GRAPH2RULE: LEARNING BEHAVIORAL RULES FROM TRANSFORMER COMPUTATION GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) exhibit distinct internal computation patterns across different input types, yet these patterns remain difficult to characterize due to complex interactions among distributed attention heads and feed-forward neurons. Our goal is to understand whether these component interactions contain behavior-specific patterns and whether these patterns can be expressed in an interpretable form. To achieve this, we propose GRAPH2RULE, a framework to represent the computation of a pretrained transformer as a directed weighted graph. In this graph, nodes represent selected attention heads and feed-forward neurons, while edges represent their forward interactions. We begin by testing whether these computation graphs contain sufficient information to distinguish behavioral classes without access to the original input text. Across three datasets and multiple transformer backbones, our graph-based classifiers achieve strong performance. Importantly, the graph representation and GNN classifier enable us to analyze the structure underlying these predictions via GNN explainability techniques. Specifically, we extract sample-level explanation subgraphs, mine recurring class-discriminative motifs from these subgraphs, and convert these motifs into interpretable logical rules. Our results show that transformer component interactions contain structured behavioral information that can be represented as graphs and translated into compact and human-interpretable symbolic rules.

1 INTRODUCTION

Transformer-based models Vaswani et al. (2017); Raffel et al. (2020); Grattafiori et al. (2024); Team et al. (2025); Brown et al. (2020) exhibit substantially different behaviors across inputs. For example, the same model may respond differently to harmful and benign prompts, or process human- and machine-generated text through different internal computations Arditi et al. (2024); Chen et al. (2025); Sclar et al. (2024). While such behavioral differences are readily observed from model outputs, considerably less is understood about how they emerge from interactions among the model's internal components. Transformer computation is distributed across many attention heads and feed-forward network (FFN) neurons, whose contributions and interactions vary with the input. This raises a fundamental question: *do different model behaviors correspond to systematically different patterns of interaction among internal components?* More importantly, if such patterns exist, can they be distilled into compact, human-interpretable descriptions of how transformer computation differs across behaviors?

Existing approaches provide complementary views of transformer internals. Model probing asks whether information about a property of interest is encoded in internal representations, typically by training a lightweight classifier on those representations Belinkov (2022). For example, previous work has shown properties such as truthfulness can exhibit linear structure in the representation space Marks & Tegmark (2024). However, probing primarily establishes whether information is present and provides limited insight into how interactions among individual attention heads and FFN neurons give rise to the observed behavior. Mechanistic interpretability studies these interactions more directly through circuit analysis Elhage et al. (2021), mostly using intervention-based techniques such as activation patching and path patching to establish causal influence of specific components and computational paths on model outputs Wang et al. (2023). These methods provide a more detailed view of internal computation, but can be computationally expensive and historically task- and model-specific, although recent work has begun to incorporate cross-task generalization

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.

He et al. (2025). Moreover, existing analyses typically identify mechanisms at the global level, rather than connecting sample-specific explanations to recurring interaction patterns that generalize across inputs. This leaves open how local explanations of transformer computation can be aggregated into compact global descriptions of behavior.

Addressing these introduces several challenges. First, transformer computation is distributed across thousands of attention heads and FFN neurons, so constructing a tractable representation requires identifying a compact set of components and interactions without discarding behaviorally relevant structure. Second, explanation of individual inputs can be very sparse and highly sample-specific Luo et al. (2020); Yuan et al. (2022), which makes it difficult to determine which interaction patterns recur consistently across behavior classes. Finally, even when recurring patterns are identified, the challenge is to determine how they combine into compact global descriptions that characterize the differences between behavioral classes Azzolin et al. (2022).

