# CIRCUITFACT: CIRCUIT DISCOVERY FOR MECHANISTIC INTERPRETABILITY OF GRAPH NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Mechanistic interpretability seeks to understand neural networks by identifying the internal computational mechanisms underlying their behavior, often represented as circuits. However, recent circuit discovery methods have focused predominantly on transformer language models, and their application to graph neural networks (GNNs) remains largely unexplored. We propose CircuitFACT, a circuit discovery method for GNNs built on computation trees, which explicitly represent the message-passing dependencies involved in a GNN prediction. CircuitFACT first constructs these computation trees, attributes their components using Shapley-based attribution, and trims the resulting structures to obtain compact circuits. These circuits provide a structured view of the computations supporting a prediction and can be related back to the corresponding elements of the input graph.

## 1 INTRODUCTION

In recent years, Graph Neural Networks (GNNs) have gained significant importance due to their ability to learn from relational data, with applications ranging from molecular property prediction and drug discovery to social network analysis and recommender systems (Fan et al., 2019; Jiang et al., 2021; Bongini et al., 2021; Huang et al., 2019). Their expressive power comes from message passing, the mechanism through which GNNs process information: at each layer, every node aggregates information from its neighbors, so that after $L$ layers its representation depends on its entire $L$-hop neighborhood. This recursive aggregation, however, makes GNNs notoriously hard to interpret. As GNNs are increasingly deployed in high-stakes scientific domains, understanding how they reach their predictions has become a pressing need (Nandan et al., 2025).

Several approaches address the problem of explaining GNNs predictions. Instance-level methods (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021) identify the nodes, edges, or subgraphs most relevant to an individual prediction. Global methods, instead provide general explanations of the models. These approaches answer the question of *what* in the input drives a prediction, but they treat the network itself as a black box: they do not reveal *how* information is processed inside the model, nor *why* a given input pattern leads to a given output (Nandan et al., 2025).

Mechanistic interpretability (Elhage et al., 2021; Conmy et al., 2023) is a research paradigm that aims to answer those questions by reverse-engineering neural networks to uncover the fundamental internal components that contribute to or causally influence specific behaviors. A key concept in this line of work is the *circuit*, a subgraph of the model's computation graph that is responsible for a given behavior (Olah et al., 2020; Anani et al., 2026). These methods, however, are generally designed and applied almost exclusively to transformer-based architectures, such as large language models and vision transformers, while their application to GNNs remains unexplored. Moreover, the central mechanistic question for GNNs is inherently different from that of language and vision models, since the distinctive capability of a GNN lies in recognising structural patterns in its input. Recent work (Ragno et al., 2025; Palumbo et al., 2026) propose to design GNNs such that mechanistic insights can be obtained directly from their weights through distillation or intense constraints of the learning procedure. However, such approaches are subject to performance decays and generally require heavy training procedures. In this paper, instead, we focus on understanding *how a generic*



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

*GNN recognises subgraphs*, namely which internal computations drive this recognition and which recurring input patterns they respond to, without constraining its training paradigm.

To this end, we introduce CircuitFACT, a novel circuit discovery method for GNNs built on the notion of computation trees (Xu et al., 2019; Armgaan et al., 2024). Since the representation of a node at a given layer is fully determined by its computation tree, computation trees provide a natural bridge between the internal activations of the model and the structure of the input graph. CircuitFACT operates in three phases: it first generates the computation trees of the input graphs, then extracts circuits by recursively estimating the contribution of each neuron through Shapley values (Shapley, 1953) and finally trims each circuit to discard branches that do not connect to the input layer. Since computation trees are a direct consequence of message passing, CircuitFACT is architecture-agnostic and can be applied to any message-passing GNN. Finally, we evaluate CircuitFACT on graph classification benchmarks and show that it is able to identify meaningful and significant pattern recognition circuits inside the model, responsible for its reasoning process. In summary, our contributions are as follows:

- We introduce CircuitFACT, the first circuit discovery method for GNNs that generalizes across message-passing architectures, identifying the neuron-level mechanisms responsible for model predictions.
- We show how computation trees can be used as a principled bridge between internal GNN computations and input-graph structure, enabling the discovery of circuits that remain interpretable in the graph domain.
- We demonstrate across multiple graph classification benchmarks[1] and GNN architectures that the discovered circuits are both relevant to the model's predictions and associated with meaningful graph patterns.

## 2 Related Work

Understanding how a trained GNN reaches its predictions has been approached from two largely disconnected directions, one asking how the model computes and the other asking what it represents, and our work is best understood by situating it against both. Mechanistic interpretability is a bottom-up research paradigm that seeks to reverse-engineer neural networks to uncover the fundamental structural components that contribute to or causally influence specific behaviors (Olah et al., 2020; Bereska & Gavves, 2024; Somvanshi et al., 2026), and its central abstraction is the circuit, a subgraph of the model's computation graph responsible for a given behavior. Early approaches use manual causal interventions such as activation and path patching (Meng et al., 2022; Wang et al., 2023), which motivated automated circuit discovery methods including iterative edge pruning via activation patching (ACDC) (Conmy et al., 2023), gradient-based approximations such as edge attribution patching and its integrated-gradients variant (EAP (Syed et al., 2024), EAP-IG (Hanna et al., 2024)), and optimization-based approaches that learn edge masks (Edge Pruning (Bhaskar et al., 2024)). However most of this work targets transformer language models and other architectures remain largely unexplored. Across these methods, circuit discovery is generally framed as the search for circuits that activate a concept, that is, a stable, recurring, human-interpretable behavior, and the circuit is only useful insofar as it can be tied back to something a human can recognize.

This framing is what connects mechanistic interpretability to a second, much larger body of work that starts directly from the concept side rather than from the computation. This second direction treats the network as a black box, asking not how it computes but which recurring input patterns drive its predictions. Instance-level methods identify the nodes, edges, or subgraphs most relevant to an individual prediction (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021). More global methods instead identify patterns that hold across predictions: GCExplainer (Magister et al., 2021) clusters node embeddings into representative subgraph concepts, GLGExplainer (Azzolin et al., 2023) combines learned graphical concepts into logical class-level explanations, and GraphTrail (Armgaan et al., 2024) represents concepts as computation trees and distills the GNN's input-output behavior into logical formulas over them. A parallel line of work pushes this concept-first idea one step further by constraining the model itself at training time, so that concepts are no longer approximated

[1] All code, data, and implementation details required to reproduce our experiments are included in the supplementary material and will be made available in a public repository upon acceptance.

after the fact but are the mechanism by which the model computes, at the cost of retraining and, often, some accuracy. Prototype-based methods (Zhang et al., 2022; Ragno et al., 2024; Dai & Wang, 2025; Vincent et al., 2026) tie predictions to learned prototype (sub)graphs, while a separate family constrains the message-passing architecture through logical formulas by construction (Azzolin et al., 2025; Ragno et al., 2025; Palumbo et al., 2026).

