are listed in Appendix D. The PARAREL results support the same findings.

LM Selection: We compare to the baseline methods using GPT-2 base (small) model, as it is the only model supported by the original implementation of every baseline method.

Baseline Methods: We compare DiscoGP with all the major prior automatic sheaf discovery methods. We categorize the methods into (1) threshold-based greedy search algorithms that includes ACDC (Conmy et al., 2023) and EAP (Syed et al., 2024); and (2) differentiable-masking-based algorithms including the weight pruning (WP) methods (Louizos et al., 2018; Cao et al., 2021; Sanh et al., 2020; De Cao et al., 2022), edge pruning (EP) method (Bhaskar et al., 2024), and our novel joint pruning method. See Appendix C for our reproduction details.

5 Experiment Results

Table 2 shows the results of DiscoGP compared to the baseline methods. Due to page limits, full results for the OQA task are shown in Appendix D; the breakdown supports the same findings. For each experiment, we run the sheaf dis-

covery method five times and report average performance. GPT-2 achieves near-perfect performance on all BLiMP and IOI tasks, so we conduct our experiments on the full datasets. However, GPT-2 performs worse on the OQA PARAREL tasks, so we run experiments only on data samples where the original model answers the question correctly, discarding prompts where it fails, as it is unclear whether searching for a sheaf over a function the LM does not have would yield meaningful results.

Overall, we can find that DiscoGP outperforms all baseline sheaf discovery methods. It achieves the highest functional fidelity — either measured in task accuracy or KL divergence — compared to other baselines while using the fewest weight parameters or connection edges.

Discussion: Greedy threshold-based methods may not be suitable for sheaf discovery. Interestingly, we observe that the performance of greedy threshold-based methods (ACDC and EAP) is less stable across tasks. For the more complex tasks, these methods reach near-random performance when given the same sparsity budget as DiscoGP. This is especially true for the PARAREL tasks (Appendix D). These surprising results, *pace* ACDC and EAP, do not argue for the validity of their respective methods, but rather highlight the difference between sheaf discovery and their patching-based automatic circuit discovery.

Now, we want to take the opportunity to elaborate on the **difference between sheaf discovery and automatic circuit discovery**. First and foremost, the two tasks differ in their goals and motivations. Let us revisit the famous example studied by Wang et al. (2022): "`When Mary and John went to the store, John gave a drink to` __", where `Mary` is the correct answer and `John` is the incorrect one. The automatic circuit discovery task aims to identify all the important connection edges and components that, when perturbed, cause the greatest change to the final output, and potentially steering the model away from responding `Mary` to `John`. Our results show that simply taking the collection of these important components does not always yield a self-contained mechanism that can perform the task in isolation. Sheaf discovery, on the other hand, aims to capture and identify that self-contained mechanism (the sheaf) and fill this research gap.

Therefore, it is appropriate for ACDC and EAP to apply ablation-based methods for automatic circuit discovery. These include *mean ablation*, which sets the activation to the average output across a reference distribution obtained by running a sample dataset through the model, and *interchange ablation*, which replaces the activation with its value from a corrupted input, created by modifying specific input tokens. However, these ablation methods may not be suitable for sheaf discovery, as they still retain a large amount of task-related information (Table 3). This observation is supported by recent work (Adolfi et al., 2025; Shi et al., 2024) showing that these ablation- and patching-based methods may not achieve optimal functional fidelity.

Task	Clean-Ablated Edge similarity		
	Mean	Interchange	Random
Agreement	0.878	0.907	0.582
IOI	0.943	0.996	0.597
OQA PARAREL	0.951	0.960	0.556

Table 3: Average cosine similarity between clean and corrupted edge hidden representations across three datasets. Mean and interchange ablations do not substantially affect the models' overall performance.