We address these challenges by introducing GRAPH2RULE, which represents the computation induced by each prompt as a graph. To keep this representation tractable, we retain a prompt-dependent subset of attention heads and FFN neurons in each layer according to component-specific scores. These selected components form the graph nodes, while directed edges encode their modeled forward interactions. Each graph is obtained from a single forward pass through a frozen transformer. This representation makes graph-classification and graph-explanation tools directly applicable to transformer internals. This also enables us to (i) train a graph neural network to classify the behavioral label, and (ii) apply graph explainability methods to identify which interactions the classifier relies on. To move beyond explanations that are sparse and specific to individual inputs, we aggregate explanations across training samples and search for recurring interaction motifs whose frequencies differ across behavioral classes. Finally, to move from these recurring motifs to a compact global description, we treat the selected motifs as concepts and combine their presence and absence into logical rules, which are evaluated on test samples to see how closely they reproduce the graph classifier's predictions.

Contributions. Our main contributions are as follows. *First,* we introduce a prompt-dependent graph representation of transformer computation in which selected attention heads and FFN neurons form nodes and their modeled forward interactions form directed weighted edges. *Second,* we develop an explanation pipeline that connects sample-level GNN explanations to recurring class-discriminative motifs and uses these motifs as concepts for constructing compact rules. *Third,* across three behavioral classification datasets and multiple transformer backbones, our classifiers operating only on the derived interaction graphs achieve high accuracies, demonstrating that substantial behavioral information is retained without access to the original text. Beyond classification, the discovered motifs yield compact rules with small-sized concepts and achieve high test fidelity to the graph classifier on test samples.

2 RELATED WORK

Model Probing. SAPLMA (Azaria & Mitchell, 2023) is exemplary of the model probing approach: a classifier—commonly a logistic regression model or a shallow MLP, referred to as a probe—is trained to evaluate statement truthfulness using the LLM's internal representations with high accuracy. Leveraging this technique, Marks & Tegmark (2024) shows that truthfulness corresponds to linear structures in the learned latent space. Discoveries have been made on other aspects of LLMs: their belief representations (Li et al., 2023), understanding of time and space (Gurnee & Tegmark, 2024) or political perspectives (Kim et al., 2025). Unsupervised methods for probing have also been explored (Burns et al., 2023; Huben et al., 2024).

Mechanistic Interpretability and Circuit Discovery. As an early foundational work in Mechanistic Interpretability, (Elhage et al., 2021) establish a mathematical framework for understanding the mechanisms underlying in-context learning by analyzing one- and two-layer attention-only transformers. Crucially, it identifies how particular behaviors arise from specific circuits — interpretable computational substructures within the transformer. Subsequent studies (Wang et al., 2023; Hanna et al., 2023; Heimersheim & Janiak, 2023) perform circuit localization on more complex models; however, rather than proposing a general framework, these works offer analytical case studies focused on a single task for a specific, moderate-sized model (typically GPT-2 Small). A shared technique among these papers is activation patching, which replaces specific node or edge activations

with corrupted values to measure their impact on the final output relative to a clean run. Addressing the lack of automation, (Conmy et al., 2023) introduces a pipeline to unify and automate circuit discovery, though it remains computationally expensive due to its continued reliance on activation patching. Most recently, (He et al., 2025) scales circuit discovery to much larger models (Llama 8B) and identifies circuits applicable across tasks by adopting a linear approximation of activation patching and optimizing metrics defined for cross-task generalization.

Contextualization of GRAPH2RULE. *Probing* is a top-down and correlation-based approach: it tests whether predefined human concepts are encoded in model representations, but cannot confirm if downstream layers actively utilize them. *Mechanistic Interpretability*—specifically circuit localization—is a bottom-up, causation-based approach that reverse-engineers low-level components and interactions to establish direct causal control over model output (Bereska & Gavves, 2024). Despite offering strong causal guarantees, Mechanistic Interpretability suffers from extreme computational costs and poor scalability. *Our method* GRAPH2RULE *bridges these two paradigms through the use of a spatial GNN as the probe.* Because spatial GNNs operate directly on the network's computational graph via message-passing mechanisms, we can directly identify representative subgraph motifs to explain the GNN's predictions and, by extension, the transformer's behavior. Consequently, our framework inherits the simplicity of training an auxiliary classifier without compromising the need to identify tangible, human-interpretable motifs for explaining model behavior.