Our work sits at the intersection of these two directions without collapsing into either: like the concept-first and self-explainable methods, we ultimately want circuits that can be read back as recurring input-graph patterns, but like mechanistic interpretability we get there by first discovering internal computation bottom-up rather than assuming or imposing the concept in advance.

Within mechanistic interpretability itself, circuit discovery has only just begun to reach GNNs. MINAR (He et al., 2026) adapts EAP and EAP-IG to message-passing networks and extracts neuron-level circuits from neural algorithmic reasoners, recovering, for instance, the theoretically optimal circuit for Bellman-Ford. However the method is validated only on shallow two-layer GNNs trained on synthetic algorithmic tasks whose ground-truth mechanism is known in advance.

**Our approach.** Our work bridges the two perspectives: we discover internal mechanisms bottom-up and interpret them top-down in terms of input-graph concepts. Building on the notion of computation trees (Xu et al., 2019; Armgaan et al., 2024), we propose a circuit discovery method that is not tied to a specific GNN architecture. Unlike GraphTrail, which seeks a global symbolic approximation of the model's input-output behavior, we identify the internal mechanisms themselves; unlike MINAR, which validates circuits against known algorithmic mechanisms on synthetic tasks, we ground internal computations in input-graph patterns, making the approach applicable to domains where no ground-truth mechanism is available.

## 3 Background

In this section, we introduce the mathematical foundations underlying our approach: *computation trees* and *Shapley values*. Computation trees sit at the basis of our approach, as we treat them as the atomic units that can be represented and manipulated by the model, while Shapley values are the tool we use to quantify the impact and attribute contributions between neurons.

### 3.1 Computation Trees

Message-passing GNNs compute a node's representation by recursively aggregating information from its neighborhood. This recursive structure can be made explicit through the notion of a *computation tree*, which unrolls the message-passing process rooted at a given node into an explicit tree of dependencies. We adopt the definition introduced by (Armgaan et al., 2024).

**Definition 1 (Graph)** *A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ over a node set $\mathcal{V}$, edge set $\mathcal{E} = \{(u, v) \mid u, v \in \mathcal{V}\}$ and a node feature matrix $\boldsymbol{X} = \{\mathbf{x}_v \mid v \in \mathcal{V}\}$ where $\mathbf{x}_v \in \mathbb{R}^d$ is the set of features characterizing each node.*

**Definition 2 (Rooted Computation Tree $\mathcal{T}^L(v)$)** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, a node $v \in \mathcal{V}$, and the number of layers $L$ in the GNN, the computation tree $\mathcal{T}^L(v)$ rooted at $v$ is constructed as follows:*

1. *Enumerate all paths (including those with repeated vertices) starting from $v$ and extending up to $L$ hops away.*
2. *Merge these paths into a tree structure where (i) the root is always $v$, and (ii) two nodes from different paths are merged if (a) they are at the same depth in their respective paths, and (b) all their ancestors in the paths have already been merged.*

The resulting tree $\mathcal{T}^L(v)$ represents how information flows to node $v$ over $L$ rounds of message passing, and is also referred to as the *receptive field* of $v$. As shown by (Armgaan et al., 2024), the computation tree is sufficient to determine the node's embedding: for an $L$-layer GNN, $h_v^L$ is fully

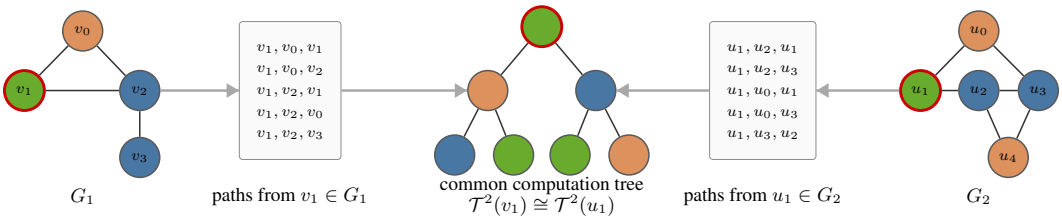



Figure 2: GraphTrail-style construction of a rooted computation tree. The root nodes $v_1$ and $u_1$ are indicated by red borders. For each root, paths up to length $L = 2$ are enumerated and merged into a computation tree. Although $G_1$ and $G_2$ have different graph structures, their rooted computation trees are isomorphic, i.e., $\mathcal{T}^2(v_1) \cong \mathcal{T}^2(u_1)$. Computation trees therefore capture the information available to an $L$-layer message-passing GNN, following GraphTrail (Armgaan et al., 2024).

determined by $\mathcal{T}^L(v)$, and any two nodes with isomorphic computation trees, $\mathcal{T}^L(v) \cong \mathcal{T}^L(u)$, receive identical embeddings, $h_v^L = h_u^L$, a consequence of the fact that the expressive power of message-passing GNNs is upper-bounded by the 1-WL test (Xu et al., 2019). This property makes computation trees a natural, architecture-agnostic unit of analysis: the structure induced by the GNN allows us to restrict attention from the exponentially large space of possible subgraphs to the linearly sized set of computation trees actually instantiated by the GNN, $\mathcal{T}(\mathcal{G}) = \{\mathcal{T}^L(v) \mid v \in \mathcal{V}\}$, without any loss of information regarding the model's computation. In our work, we build on this property and treat computation trees as the atomic concepts over which the model's internal computation can be represented, tracked, and attributed.

Figure 2 illustrates the construction of computation trees from two different input graphs. Despite the graphs having different structures, the rooted computation trees can be isomorphic, leading to identical node embeddings.