Task	*Evaluation Tasks*					
	AGA	ANA	DNA	DNA i	DNA a	DNA ai
AGA	-	98.0	99.7	99.7	91.9	94.8
ANA	94.0	-	99.7	100	91.9	92.0
DNA	92.3	86.3	-	93.0	90.3	91.2
DNA i	91.3	80.3	93.7	-	94.4	93.1
DNA a	93.0	94.6	94.2	90.5	-	94.9
DNA ia	91.7	90.1	92.3	94.5	94.2	-
Orig.	99.0	100	94.7	95.3	96.0	95.7

Table 4: Composing sheaves can largely preserve functional performance. Each entry shows the performance (accuracy in %) of a composed circuit (row + column) evaluated on the task associated with the column. For example, the value in column AGA, row ANA shows the performance of the composed circuit (ANA + AGA) on the AGA task. Original (non-composed) sheaf performance is listed in the final row for reference.

6 Analyses and Findings

Finding 1: Sheaves identified by DiscoGP can be composed while preserving the functionality. We find that *functional composition of sheaves* is possible under the DiscoGP framework. That is, suppose we have two sheaves that perform task A and task B, respectively. Simply composing their masks, $m = m_A \cup m_B$, can yield a new sheaf that performs both tasks with largely the same performance. Table 4 shows the performance of such compositions across different BLiMP paradigms.

Overall, we observe good composition performance, with the composed sheaves' accuracies reaching 80-100% across all BLiMP tasks. To

Sheaf 1	Sheaf 2	Edge Overlap	Weight Overlap
AGA	DNA	14.86% (251)	2.69% (8020)
ANA	DNA	16.19% (277)	1.12% (14816)
ANA	AGA	18.32% (266)	0.91% (17693)
DNA	DNA irr	21.07% (317)	4.72% (69364)
DNA	DNA adj	18.46% (332)	4.96% (74782)
DNA	DNA irr adj	18.24% (323)	6.06% (96727)

Table 5: Sheaf overlap across different BLiMP tasks. The results indicate a trend where similar tasks exhibit higher sheaf overlaps. The overlap percentage are followed by the exact number of overlaps in brackets.

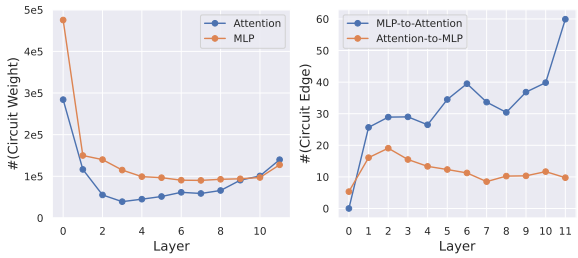


Figure 3: **Left**: Number of unmasked MLP and attention weights at each layer of the capital city OQA sheaf. **Right**: Number of edges ending at each layer from preceeding MLPs to current-layer attention heads and from preceeding attention heads to current-layer MLP.

the best of our knowledge, our result is the first successful sheaf or circuit composition *in the wild*. Mondorf et al. (2025) studied circuit composition, but their experiments were limited to synthetic toy models generated using Tracr (Lindner et al., 2023). We show that sheaves in real-world transformer models can also be composed to achieve task unity. This suggests that some degree of modularity has emerged in LMs through the pre-training process. We hope this finding will motivate future work on modularity and sheaf composition.

Finding 2: Sheaf similarity reflects functional similarity. Table 5 illustrates the overlap levels between different sheaves. The overlap percentages are calculated by dividing the number of overlap cases by the size of the logical union of the two masks. In this analysis, we only considered the agreement tasks as their task similarity is easier to perceive. BLiMP offers several variants of the DNA tasks, and we observed a relatively high level of sheaf overlap in terms of weights and edges among them. The ANA and AGA tasks exhibit greater similarity to each other compared to DNA tasks, as ANA and AGA follow similar templates (see Appendix B). This similarity is reflected in the level of edge overlap. Curiously, the weight overlap between the AGA and the ANA sheaves is low. We conjecture that this distinction between weight and edge overlap is due to the different roles they play: weights store information, while edges guide the function of the task. While ANA and AGA share similar templates (and therefore exhibit higher edge overlap), performing the task requires distinct parametrized information (resulting in lower weight overlap).