3 PRELIMINARIES AND PROBLEM SETUP

3.1 TRANSFORMER COMPONENTS AND OUR COMPUTATION GRAPH

We consider a frozen pretrained transformer model M with L layers, model dimension d, H attention heads per layer, and a feed-forward width d_{ff}. For an input prompt p, no parameters of M are updated during graph construction. We treat each attention head and each FFN neuron as an atomic transformer component. Thus, layer ℓ contains the head set $\{h_{\ell,1}, \ldots, h_{\ell,H}\}$ and the neuron set $\{f_{\ell,1}, \ldots, f_{\ell,d_{\text{ff}}}\}$. Let $x_t^{(\ell)} \in \mathbb{R}^d$ denote the hidden representation of token position t at layer ℓ, and let $p_t^{(\ell,h)}$ denote the post-softmax attention distribution over key positions produced by head h in layer ℓ for query position t. We further let $a_{t,k}^{(\ell)}$ denote the intermediate activation of FFN neuron k at token position t. For a standard feed-forward block, the intermediate activation is $a^{(\ell)} = \text{act}\left(W_{\text{wi}}^{(\ell)} x^{(\ell)}\right)$. For gated FFNs, such as SwiGLU or GeGLU (Shazeer, 2020), $a^{(\ell)} = \text{act}\left(W_{\text{gate}}^{(\ell)} x^{(\ell)}\right) \odot W_{\text{up}}^{(\ell)} x^{(\ell)}$, where $\odot$ denotes element-wise multiplication. We denote $W_q^{(\ell,h)}$ and $W_o^{(\ell,h)}$ as the head-specific query and output projection blocks for head h in layer ℓ. Similarly, $W_{\text{wi},k}^{(\ell)}$ and $W_{\text{wo},k}^{(\ell)}$ denote the input-projection row and output-projection column associated with FFN neuron k. These quantities are later used for component scoring and edge weighting.

Our Computation Graph. We represent the selected transformer components and their modeled forward interactions as a directed weighted graph $G = (V, E, w, X)$, where V is the set of retained transformer components, E is the directed edge set, $w : E \to \mathbb{R}_{>0}$ assigns scalar weight to each edge and $X \in \mathbb{R}^{|V| \times F}$ is node-feature matrix. The row x_v denotes the feature vector associated with node v. The resulting graph should be viewed as a local abstraction of forward computation among selected components.

3.2 OUR PROBLEM FORMULATION

Let $\mathcal{D} = \{(p_i, y_i)\}_{i=1}^N$ denote a dataset with input prompts as p_i and binary behavioral labels as y_i such as, benign/harmful prompts, true/false statements, or human/ChatGPT generated answers. Let $\mathcal{M}$ denote a frozen pretrained transformer. Our goal is to determine whether the computation structure induced by an input contains behavior-specific patterns and, if so, whether these patterns can be summarized in a compact and interpretable form. We formulate this goal with three connected problems. First, we construct a graph representation of the computation induced by each prompt. Second, we test whether this graph alone contains sufficient information to distinguish the behavioral

classes. Finally, we aim to find recurring interaction patterns across sample-level explanations that differ between behavioral classes, and to compose these patterns into compact symbolic rules.

Problem 1: Computation graph construction. For each prompt p_i, we seek a mapping $\mathcal{F}$ that represents the computation induced in the transformer $\mathcal{M}$ as a directed weighted graph $G_i = \mathcal{F}(\mathcal{M}, p_i)$, where the nodes correspond to selected transformer components and the edges represent their modeled forward interactions.

Problem 2: Behavioral graph classification. Given the resulting graph dataset $\{(G_i, y_i)\}_{i=1}^{N}$, our goal is to build a graph classifier Φ_θ that predicts the behavioral label using only G_i, with $\hat{y}_i^{\mathrm{GNN}} = \arg\max_c \Phi_\theta(G_i)_c$. This determines whether the constructed computation graphs contain sufficient information to distinguish the behavioral classes.