### 3.2 Shapley Values

To quantify the contribution of individual components (in our case, neurons) to a model's output, we rely on the Shapley value, a solution concept from cooperative game theory originally introduced by (Shapley, 1953). Given a set of $p$ players, denoted by $[p] = \{1, \ldots, p\}$, and a value function $\nu : 2^{[p]} \to \mathbb{R}$ that assigns a payoff to every possible coalition (subset) of players, the Shapley value $\phi_i$ of player $i$ is defined as the weighted average marginal contribution across all possible coalitions not containing $i$:

$$\phi_i = \sum_{S \subseteq [p] \setminus \{i\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(\nu(S \cup \{i\}) - \nu(S)\right). \tag{1}$$

Equivalently, $\phi_i$ is the expected marginal contribution of player $i$ when players are added according to a uniformly random permutation.

For a cooperative game, the Shapley value is the unique allocation rule satisfying the axioms of efficiency, symmetry, additivity, and null-player (Shapley, 1953), providing a principled way to distribute among the players the total payoff gained by the full coalition relative to the empty coalition. In neural networks, Shapley values have been widely used as a principled attribution method to quantify the contribution of individual features or components to a model's output, most notably through the SHAP framework (Lundberg & Lee, 2017). In particular, Kernel SHAP exploits an equivalence between Shapley values and a specific weighted linear regression problem to estimate additive feature attributions. Rather than the regression-based surrogate of SHAP, we adopt Monte Carlo permutation sampling (Castro et al., 2009), a standard unbiased estimator of Eq. equation 1: we draw $M$ random orderings of the players and, starting from the fully ablated state, reinstate one player at a time, recording the marginal change in the value function caused by each player joining the coalition. Averaging these marginal contributions over the $M$ sampled permutations yields an unbiased estimate of $\phi_i$ for every player.

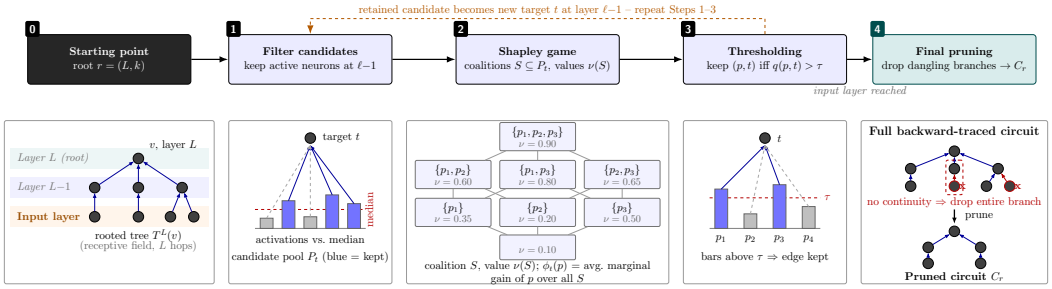



Figure 3: CIRCUITFACT circuit discovery procedure. We build a circuit by recursively tracing backward from a root neuron: filtering active candidates by median activation, attributing their contribution via Shapley values, and keeping only edges above a threshold $\tau$, repeating layer by layer down to the input. Branches that never reach the input are pruned entirely, yielding the compact circuit $C_r$.

## 4 METHOD

CIRCUITFACT discovers neuron-level circuits that explain how a message-passing GNN recognizes recurring patterns in its input graphs. Our central premise is that the structural part of a GNN's computation takes place in its message-passing layers: through successive neighborhood aggregation, these layers transform the rooted computation tree of each node into a learned representation. CIRCUITFACT uses computation trees as the bridge between input-graph structure and internal neural computation. For each neuron in the final message-passing layer, we trace backward through the preceding layers to neurons that consistently contribute to its activation. We estimate these dependencies using Shapley values, retaining only connections that are locally influential, and finally prune branches that do not connect the root to the input layer. The result is a compact directed circuit that can be analyzed both as an internal model mechanism and as a recurring pattern in the input graph.

**Computation Graphs and Circuits.** Let $f$ be a trained message-passing GNN with $L$ convolutional layers, and let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ be an input graph. For a node $v \in \mathcal{V}$, the $L$-layer message-passing computation associated with $v$ can be unfolded into the rooted computation tree $\mathcal{T}^L(v)$ introduced in Section 3.2. This tree records the node occurrences and paths through which information can reach $v$ after $L$ rounds of message passing. We use it to make the internal dependencies of the forward pass explicit.

For a computation tree $T$ rooted at a node $v$, let $h_k^{(\ell)}(T)$ denote the activation of neuron $k$ at layer $\ell$ at the root of $T$, where $\ell = 0$ denotes the input layer. We define the computation graph $\mathcal{G}_{\text{comp}}(T) = \big(\mathcal{V}_{\text{comp}}(T), \mathcal{E}_{\text{comp}}(T)\big)$ as the directed layered graph whose vertex set $\mathcal{V}_{\text{comp}}(T)$ contains the neuron instances $(\ell, k)$ activated while evaluating $T$, and whose edge set $\mathcal{E}_{\text{comp}}(T)$ contains an edge $(\ell - 1, j) \rightarrow (\ell, k)$ whenever the layer-$(\ell-1)$ activations of neuron $j$, aggregated over the root and its children in $T$, are used to compute the activation of neuron $k$ at layer $\ell$ at the root. Thus, the computation graph represents the dependency structure induced by message passing at the level of individual neurons.

**Definition 3 (Circuit)** *Let $r = (L, k)$ be a root neuron in the final convolutional layer, and let $\mathcal{I}_r$ be a collection of computation trees on which $r$ is evaluated. A* circuit *for $r$ is a directed layered graph*

$$\mathcal{C}r = (\mathcal{V}_r, \mathcal{E}_r) \text{ with } \mathcal{V}_r \subseteq \bigcup_{T \in \mathcal{I}_r} \mathcal{V}_{\text{comp}}(T) \text{ and } \mathcal{E}_r \subseteq \bigcup_{T \in \mathcal{I}_r} \mathcal{E}_{\text{comp}}(T) \tag{2}$$

*where every edge connects neurons in consecutive layers, every retained neuron lies on at least one directed path from an input-layer neuron to $r$, and every edge is selected by the attribution rule defined below 4. A circuit therefore represents a recurrent internal mechanism through which input features are combined to support the root activation.*