Finding 3: Unveiling the factual recall pipeline in GPT. Lastly, we confirm the *factual recall pipeline* hypothesis: that recall occurs in two distinct stages (Meng et al., 2022; Geva et al., 2023; Niu et al., 2024; Hernandez et al., 2024). The left panel of Figure 3 illustrates the layer-wise average number of MLP and attention weight parameters retained in the 12 relation-specific DiscoGP sheaves learned from PARAREL. We observe that MLPs retain substantially more weights in the OQA sheaves compared to attention heads, especially in the lower transformer layers. This finding aligns with recent work that observed MLP sublayers function as key-value memory for factual knowledge *extraction* (Geva et al., 2022). Conversely, the right panel of Figure 3 shows the number of sheaf edges at each layer, detailing connections from lower-layer attention heads to current-layer MLPs (Attention to MLP) and from preceding MLPs to current-layer attention heads (MLP to Attention). Notably, the set of connections in upper layers is dominated by MLP-to-attention edges. This observation supports recent findings in mechanistic interpretability research suggesting that attention heads play a major role in *propagating* the retrieved factual knowledge from early-site MLPs to upper transformer layers, thereby selecting the most relevant information for answering questions (Geva et al., 2023).

7 Conclusion

In this work, we propose a novel sheaf discovery task that addresses the research gap left by previous automatic circuit discovery studies. We also introduce DiscoGP, a state-of-the-art sheaf discovery framework that identifies sheaves with the highest functional fidelity using the fewest connections and edges, by combining weight and edge pruning. This method enables neuron-level granularity and reveals several novel insights into the internal workings of LLMs (sheaf modularity and overlap), while also confirming previously observed trends (the factual recall pipeline). We hope our work inspires further research into sheaf discovery as we edge closer to prying open the LM "black box."

Limitations

While our experimental setup is sufficiently comprehensive for the purposes of this study, there is always room to expand the range of tasks and language models evaluated. We focus on GPT-2 to enable direct comparisons with other publicly available systems, but future work could consider larger or more recent models. Additionally, our experiments are limited to English, and extending the analysis to other languages would help assess the generality of our findings.

References

Federico Adolfi, Martina G. Vilas, and Todd Wareham. 2025. The computational complexity of circuit discovery for inner interpretability. In *The Thirteenth International Conference on Learning Representations*.

Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. 2023. Discovering Knowledge-Critical Subnetworks in Pretrained Language Models. *Preprint*, arXiv:2310.03084.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *Preprint*, arXiv:1308.3432.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding transformer circuits with edge pruning. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*.

Steven Cao, Victor Sanh, and Alexander Rush. 2021. Low-Complexity Probing via Finding Subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 960–966, Online. Association for Computational Linguistics.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge Neurons in Pretrained Transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Nicola De Cao, Leon Schmid, Dieuwke Hupkes, and Ivan Titov. 2022. Sparse Interventions in Language Models with Differentiable Masking. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 16–27, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and Improving Consistency in Pretrained Language Models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart Shieber, Tal Linzen, and Yonatan Belinkov. 2021. Causal Analysis of Syntactic Agreement Mechanisms in Neural Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1828–1843, Online. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2018. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal Abstractions of Neural Networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586. Curran Associates, Inc.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting Recall of Factual Associations in Auto-Regressive Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages

12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing Model Behavior with Path Patching. *Preprint*, arXiv:2304.05969.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, Online. Association for Computational Linguistics.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have Faith in Faithfulness: Going Beyond Circuit Overlap When Finding Model Mechanisms. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *The Twelfth International Conference on Learning Representations*.

Yihuai Hong, Lei Yu, Haiqin Yang, Shauli Ravfogel, and Mor Geva. 2024. Intrinsic Evaluation of Unlearning Using Parametric Knowledge Traces. *Preprint*, arXiv:2406.11614.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2023. Sparse Autoencoders Find Highly Interpretable Features in Language Models. In *The Twelfth International Conference on Learning Representations*.

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

David Lindner, Janos Kramar, Sebastian Farquhar, Matthew Rahtz, Thomas McGrath, and Vladimir Mikulik. 2023. Tracr: Compiled transformers as a laboratory for interpretability. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2024. Sparse Feature Circuits: Discovering and Editing Interpretable Causal Graphs in Language Models. In *The Thirteenth International Conference on Learning Representations*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36.