Problem 3: Global rule discovery. Given a trained graph classifier, we seek recurring class-discriminative motifs from its sample-level local explanation graphs and treat the selected motifs $\mathcal{P} = \{P_1, \ldots, P_K\}$ as concepts. For sample i, their presence defines a binary concept vector $\mathbf{c}_i$, where $c_{ij} = \mathbb{I}[P_j \subseteq S_i^{\mathrm{exp}}]$. We then seek a compact Boolean rule that reproduces the GNN predictions $r^\star \approx \arg\min_{r \in \mathcal{R}} \sum_{i=1}^{N_{\mathrm{tr}}} \mathbb{I}\left[r(\mathbf{c}_i) \neq \hat{y}_i^{\mathrm{GNN}}\right]$. The resulting rule provides a global symbolic surrogate of the trained graph classifier and is evaluated by its fidelity on test graphs.

4 Our Proposed Method: Graph2Rule

We propose Graph2Rule, a framework that converts prompt-dependent transformer computation into a graph representation and then analyzes this representation at both the predictive and structural levels. The framework consists of three stages: graph construction, graph classification, and explanation-driven motif and rule discovery. Figure 1 shows the entire pipeline.

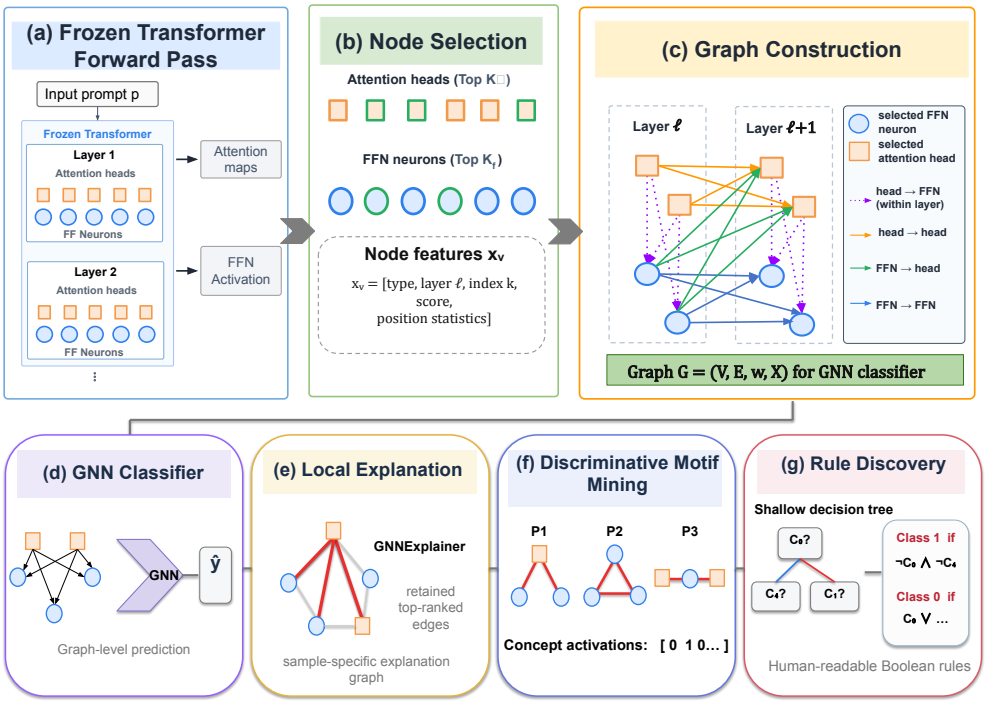


Figure 1: **Overall pipeline of Graph2Rule:** Given an input prompt, we extract prompt-dependent attention and FFN statistics from a frozen transformer, select representative components, and construct a directed weighted computation graph. A graph classifier predicts the behavioral class from this representation. The trained classifier is then analyzed through local graph explanations, discriminative motif mining, and concept-based rule discovery, producing compact Boolean rules that summarize recurring component-interaction patterns underlying the GNN predictions.