The circuit finally identifies the directed, layerwise dependencies by which information reaches the root. We root circuits in the final convolutional layer because this is the stage at which a message-passing GNN has completed its structural computation. After $L$ layers, $h_v^{(L)}$ incorporates the information available in the $L$-hop computation tree of $v$, so its coordinates encode learned structural and feature detectors. In node classification, this representation is passed directly to the prediction head; the head maps its coordinates to class logits but does not expand the receptive field or recognize additional graph structure. In graph classification, the final convolutional layer first produces the collection $\{h_v^{(L)}\}_{v\in\mathcal{V}}$, which is aggregated into a graph embedding as

$$h_G = \mathrm{POOL}\left(\left\{h_v^{(L)}\right\}_{v\in\mathcal{V}}\right), \qquad \mathrm{POOL} \in \{\mathrm{sum}, \mathrm{mean}, \mathrm{max}\}. \tag{3}$$

The pooling operation does not recognize a new local structure; it aggregates evidence from the structural detectors already computed at node level. Sum pooling accumulates this evidence, mean pooling measures its average prevalence, and max pooling records its strongest occurrence. Rooting circuits before pooling thus provides a common treatment of node- and graph-level tasks: it identifies the message-passing mechanism that recognizes a pattern, while the subsequent readout only combines, counts, or summarizes the resulting node-level activations.

**Circuit Discovery.** Figure 3 shows an overview of CIRCUITFACT discovery process. Given a root neuron $r = (L, k)$, CIRCUITFACT constructs $\mathcal{C}_r$ recursively from layer $L$ toward the input layer. At each step, the method considers a target neuron $t$ at layer $\ell$ and determines which neurons in layer $\ell-1$ recurrently contribute to its activation while remaining relevant to the original root $r$. Restricting the analysis to consecutive layers respects the direction of the message-passing computation and gives each retained edge a direct mechanistic interpretation.

Attribution is performed only over neurons that are active on the computation trees associated with the current target. Let $a_{i,k}^{(\ell)}$ be the activation of neuron $k$ at layer $\ell$ on computation tree $T_i$. We consider the neuron active when its activation exceeds the median of its activation distribution over the computation trees considered for the circuit:

$$a_{i,k}^{(\ell)} > \mathrm{Quantile}_{0.5}\left(\left\{a_{j,k}^{(\ell)}\right\}_{T_j\in\mathcal{I}_r}\right). \tag{4}$$

At the input layer, activity is determined directly from the input representation; for molecular graphs, for instance, an input feature is active when the corresponding atom-type indicator is present. This filter excludes neurons that are not engaged by the inputs under analysis and limits Shapley estimation to the candidate neurons that can plausibly participate in the realized computation.

Let $P_t$ be the active candidate neurons at layer $\ell - 1$ for a target neuron $t$ at layer $\ell$. We interpret $P_t$ as the set of players in a cooperative game. For a coalition $S \subseteq P_t$, let $\nu_t(S)$ be the activation of $t$ when the candidates in $S$ retain their observed activations and the remaining candidates are set to a baseline, for which we use the observed minimum activation. The contribution $\phi_t(p)$ of a candidate $p \in P_t$ is then its Shapley value in this game, as defined in Equation 1. Since exact computation is exponential in $|P_t|$, we exploit the permutation-based formulation of the Shapley value and estimate $\phi_t(p)$ by averaging the marginal contribution of $p$ over $M$ random permutations of $P_t$, revealing candidates one at a time.

For each candidate neuron $p$, CIRCUITFACT computes $\phi_{i,t}(p)$ the local contribution to the immediate target $t$ on computation tree $T_i$. For a target $t$ and candidate $p$, we aggregate the attributions over the set $\mathcal{I}_t$ of computation trees associated with $t$. The fraction of trees in which $p$ is locally supportive is

$$q(p,t) = \frac{1}{|\mathcal{I}_t|}\sum_{T_i\in\mathcal{I}_t} \mathbf{1}[\phi_{i,t}(p) > 0]\,. \tag{5}$$

We add the directed edge $(p, t)$ to the circuit if and only if

$$q(p,t) > \tau, \tag{6}$$

where $\tau \in [0, 1]$ controls the minimum fraction of computation trees in which $p$ must support $t$. Larger values of $\tau$ yield more conservative circuits that retain only mechanisms shared across most trees, whereas smaller values also admit mechanisms that operate on only a subset of them.

The rule in Equation 6 is applied recursively. CIRCUITFACT starts from $r$ at layer $L$; whenever a candidate $p$ is retained, it adds $(p, t)$ to the circuit and treats $p$ as the next target at layer $\ell - 1$.

After backward tracing, some branches may terminate before reaching the input layer. We therefore keep only the neurons and edges that lie on at least one directed path from an input-layer neuron to the root. This final pruning step removes incomplete branches and yields the circuit of Definition 3.

# 5 EXPERIMENTS

The evaluation of the discovered circuits is guided by two fundamental questions. First, we ask whether a circuit is actively used by the network to reach its decision, that is, whether intervening on it affects the prediction more than intervening on a random set of neurons, and whether it behaves differently from such a set. Second, we ask whether circuits reflect a behavior that can be observed in the input graphs. We address these two questions in Sections 5.1 and 5.2, respectively. We evaluate our circuits on four graph classification benchmarks, namely MUTAG (Ivanov et al., 2019), NCI1 (Wale & Karypis, 2006), AIDS (Riesen & Bunke, 2008), and Mutagenicity (Kazius et al., 2004), using three GNN architectures: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019). Since existing circuit discovery methods for GNNs are not directly applicable to our setting, we compare CircuitFACT with a baseline based on activation ablation, which we refer to as *Zero-ablation*. For each candidate neuron, this baseline sets its activation to the lowest percentile of its activation distribution, and considers the effect on a downstream neuron in the next layer significant if the activation of the latter crosses its median value. The same threshold $\tau$ used by CircuitFACT then determines whether the corresponding connection is retained, based on the fraction of computation trees in which the effect is significant.

## 5.1 ARE CIRCUITS ACTIVELY USED BY THE NETWORK?

We consider a circuit *necessary* (Azzolin et al., 2025) if intervening on it affects the model's decision significantly more than intervening on a random set of neurons of the same size. Since a circuit may either support or oppose the predicted class $\hat{y}_g$ of a graph $g$, the appropriate intervention depends on the sign of its contribution to the margin between the two classes in the final classification layer. We therefore apply one of two interventions:

- **Ablation** (positive contribution): the circuit supports the predicted class, so we set its neurons to zero and expect the logit of $\hat{y}_g$ to decrease.
- **Activation** (non-positive contribution): the circuit does not support the predicted class, so we set its neurons to the 90th percentile of their activation distribution and test whether this intervention alone is sufficient to induce the class promoted by the circuit.