Philipp Mondorf, Sondre Wold, and Barbara Plank. 2025. Circuit Compositions: Exploring Modular Structures in Transformer-Based Language Models. *Preprint*, arXiv:2410.01434.

Neel Nanda. 2022. A comprehensive mechanistic interpretability explainer & glossary.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale.

Neel Nanda and Joseph Bloom. 2022. TransformerLens.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? In *The Twelfth International Conference on Learning Representations*.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom In: An Introduction to Circuits. *Distill*, 5(3):10.23915/distill.00024.001.

OpenAI. 2023. GPT-4 Technical Report. *Preprint*, arXiv:2303.08774.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, page 24.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Preprint*, arXiv:1910.10683.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Pedro Savarese, Hugo Silva, and Michael Maire. 2020. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems*, volume 33, pages 11380–11390. Curran Associates, Inc.

Claudia Shi, Nicolas Beltran-Velez, Achille Nazaret, Carolina Zheng, Adrià Garriga-Alonso, Andrew Jesson, Maggie Makar, and David Blei. 2024. Hypothesis testing the circuit hypothesis in LLMs. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution Patching Outperforms Automated Circuit Discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 Small. In *The Eleventh International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Zhengxuan Wu, Atticus Geiger, Thomas Icard, Christopher Potts, and Noah Goodman. 2023. Interpretability at Scale: Identifying Causal Mechanisms in Alpaca. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

Lei Yu, Meng Cao, Jackie CK Cheung, and Yue Dong. 2024a. Mechanistic Understanding and Mitigation of Language Model Non-Factual Hallucinations. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7943–7956, Miami, Florida, USA. Association for Computational Linguistics.

Lei Yu, Virginie Do, Karen Hambardzumyan, and Nicola Cancedda. 2024b. Robust LLM safeguarding via refusal feature adversarial training. In *The Thirteenth International Conference on Learning Representations*.

Dinghuai Zhang, Kartik Ahuja, Yilun Xu, Yisen Wang, and Aaron Courville. 2021. Can subnetwork structure be the key to out-of-distribution generalization? In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12356–12367. PMLR.

Fred Zhang and Neel Nanda. 2023. Towards Best Practices of Activation Patching in Language Models: Metrics and Methods. In *The Twelfth International Conference on Learning Representations*.

A DiscoGP Implementation Details

Post-hoc Sheaf Pruning Since the training objective (8) does not consider graph connectivity, we can further simplify the model by (1) removing a node v from the computation graph if all of its weights have been pruned, and (2) performing a reverse BFS from the output node to eliminate edges that do not contribute to the final result.

Split QKV Pruning Following Conmy et al. (2023), we separate the query (Q), key (K) and value (V) activations and introduce an "output" node within each attention head. Figure 4 shows an illustration of the configuration.

B Evaluation Tasks & Data

BLiMP BLiMP (Warstadt et al., 2020) consists of 67 individual datasets, each containing minimally different sentence pairs that contrast in grammatical acceptability and isolate specific phenomena in syntax, morphology, or semantics. However, BLiMP was designed for bidirectional LMs such as BERT, which require the model to attend to both

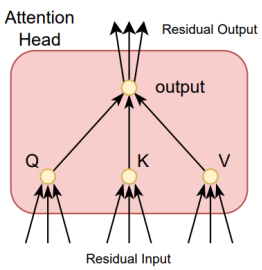


Figure 4: Split QKV Pruning.

Agreement Phenonemon	Good sentence	Bad sentence	Converted input query	True answer	False answer
Anaphor Gender Agreement	Katherine can't help herself.	Katherine can't help himself.	Katherine can't help	herself	himself
Anaphor Number Agreement	Susan revealed herself.	Susan revealed themselves.	Susan revealed	herself	themselves
Det Noun Agr. 1	Raymond is selling this sketch.	Raymond is selling this sketches.	Raymond is selling this	sketch	sketches
Det Noun Agr. Irr. 1	Laurie hasn't lifted those cacti.	Laurie hasn't lifted those cactus.	Laurie hasn't lifted those	cacti	cactus
Det Noun Agr. with Adj. 1	Rebecca was criticizing those good documentaries.	Rebecca was criticizing those good documentary.	Rebecca was criticizing those good	documentaries	documentary
Det Noun Agr. with Adj. Irr. 1	Some waiters broke this lost foot.	Some waiters broke this lost feet.	Some waiters broke this lost	foot	feet

Table 6: Examples of the BLiMP and their converted data.