4.1 GRAPH CONSTRUCTION

For each input prompt, we construct a directed weighted graph $G = (V, E, w, X)$, where nodes represent selected transformer components and edges represent modeled forward interactions between them. The graph is extracted from a frozen pretrained transformer using a single forward pass, and no transformer parameters are updated. Since component selection and node features depend on the internal activations induced by the input, each prompt produces a different computation graph. The edge set is determined by the selected components, while the default edge weights are derived from the frozen model parameters.

Node selection. To obtain a compact graph, we extract only a subset of components from each layer using prompt-dependent scores. For an attention head h in layer ℓ, let $p_t^{(\ell,h)}$ denote the post-softmax attention distribution over key positions for query position t. To reduce the effect of attention sinks Xiao et al. (2024), we exclude attention to the first valid key position and renormalize the remaining probabilities. Let $\mathcal{T}$ denote the set of valid non-padding token positions, t_0 the first valid token position, and $\mathcal{T}_q = \mathcal{T} \setminus \{t_0\}$ as the retained query positions. Let $\tilde{p}_t^{(\ell,h)}$ denote the sink-excluded and renormalized attention distribution. We define attention concentration score as

$$s_{\text{attn}}^{(\ell,h)} = 1 - \frac{1}{|\mathcal{T}_q|} \sum_{t \in \mathcal{T}_q} \frac{\mathcal{H}\left(\tilde{p}_t^{(\ell,h)}\right)}{\log(|\mathcal{T}| - 1)}, \text{where } \mathcal{H}\left(\tilde{p}_t^{(\ell,h)}\right) = -\sum_j \tilde{p}_{t,j}^{(\ell,h)} \log \tilde{p}_{t,j}^{(\ell,h)}. \tag{1}$$

A larger value corresponds to more concentrated attention. For FFN neuron k in layer ℓ, we measure its average activation magnitude across valid token positions as $s_{\text{ffn}}^{(\ell,k)} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left| a_{t,k}^{(\ell)} \right|$. For each layer, we retain the top K_h attention heads and top K_f FFN neurons according to their scores.

Node features. Each retained node is assigned features consisting of its component type, layer index, component index, component score, and positional statistics over the input sequence. These include the maximum value, the center of mass over the positions, the late-token concentration ratio, peak position, and a sinusoidal encoding of the center-of-mass position. For attention heads, these quantities are computed from the attention distribution, while for FFN neurons they are computed from the activation profile. These features describe where and how strongly a component is activated; they do not include token identities or residual-stream hidden states.

Edge construction. We add directed edges according to the forward computation order of the transformer. Within each layer, we connect every retained attention head to every retained FFN neuron, since the attention block precedes the feed-forward block. Across consecutive layers $\ell \to \ell + 1$, we add head-to-head, FFN-to-head, and FFN-to-FFN edges. We omit direct cross-layer head-to-FFN edges to limit edge-type growth. Cross-layer influence is still represented through the retained head-to-head, FFN-to-head, and FFN-to-FFN connections, yielding a compact layered abstraction of forward interactions among the selected components.

Edge weighting. By default, edge weights are computed from the frozen transformer parameters. Let u and v denote the source and destination components, respectively. For attention heads, $W_q^{(v)}$ and $W_o^{(u)}$ denote the destination query block and source output block. For FFN neurons, $W_{wi,v}$ and $W_{wo,u}$ denote the destination input-projection row and source output-projection column. We define

$$w_{h \to h} = \left\| W_q^{(v)} W_o^{(u)} \right\|_F, \qquad w_{h \to f} = \left\| W_{wi,v} W_o^{(u)} \right\|_2, \tag{2}$$

and

$$w_{f \to h} = \left\| W_q^{(v)} W_{wo,u} \right\|_2, \qquad w_{f \to f} = \left| \langle W_{wi,v}, W_{wo,u} \rangle \right|. \tag{3}$$

For gated FFN backbones, $W_{wi,v}$ is taken from the up-projection, while the gated activation is used only for node scoring. The norm is chosen based on the interaction output: matrix-valued head-to-head interactions use the Frobenius norm, vector-valued head–FFN interactions use the l_2 norm, and scalar FFN-to-FFN interactions use the absolute value. Additional details are in Appendix A.3.1.