For each circuit $C$, we construct $K = 50$ layer-matched random control sets. Let $n_\ell(C)$ denote the number of neurons of $C$ in layer $\ell$. Each control set contains exactly $n_\ell(C)$ neurons sampled uniformly without replacement from layer $\ell$, thereby matching the circuit's size and layer-wise composition while randomizing neuron identity. We apply to each control set the same intervention as to the original circuit and evaluate it on the same graphs.

To make the intervention effect comparable across the two cases, we define a signed effect that is positive when the intervention acts in the expected direction: $E(g) = z_{\text{base}}(g) - z_{\text{int}}(g)$ for ablation, and $E(g) = z_{\text{int}}(g) - z_{\text{base}}(g)$ for activation, where $z$ denotes the logit of the targeted class. We then compare the mean effect $\bar{E}(C)$ of the circuit with those of its random controls, $\bar{E}(R_k)$, and estimate the empirical $p$-value as

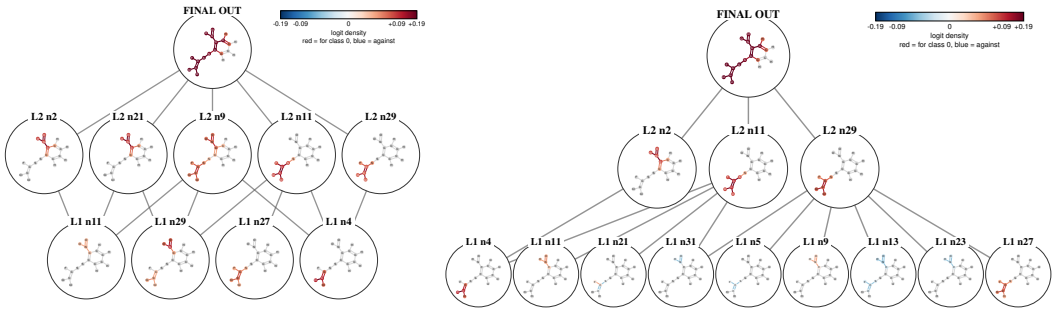
$$p = \frac{1 + \sum_{k=1}^{K} \mathbf{1}\left[\bar{E}(R_k) \geq \bar{E}(C)\right]}{K + 1}.$$

A circuit is deemed significant when the intervention determined by the sign of its contribution yields $p \leq 0.05$.

Table 1 reports results for the significant circuits identified by each method, averaged over the two classes. When significant circuits are found, both Shapley and Zero-ablation generally obtain substantial intervention effects, indicating that the extracted circuits have a causal influence on the

| Model | Method | MUTAG | | | | NCI1 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Abl.% ↑ | Act.% ↑ | Size ↓ | N Circ. | Abl.% ↑ | Act.% ↑ | Size ↓ | N Circ. |
| GAT | Shapley | **98.5** | 97.3 | 62.0 | 8.5 | **74.1** | 63.7 | 43.0 | 2.5 |
| | Zero-ablation | 64.5 | **100.0** | **55.5** | 4.5 | 59.6 | **73.7** | **41.5** | 2.5 |
| GCN | Shapley | **90.0** | **93.2** | **66.0** | 9.0 | **87.4** | **73.9** | **62.0** | 5.0 |
| | Zero-ablation | | / | | | | / | | |
| GIN | Shapley | 87.4 | **96.0** | 43.5 | 10.0 | 71.0 | 42.5 | 53.0 | 2.5 |
| | Zero-ablation | **96.9** | 54.1 | **36.5** | 4.5 | **71.6** | **67.7** | **41.5** | 2.5 |
| Model | Method | AIDS | | | | Mutagenicity | | | |
| | | Abl.% ↑ | Act.% ↑ | Size ↓ | N Circ. | Abl.% ↑ | Act.% ↑ | Size ↓ | N Circ. |
| GAT | Shapley | **56.1** | **37.0** | 32.5 | 3.0 | **95.3** | 61.5 | **29.5** | 4.0 |
| | Zero-ablation | 50.4 | 31.9 | **29.0** | 2.0 | 76.5 | **75.0** | 37.5 | 3.0 |
| GCN | Shapley | **53.4** | 74.6 | 70.0 | 10.5 | **89.1** | **60.0** | **39.0** | 4.0 |
| | Zero-ablation | 50.0 | **89.4** | **43.5** | 3.0 | | / | | |
| GIN | Shapley | **65.0** | **65.8** | **54.5** | 7.0 | 44.2 | **54.0** | **17.5** | 2.0 |
| | Zero-ablation | | / | | | **76.8** | 47.1 | 42.0 | 3.0 |

Table 1: Class-averaged results for significant circuits. Bold marks the better of Shapley and zero-ablation for each model, metric, and dataset (higher for Abl.% and Act.%, smaller for Size). Size is the number of neurons kept by the union of the circuits of a class (out of the 96 neurons of layers 1–3), averaged over the two classes. "/" indicates that no circuits were found for at least one of the two classes.




(a) (b)

Figure 4: Shapley (left) vs. zero-ablation GCN (right) circuits on a mutagenic molecule. In both circuits, neuron attributions concentrate on the two mutagenic substructures. In the Shapley circuit, contributions along connected neurons add up consistently toward the prediction, whereas the zero-ablation circuit contains neurons with opposing contributions, suggesting that some of its connections are spurious. The Shapley circuit also typically involves far fewer active neurons.

model's output. Ablation scores tend to be higher than activation scores on average, although this pattern is not systematic across all configurations.

The clearest difference between the two methods is the consistency with which they identify significant circuits. Shapley yields significant circuits for both classes in all 12 model–dataset configurations, whereas Zero-ablation fails to do so in 4 of them. Moreover, in most configurations where both methods succeed, Shapley identifies at least as many significant circuits as Zero-ablation. These results suggest that accounting for each neuron's contribution in the context of the other candidate neurons leads to more robust circuit identification than selecting neurons based on their individual zero-ablation effect.