Templates
Then, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] had a lot of fun at the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] were working at the [PLACE]. [B] decided to give a [OBJECT] to [A]
Then, [B] and [A] were thinking about going to the [PLACE]. [B] wanted to give a [OBJECT] to [A]
Then, [B] and [A] had a long argument, and afterwards [B] said to [A]
After [B] and [A] went to the [PLACE], [B] gave a [OBJECT] to [A]
When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give it to [A]
When [B] and [A] got a [OBJECT] at the [PLACE], [B] decided to give the [OBJECT] to [A]
While [B] and [A] were working at the [PLACE], [B] gave a [OBJECT] to [A]
While [B] and [A] were commuting to the [PLACE], [B] gave a [OBJECT] to [A]
After the lunch, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Afterwards, [B] and [A] went to the [PLACE]. [B] gave a [OBJECT] to [A]
Then, [B] and [A] had a long argument. Afterwards [B] said to [A]
The [PLACE] [B] and [A] went to had a [OBJECT]. [B] gave it to [A]
Friends [B] and [A] found a [OBJECT] at the [PLACE]. [B] gave it to [A]

Table 7: Sentence templates for generating the IOI dataset.

Placeholder Type	Candidate Infilling Words
[A] and [B] (names)	Michael, Christopher, Jessica, Matthew, Ashley, Jennifer, Joshua Daniel, David, James, Robert, John, Joseph, Andrew, Ryan, Bran Justin, Sarah, William, Jonathan, Stephanie, Brian, Nicole, Nicho Heather, Eric, Elizabeth, Adam, Megan, Melissa, Kevin, Steven, Timothy, Christina, Kyle, Rachel, Laura, Lauren, Amber, Brittan Richard, Kimberly, Jeffrey, Amy, Crystal, Michelle, Tiffany, Jere Mark, Emily, Aaron, Charles, Rebecca, Jacob, Stephen, Patrick, Kelly, Samantha, Nathan, Sara, Dustin, Paul, Angela, Tyler, Scot Andrea, Gregory, Erica, Mary, Travis, Lisa, Kenneth, Bryan, Lin Jose, Alexander, Jesse, Katie, Lindsay, Shannon, Vanessa, Court Alicia, Cody, Allison, Bradley, Samuel.
[PLACE]	store, garden, restaurant, school, hospital, office, house, station.
[OBJECT]	ring, kiss, bone, basketball, computer, necklace, drink, snack.

Table 8: Candidate infilling words of IOI sentence templates.

Relation ID	Relation	No. of queries	Sample Query	True answer
P103	native language	977	The mother tongue of Victor Horta is	Dutch
P138	named after	645	Rawlings Gold Glove Award, which is named for	glove
P159	headquarters location	967	The headquarter of Strait Shipping is located in	Wellington
P176	manufacturer	982	Honda RA272 is produced by	Honda
P264	record label	429	Johnny Carroll's record label is	Decca
P279	subclass of	964	Nucleoporin 62, a type of	protein
P30	continent	975	Romulus Glacier is located in	Antarctica
P407	language of work or name	877	Ten Years Gone is a work written in	English
P449	original network	881	Himalaya with Michael Palin was originally aired on	BBC
P495	country of origin	909	Mundo Obrero was from	Spain
P1376	capital of	234	Guangzhou is the capital of	Guangdong
P36	capital	703	The capital city of Porto District is	Porto

Table 9: PARAREL relations and sample queries used for circuit discovery.

preceding and following context. Therefore, we use the six BLiMP paradigms applicable to decoder-only LMs (specifically GPT-2). See Table 6 for example contrasting sentence pairs and their corresponding query prompts for circuit discovery.