4.2 GRAPH CLASSIFICATION

The constructed graphs form a labeled graph-classification dataset $\{(G_i, y_i)\}_{i=1}^N$, where each graph $G_i = (V_i, E_i, w_i, X_i)$ corresponds to one input sample and inherits its behavioral label y_i. We train

a GNN to predict y_i from G_i. For the message-passing models, node representations are updated by aggregating information from neighboring transformer components and are then combined into a graph-level representation using global mean pooling. The pooled graph representation is passed to a classification head and optimized with cross-entropy loss. We evaluate GCN and GAT as message-passing architectures. We additionally evaluate an MLP baseline that operates on the same graph-derived node features but without message passing, allowing us to assess how much behavioral information is already captured by the selected components and their node features alone. Detailed architecture and training settings are provided in Appendix A.3.2. Beyond evaluating whether the constructed graphs contain behavior-discriminative information, the trained graph classifier also provides the predictive model that we subsequently explain to identify recurring interaction patterns.

4.3 Final Explanation Framework

After training the graph classifier, we analyze which transformer-component interactions support its predictions. The explanation framework consists of three stages: local graph explanation, discriminative subgraph mining, and global rule discovery.

Local graph explanations. For each computation graph, we employ GNNExplainer Ying et al. (2019) on the trained model to compute importance scores for nodes and edges for its prediction. GNNexplainer seeks a compact subgraph $G_S \subseteq G$ that maximizes the mutual information between the subgraph and the model prediction,

$$G_S^* = \arg\max_{G_S \subseteq G} \mathrm{MI}(Y; G_S) = \arg\min_{G_S \subseteq G} H(Y \mid G = G_S). \tag{4}$$

where $H(\cdot)$ denotes the Shannon entropy. Since GNNExplainer relies on stochastic optimization, we repeat the explanation procedure for multiple initializations. To make importance scores comparable across runs, node and edge masks are independently converted to normalized ranks. We take the average score across runs as the final importance score for each node and edge. For each input graph G_i, we retain the highest-ranked edges to form a sparse explanation subgraph $S_i^{\exp}$. This subgraph is not necessarily connected and may contain multiple disconnected components.

Discriminative subgraph mining. Because individual feed-forward network (FFN) neurons exhibit high activation sparsity across samples, we adopt a coarser structural representation during motif mining: attention heads retain their individual identities, whereas FFN neurons are grouped into deterministic, within-layer blocks. Specifically, for an FFN neuron $f_{\ell,k}$ at index k in layer ℓ, its block-level mapping $b(f_{\ell,k})$ is defined as $b(f_{\ell,k}) = \left(\ell, \lfloor \frac{k}{B} \rfloor\right)$, where B denotes the predefined block size. Hereafter, all explanation subgraphs are represented under this transformed identity space.

Let $\mathcal{S}$ denote the complete set of correctly classified training explanation subgraphs across all classes. To discover recurring patterns, we define the candidate space $\mathcal{P}$ as all connected subgraphs P that appear in at least one explanation graph $S_i \in \mathcal{S}$. We employ gSpan (Yan & Han, 2002) over $\mathcal{S}$ to extract the set of frequent motifs $\mathcal{P}^* \subseteq \mathcal{P}$: $\mathcal{P}^* = \{P \in \mathcal{P} \mid f(P) \geq \tau\}$, where $f(P)$ is the fraction of explanation subgraphs in $\mathcal{S}$ containing P, and $\tau \in (0, 1]$ is the minimum support threshold. For each motif $P \in \mathcal{P}^*$, let $f_c(P)$ denote the fraction of class-c explanation subgraphs in $\mathcal{S}$ containing P. We define its class-discriminative contrast as: $\Delta(P) = f_1(P) - f_0(P)$. A positive contrast indicates that P occurs more frequently in class 1, whereas a negative value indicates a stronger association with class 0. Finally, we rank the mined motifs by $|\Delta(P)|$ and retain the most class-discriminative patterns for downstream rule discovery.