### 5.2 Qualitative Evaluation

Although CircuitFACT extracts circuits from the network's internal computation graph, computation trees provide a mapping back to the original graph domain. In particular, a circuit can be projected onto the computation trees in which it occurs and related to the corresponding rooted neighborhoods of the input graph. Given a molecule and a target neuron in the last GNN layer, we estimate the Shapley value of each circuit neuron on every computation tree on which the target neuron $j$ is active, using the same players and baseline as in circuit discovery. As payoff, we take the target neuron's contribution to the logit of the predicted class $c$, namely $w_{c,j}h_{r,j}^{(L)}$, where $r$ denotes the root of the tree. To project these internal contributions onto the molecule, each neuron-level attribution is propagated to the atoms and bonds contained in the corresponding $\ell$-hop receptive field, and contributions from overlapping computation trees are summed. The resulting scores should therefore be interpreted as a projection of internal neuron contributions onto the input graph, rather than as direct atom-level Shapley values. The target neuron itself requires separate treatment, since it has no downstream neuron within the circuit. With additive pooling followed by a linear classifier, however, its contribution to a class logit decomposes exactly over the graph nodes. For a last-layer neuron $j$, its contribution to class $c$ is

$$w_{c,j}\sum_{v\in V}h_{v,j}^{(L)}=\sum_{v\in V}w_{c,j}h_{v,j}^{(L)},$$

which directly provides an atom-level decomposition of the contribution carried by that neuron. We visualize the resulting projected attributions as one heatmap per circuit neuron, with red and blue indicating positive and negative contributions, respectively. Figure 4 compares the circuits obtained with Shapley and Zero-ablation for the same target neuron on a mutagenic molecule, using the GCN trained on Mutagenicity. In both cases, neuron attributions concentrate on two known mutagenic substructures, the aromatic nitro group and the halogenated alkene Debnath et al. (1991), while the aromatic ring contributes little beyond the carbon bearing the nitro group. The two methods differ, however, in how these contributions are organized. Although the union of the circuits selected by Zero-ablation is generally smaller across the quantitative experiments (Table 1), fewer neurons of the Shapley circuit are active on this particular molecule. Moreover, the contributions propagated through the Shapley circuit are consistently aligned with the predicted class. In contrast, the Zero-ablation circuit contains connected neurons whose contributions oppose the predicted class. This illustrates a less coherent organization of the class-specific contribution and is consistent with Zero-ablation retaining connections that are not informative for this particular prediction.

## 6 Conclusions

We introduced CircuitFACT, a circuit discovery method for message-passing GNNs based on computation trees. By leveraging computation trees as a bridge between graph structure and internal neural computation, CircuitFACT extracts compact neuron-level circuits that can be interpreted both mechanistically and in terms of input-graph patterns.

Our experiments show that the discovered circuits correspond to computational pathways actively used by the network and provide more consistent mechanisms than a zero-ablation baseline. Furthermore, projecting circuits back onto the input graph reveals meaningful and domain-relevant structural patterns underlying model predictions.

More broadly, this work takes a first step toward mechanistic interpretability for graph neural networks. We hope it will stimulate future research on understanding GNNs through their internal computational mechanisms rather than solely through input-level explanations.

## Reproducibility statement

We have made every effort to ensure the reproducibility of our results. All models, hyperparameters, and training procedures are described in the main text and Appendix. An anonymized repository containing source code and datasets is provided in the supplementary materials and will be made available upon acceptance, allowing independent verification of all tests. All preprocessing scripts are supplied to facilitate replication of the reported results.

## REFERENCES


Alaa Anani, Tobias Lorenz, Bernt Schiele, Mario Fritz, and Jonas Fischer. Certified circuits: Stability guarantees for mechanistic circuits. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=IU0c9sP5b5.

Burouj Armgaan, Manthan Dalmia, Sourav Medya, and Sayan Ranu. GraphTrail: Translating GNN predictions into human-interpretable logical rules. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2023.

Steve Azzolin, Sagar Malhotra, Andrea Passerini, and Stefano Teso. Beyond topological self-explainable GNNs: A formal explainability perspective. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2045–2080. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/azzolin25a.html.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for AI safety – a review. *arXiv preprint arXiv:2404.14082*, 2024.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. In *Advances in Neural Information Processing Systems*, 2024.

Pietro Bongini, Monica Bianchini, and Franco Scarselli. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450:242–252, Aug 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.04.039. URL http://dx.doi.org/10.1016/j.neucom.2021.04.039.

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the Shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. doi: 10.1016/j.cor.2008.04.004.

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

Enyan Dai and Suhang Wang. Towards prototype-based self-explainable graph neural network. *ACM Transactions on Knowledge Discovery from Data*, 19(2):1–20, 2025.

Asim Kumar Debnath, Rosa L. Lopez de Compadre, Gargi Debnath, Alan J. Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, 34(2):786–797, 1991.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference on - WWW '19*. ACM Press, 2019. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *Conference on Language Modeling (COLM)*, 2024.

Jesse He, Helen Jenne, Max Vargas, Davis Brown, Gal Mishne, Yusu Wang, and Henry Kvinge. Minar: Mechanistic interpretability for neural algorithmic reasoning, 2026. URL https://arxiv.org/abs/2602.21442.

Lianzhe Huang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. Text level graph neural network for text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3444–3450, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1345. URL https://aclanthology.org/D19-1345.

Sergei Ivanov, Sergei Sviridov, and Evgeny V. Burnaev. Understanding isomorphism bias in graph data sets. In *Geometric Learning and Graph Representations ICLR Workshop*, 2019.

Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1), February 2021. doi: 10.1186/s13321-020-00479-8. URL https://doi.org/10.1186/s13321-020-00479-8.

Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, December 2004. ISSN 1520-4804.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4768–4777. Curran Associates Inc., 2017. ISBN 9781510860964.

Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Lió. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *ArXiv*, abs/2107.11889, 2021.

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

Mauparna Nandan, Soma Mitra, and Debashis De. Graphxai: a survey of graph neural networks (gnns) for explainable ai (xai). *Neural Computing and Applications*, 37(17):10949–11000, March 2025. ISSN 1433-3058. doi: 10.1007/s00521-025-11054-3. URL http://dx.doi.org/10.1007/s00521-025-11054-3.

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

Simone Palumbo, Alessio Ragno, Roberto Capobianco, Marc Plantevit, and Céline Robardet. *PLEX: Edge-Aware Logic-Based Self-Explainable Graph Neural Networks with Polyadic Predicates*, pp. 487–503. Springer Nature Switzerland, Sept 2026. ISBN 9783032376640. doi: 10.1007/978-3-032-37664-0_28. URL http://dx.doi.org/10.1007/978-3-032-37664-0_28.