Indirect object identification Wang et al. (2022) created dataset samples for IOI using templates with random single-token names, places and items. We follow their data curation pipeline by taking the same set of 15 templates and candidate infilling words to generate our circuit discovery dataset. At each trial, we randomly draw a template and a set of infilling tokens to construct a full sentence. We then convert the generated sentence into a binary classification question, where the input prompt is the sentence prefix without the last indirect object, and the two candidate next tokens are the indirect object and the subject tokens. See Table 7 and 8 for a complete list of IOI sentence templates and candidate infilling words.

PARAREL We use the PARAREL dataset by Elazar et al. (2021) that consists of 38 relation types and 27,738 (subject, relation, object) fact triples such as (*Canada*, capital city, *Ottawa*). We then use the templates created by (Dai et al., 2022) to convert each fact triple into multiple query prompts (e.g. "`The capital city of Canada is __`"). We take prompts generated from triples with 12 out of 38 PARAREL relations that satisfy the following two conditions: 1) there is a unique object entity answer for each (subject, relation) pair; and 2) the object word always comes at the end of the template-generated sentence so that it can be predicted by an autoregressive language model. We finally obtained a total of 9,543 queries as our dataset of open-domain question answering, and we learn a circuit for each relational dataset for every circuit discovery method. See Table 9 for a list of the 12 relations we used together with the example fact triples and queries.

C Baseline Methods

We obtain the original implementations released by the authors and adapt them to work with the same task and configurations as DiscoGP.[4]

For the threshold-based greedy search algorithms, since performance is not an objective in the circuit discovery process, we can obtain circuits with any level of sparsity by adjusting the thresholds. Therefore, we tune the threshold τ for each task and report the result that has a comparable — and larger — sparsity budget than DiscoGP. This puts ACDC and EAP at an advantage compared to DiscoGP in the sparsity–performance trade-off, yet our results show that DiscoGP still outperforms both.

D Detailed PARAREL Results

Table 10 lists our PARAREL results. Again, DiscoGP achieves the best performance across all tasks while mostly using the fewest weight parameters and edges. The PARAREL task differs from the BLiMP and IOI tasks in that test set and training set performance diverge significantly. This is expected, as factual information tends to be more dispersed. For example, Dai et al. (2022); Niu et al. (2024) found that each piece of factual information (e.g., Canada's capital is Ottawa) can be attributed to a handful of neurons, while Niu et al. (2024) found that the entire determiner–noun agreement can be attributed to the same amount of neurons.

[4]Edge pruning: https://github.com/princeton-nlp/Edge-Pruning, ACDC: https://github.com/ArthurConmy/Automatic-Circuit-Discovery/ and EAP https://github.com/Aaquib111/edge-attribution-patching.