Alessio Ragno, Biagio La Rosa, and Roberto Capobianco. Prototype-based interpretable graph neural networks. *IEEE Trans. Artif. Intell.*, 5(4):1486–1495, 2024. doi: 10.1109/TAI.2022.3222618.

Alessio Ragno, Marc Plantevit, and Céline Robardet. On logic-based self-explainable graph neural networks. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In Niels da Vitoria Lobo, Takis Kasparis, Fabio Roli, James T. Kwok, Michael Georgiopoulos, Georgios C. Anagnostopoulos, and Marco Loog (eds.), *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 287–297, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-89689-0.

Lloyd S Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games II*, pp. 307–317. Princeton University Press, Princeton, 1953.

Shriyank Somvanshi, Md Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi Dutta, and Subasish Das. Bridging the black box: A survey on mechanistic interpretability in AI. *ACM Computing Surveys*, 58(8), February 2026. doi: 10.1145/3787104.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Elouan Vincent, Julien Perez, Marc Plantevit, and Céline Robardet. *Optimal Transport Prototypes and Symbolic Logic Combination for Self-explainable Graph Neural Networks*, pp. 667–685. Springer Nature Switzerland, Sept 2026. ISBN 9783032376640. doi: 10.1007/978-3-032-37664-0_38. URL http://dx.doi.org/10.1007/978-3-032-37664-0_38.

Nikil Wale and George Karypis. Comparison of Descriptor Spaces for Chemical Compound Retrieval and Classification. In *International Conference on Data Mining - Workshops (ICDMW'06)*. IEEE, 2006.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. In *International Conference on Learning Representations (ICLR)*, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International conference on machine learning*, pp. 12241–12252. PMLR, 2021.

Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. Protgnn: Towards self-explaining graph neural networks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event*, pp. 9127–9135. AAAI Press, 2022. doi: 10.1609/AAAI.V36I8.20898.

# A Appendix

## A.1 Implementation details

Table 2: Values of $\tau$ (pass fraction) per dataset. Shapley: $\tau$ tuned per architecture; zero-ablation: best of $\{0.7, 0.8\}$ per architecture.

| | Shapley $\tau$ | | | Zero-ablation $\tau$ | | |
|---|---|---|---|---|---|---|
| Dataset | GAT | GCN | GIN | GAT | GCN | GIN |
| MUTAG | 0.5 | 0.3 | 0.3 | 0.7 | 0.8 | 0.7 |
| NCI1 | 0.5 | 0.7 | 0.3 | 0.7 | 0.7 | 0.7 |
| AIDS | 0.5 | 0.6 | 0.3 | 0.7 | 0.7 | 0.7 |
| Mutagenicity | 0.4 | 0.3 | 0.5 | 0.7 | 0.7 | 0.7 |

Table 3: Results on MUTAG. Pred. change ablation and activation are the necessity (excitatory neurons zeroed) and induction (neurons set to a high activation) prediction-change rates; Circuit size is the number of neurons (out of the 96 in layers 1–3) kept by the union of the circuits of the class ("$\sim$": approximate, several circuits are merged); Circuits is their number. Sel. "Sig" keeps only circuits with $p \leq 0.05$ in at least one necessity direction. Shapley uses $\tau$=0.5 (GAT), 0.3 (GCN), 0.3 (GIN); zero-ablation uses $\tau$=0.7 (GAT, GIN) and 0.8 (GCN) (best of $\{0.7, 0.8\}$ per architecture). "–" = no circuits.

| | | | Non-mutagenic | | | | Mutagenic | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Sel. | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) |
| GAT | Shapley | All | 92.3 | 97.5 | $\sim$67 | 14 | 94.0 | 67.7 | $\sim$71 | 14 |
| | | Sig. | 97.0 | 96.6 | $\sim$61 | 10 | 100.0 | 98.1 | $\sim$63 | 7 |
| | Zero-ablation | All | 26.7 | 80.9 | $\sim$68 | 12 | 67.1 | 59.2 | $\sim$71 | 14 |
| | | Sig. | 28.9 | 100.0 | $\sim$49 | 3 | 100.0 | 100.0 | $\sim$62 | 6 |
| GCN | Shapley | All | 47.1 | 84.5 | $\sim$75 | 15 | 94.9 | 87.2 | $\sim$73 | 13 |
| | | Sig. | 80.0 | 86.4 | $\sim$66 | 9 | 100.0 | 100.0 | $\sim$66 | 9 |
| | Zero-ablation | All | 22.7 | 80.7 | $\sim$69 | 15 | 100.0 | 32.9 | $\sim$65 | 13 |
| | | Sig. | 100.0 | 46.9 | $\sim$29 | 3 | – | – | – | 0 |
| GIN | Shapley | All | 90.3 | 82.5 | $\sim$55 | 12 | 44.3 | 59.0 | $\sim$66 | 17 |
| | | Sig. | 99.4 | 100.0 | $\sim$39 | 10 | 75.4 | 92.0 | $\sim$48 | 10 |
| | Zero-ablation | All | 9.1 | 20.1 | $\sim$56 | 12 | 81.3 | 64.1 | $\sim$60 | 17 |
| | | Sig. | 93.8 | 12.2 | $\sim$24 | 1 | 100.0 | 96.1 | $\sim$49 | 8 |

Table 4: Results on NCI1. Pred. change ablation and activation are the necessity (excitatory neurons zeroed) and induction (neurons set to a high activation) prediction-change rates; Circuit size is the number of neurons (out of the 96 in layers 1–3) kept by the union of the circuits of the class ("∼": approximate, several circuits are merged); Circuits is their number. Sel. "Sig" keeps only circuits with $p \leq 0.05$ in at least one necessity direction. Shapley uses $\tau$=0.5 (GAT), 0.7 (GCN), 0.3 (GIN); zero-ablation uses $\tau$=0.7 (best of $\{0.7, 0.8\}$ per architecture). "–" = no circuits.

| | | | Inactive | | | | Active | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Sel. | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) |
| GAT | Shapley | All | 75.2 | 67.3 | ∼73 | 12 | 16.4 | 36.1 | ∼77 | 15 |
| | | Sig. | 67.9 | 46.6 | ∼53 | 3 | 80.2 | 80.8 | ∼33 | 2 |
| | Zero-ablation | All | 77.6 | 46.2 | ∼74 | 15 | 9.1 | 23.3 | ∼72 | 17 |
| | | Sig. | 100.0 | 99.8 | ∼38 | 1 | 19.2 | 47.5 | ∼45 | 4 |
| GCN | Shapley | All | 73.1 | 68.8 | ∼75 | 13 | 50.3 | 53.0 | ∼73 | 12 |
| | | Sig. | 99.9 | 84.8 | ∼60 | 4 | 74.9 | 63.0 | ∼64 | 6 |
| | Zero-ablation | All | 77.8 | 48.2 | ∼68 | 13 | 15.6 | 43.3 | ∼68 | 12 |
| | | Sig. | 100.0 | 91.0 | ∼32 | 1 | – | – | – | 0 |
| GIN | Shapley | All | 53.7 | 42.1 | ∼79 | 17 | 21.3 | 59.9 | ∼77 | 15 |
| | | Sig. | 100.0 | 16.7 | ∼58 | 3 | 42.0 | 68.3 | ∼48 | 2 |
| | Zero-ablation | All | 77.4 | 41.0 | ∼69 | 17 | 17.7 | 67.7 | ∼70 | 15 |
| | | Sig. | 99.9 | 39.1 | ∼39 | 3 | 43.2 | 96.3 | ∼44 | 2 |

Table 5: Results on AIDS. Pred. change ablation and activation are the necessity (excitatory neurons zeroed) and induction (neurons set to a high activation) prediction-change rates; Circuit size is the number of neurons (out of the 96 in layers 1–3) kept by the union of the circuits of the class ("∼": approximate, several circuits are merged); Circuits is their number. Sel. "Sig" keeps only circuits with $p \leq 0.05$ in at least one necessity direction. Shapley uses $\tau$=0.5 (GAT), 0.6 (GCN), 0.3 (GIN); zero-ablation uses $\tau$=0.7 (best of $\{0.7, 0.8\}$ per architecture). "–" = no circuits.

| | | | Class 0 | | | | Class 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Sel. | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) |
| GAT | Shapley | All | 43.8 | 22.5 | ∼54 | 11 | 5.0 | 31.4 | ∼75 | 17 |
| | | Sig. | 92.3 | 34.4 | ∼23 | 3 | 19.9 | 39.7 | ∼42 | 3 |
| | Zero-ablation | All | 65.6 | 22.1 | ∼68 | 13 | 0.1 | 30.6 | ∼61 | 16 |
| | | Sig. | 100.0 | 36.0 | ∼42 | 2 | 0.8 | 27.8 | ∼16 | 2 |
| GCN | Shapley | All | 96.2 | 49.7 | ∼72 | 10 | 6.6 | 53.7 | ∼81 | 18 |
| | | Sig. | 100.0 | 82.7 | ∼63 | 5 | 6.8 | 66.4 | ∼77 | 16 |
| | Zero-ablation | All | 100.0 | 63.3 | ∼68 | 11 | 0.0 | 37.3 | ∼79 | 18 |
| | | Sig. | 100.0 | 94.4 | ∼56 | 4 | 0.0 | 84.4 | ∼31 | 2 |
| GIN | Shapley | All | 56.9 | 33.6 | ∼79 | 17 | 21.1 | 41.8 | ∼72 | 15 |
| | | Sig. | 100.0 | 72.1 | ∼50 | 6 | 30.1 | 59.4 | ∼59 | 8 |
| | Zero-ablation | All | 72.0 | 25.6 | ∼77 | 17 | 5.8 | 42.5 | ∼70 | 15 |
| | | Sig. | 100.0 | 27.5 | ∼50 | 3 | – | – | – | 0 |

Table 6: Results on Mutagenicity. Pred. change ablation and activation are the necessity (excitatory neurons zeroed) and induction (neurons set to a high activation) prediction-change rates; Circuit size is the number of neurons (out of the 96 in layers 1–3) kept by the union of the circuits of the class ("$\sim$": approximate, several circuits are merged); Circuits is their number. Sel. "Sig" keeps only circuits with $p \leq 0.05$ in at least one necessity direction. Shapley uses $\tau$=0.4 (GAT), 0.3 (GCN), 0.5 (GIN); zero-ablation uses $\tau$=0.7 (best of $\{0.7, 0.8\}$ per architecture). "–" = no circuits.

| | | | Mutagenic | | | | Non-mutagenic | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Method | Sel. | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) | Pred. change ablation (%) | Pred. change activation (%) | Circuit size | Circuits (#) |
| GAT | Shapley | All | 83.5 | 63.6 | $\sim$65 | 14 | 41.6 | 20.9 | $\sim$55 | 15 |
| | | Sig. | 98.5 | 72.4 | $\sim$47 | 7 | 92.2 | 50.6 | $\sim$12 | 1 |
| | Zero-ablation | All | 68.5 | 64.5 | $\sim$69 | 15 | 26.9 | 39.5 | $\sim$67 | 14 |
| | | Sig. | 100.0 | 91.9 | $\sim$38 | 3 | 53.1 | 58.1 | $\sim$37 | 3 |
| GCN | Shapley | All | 66.1 | 52.3 | $\sim$74 | 16 | 59.9 | 47.8 | $\sim$73 | 15 |
| | | Sig. | 99.8 | 66.1 | $\sim$30 | 4 | 78.4 | 53.9 | $\sim$48 | 4 |
| | Zero-ablation | All | 82.8 | 53.8 | $\sim$75 | 16 | 50.4 | 47.2 | $\sim$75 | 15 |
| | | Sig. | 99.9 | 74.0 | $\sim$62 | 8 | – | – | – | 0 |
| GIN | Shapley | All | 58.9 | 41.7 | $\sim$54 | 14 | 33.7 | 38.1 | $\sim$65 | 17 |
| | | Sig. | 69.1 | 51.5 | $\sim$26 | 3 | 19.4 | 56.4 | $\sim$9 | 1 |
| | Zero-ablation | All | 60.1 | 49.8 | $\sim$71 | 14 | 41.5 | 44.2 | $\sim$76 | 17 |
| | | Sig. | 82.7 | 33.0 | $\sim$39 | 3 | 70.9 | 61.2 | $\sim$45 | 3 |