Task	Discovery Method	Test Set Acc. (higher is better)	Train Set Acc. (higher is better)	KL Div. (lower is better)	Comp. Acc. (%) (random* is better)	Weight Density (%) (lower is better)	Edge Density (%) (lower is better)
P30	ACDC	0.30	0.27	0.3194	1.20	100	4.57
	EAP	1.18	1.63	0.3900	0.08	100	6.42
	Edge	92.1	89.5	0.0115	0.90	100	**2.34**
	Weight	86.8	92.6	0.0093	0.23	3.86	100
	DiscoGP	**95.6**	**92.6**	**0.0076**	0.35	**3.64**	3.01
P36	ACDC	0.72	0.86	0.3706	0.42	100	5.99
	EAP	1.18	1.86	0.3272	1.21	100	4.59
	Edge	62.7	90.5	0.0164	0.86	100	3.45
	Weight	67.3	90.3	0.0191	1.04	4.54	100
	DiscoGP	**69.2**	**91.1**	**0.0094**	0.85	**4.17**	**3.22**
P103	ACDC	0.54	1.16	0.2913	0.36	100	5.18
	EAP	0.93	0.57	0.3329	0.51	100	5.32
	Edge	91.4	88.1	0.0345	0.88	100	2.02
	Weight	83.0	87.4	0.0231	0.96	**4.35**	100
	DiscoGP	**93.5**	**89.7**	**0.0202**	0.15	4.7	**3.36**
P138	ACDC	0.96	0.59	0.3096	1.29	100	4.99
	EAP	1.98	0.78	0.2429	0.31	100	5.40
	Edge	64.9	96	0.022	1.52	100	2.33
	Weight	63.3	92.4	0.0375	0.73	1.57	100
	DiscoGP	**68.0**	**94.9**	**0.029**	0.46	**1.34**	**1.9**
P159	ACDC	0.56	1.64	0.3630	0.35	100	4.92
	EAP	1.78	1.44	0.3011	0.30	100	6.41
	Edge	57.3	84.2	0.0552	0.91	100	**2.05**
	Weight	58.8	88.7	0.0276	0.59	3.38	100
	DiscoGP	**62.5**	**89.8**	0.0168	0.57	**3.79**	2.81
P176	ACDC	0.53	1.77	0.3823	0.48	100	6.99
	EAP	0.91	1.39	0.3050	1.26	100	4.89
	Edge	86.5	98.6	0.0117	0.47	100	3.04
	Weight	86.0	99.2	**0.0095**	0.88	1.34	100
	DiscoGP	**95.6**	**99.4**	0.0104	0.85	**1.01**	**2.73**
P264	ACDC	1.51	0.51	0.2250	0.57	100	4.48
	EAP	0.27	0.39	0.2165	1.26	100	6.24
	Edge	77.3	89.4	0.0297	0.16	100	2.45
	Weight	82.3	90.8	0.0266	1.24	3.58	100
	DiscoGP	**82.9**	**90.3**	**0.0245**	0.77	**3.36**	**2.43**
P279	ACDC	1.30	0.54	0.3590	0.77	100	4.69
	EAP	0.74	0.55	0.3153	0.52	100	6.34
	Edge	69.5	87.0	0.0562	0.68	100	4.98
	Weight	75.5	93.9	0.0337	0.13	2.53	100
	DiscoGP	**76.9**	**95.2**	**0.0200**	0.47	**2.14**	**3.57**
P407	ACDC	1.41	1.51	0.3492	0.32	100	4.96
	EAP	0.49	0.66	0.2036	0.03	100	5.78
	Edge	80.1	93.9	0.0085	0.55	100	**2.1**
	Weight	77.0	94.1	0.0097	0.29	**1.94**	100
	DiscoGP	**83.3**	**95.0**	**0.0073**	0.97	2.24	2.89
P449	ACDC	0.59	1.20	0.5230	0.20	100	6.88
	EAP	0.87	0.33	0.4976	0.82	100	6.87
	Edge	70.4	93.3	**0.0090**	0.95	100	**3.36**
	Weight	71.4	93.7	0.0098	1.39	2.7	100
	DiscoGP	**74.7**	**93.7**	0.0099	1.09	**2.58**	3.43
P495	ACDC	0.22	0.22	0.5130	0.21	100	4.37
	EAP	1.30	0.47	0.4058	0.43	100	6.12
	Edge	65.8	86.1	0.115	0.76	100	3.92
	Weight	65.4	87.1	0.102	0.70	2.54	100
	DiscoGP	**70.7**	**90.3**	**0.082**	0.63	**2.08**	**2.17**
P1376	ACDC	1.22	1.76	0.5535	0.65	100	6.14
	EAP	0.38	0.76	0.5551	0.40	100	6.66
	Edge	49.4	89.3	0.101	0.77	100	3.57
	Weight	55.2	92.5	0.082	0.24	**1.68**	100
	DiscoGP	**57.7**	**94.6**	**0.047**	0.28	2.13	**3.36**

Table 10: Sheaf Discovery Performance Comparison across PARAREL relations. Again, DiscoGP achieves the best performance across all tasks while mostly using the fewest weight parameters and edges. The best-performing methods are highlighted in **bold**. *: For complement sheaf accuracy, successful searches are expected to yield random performance. Therefore, scores in the vicinity of random indicate good performance, and direct comparison of complement scores is not meaningful.