Global rule discovery. Discriminative subgraph mining identifies recurring motifs and quantifies how strongly each motif is associated with classes through $|\Delta(P)|$. However, these motif-level statistics consider each pattern individually and do not reveal the relationships among multiple discriminative motifs. For example, several concepts may individually favor class 1 while another favors class 0, but the motif scores alone do not indicate whether the prediction depends on their conjunction, disjunction, or absence. We treat the motifs as binary concepts and learn a compact symbolic rule over their presence and absence, yielding a global, human-readable description.
Let $\{P_1, \ldots, P_K\}$ denote the K motifs retained after ranking by $|\Delta(P)|$. We treat each retained motif P_j as a concept C_j. For each sample i, the activation of concept C_j is defined as $c_j^{(i)} = \mathbb{I}(P_j \subseteq S_i^{\exp})$, where $c_j^{(i)} = 1$ if motif P_j is present in the explanation graph and 0 otherwise. Thus the resulting concept vector $\mathbf{c}_i = [c_1^{(i)}, \ldots, c_K^{(i)}]$ records which of the selected concepts are active for sample i.

Table 1: Classification performance across datasets, graph classifiers, and representative backbone models. Each cell reports mean ± standard deviation over four independent runs. Boldface denotes the best-performing backbone for each classifier and dataset. Complete results, including Gemma-1B, are reported in the Appendix A.4. Overall, the results show that the constructed graphs contain strong behavioral information across datasets and transformer backbones.

Metric	Method	WildJailbreak			GoT-Cities			HC3		
		T5	Llama-1B	Llama-3B	T5	Llama-1B	Llama-3B	T5	Llama-1B	Llama-3B
Accuracy	MLP	0.91 ± 0.01	0.92 ± 0.01	$\mathbf{0.93 \pm 0.01}$	0.55 ± 0.01	0.94 ± 0.01	$\mathbf{0.99 \pm 0.00}$	0.95 ± 0.00	0.98 ± 0.01	$\mathbf{0.99 \pm 0.00}$
	GCN	0.77 ± 0.01	$\mathbf{0.91 \pm 0.00}$	0.91 ± 0.00	0.51 ± 0.01	0.85 ± 0.01	$\mathbf{0.97 \pm 0.01}$	0.76 ± 0.01	0.85 ± 0.01	$\mathbf{0.93 \pm 0.01}$
	GAT	0.85 ± 0.02	$\mathbf{0.92 \pm 0.01}$	0.92 ± 0.00	0.51 ± 0.02	0.92 ± 0.02	$\mathbf{0.99 \pm 0.00}$	0.89 ± 0.01	0.94 ± 0.01	$\mathbf{0.97 \pm 0.01}$
Macro-F1	MLP	0.91 ± 0.01	0.92 ± 0.01	$\mathbf{0.93 \pm 0.01}$	0.55 ± 0.01	0.94 ± 0.01	$\mathbf{0.99 \pm 0.00}$	0.95 ± 0.00	0.98 ± 0.01	$\mathbf{0.99 \pm 0.00}$
	GCN	0.77 ± 0.01	$\mathbf{0.91 \pm 0.00}$	0.91 ± 0.00	0.51 ± 0.01	0.85 ± 0.01	$\mathbf{0.97 \pm 0.01}$	0.76 ± 0.01	0.85 ± 0.01	$\mathbf{0.93 \pm 0.01}$
	GAT	0.85 ± 0.02	$\mathbf{0.92 \pm 0.01}$	0.92 ± 0.00	0.50 ± 0.02	0.92 ± 0.02	$\mathbf{0.99 \pm 0.00}$	0.89 ± 0.01	0.94 ± 0.01	$\mathbf{0.97 \pm 0.01}$

We fit a shallow decision tree (Breiman et al., 1984) on the training concept vectors using the predictions of the original GNN as targets. Using the GNN predictions rather than the ground-truth labels allows the rule learner to summarize the behavior of the graph classifier itself. The tree captures conditional relationships among motif presence and absence: each root-to-leaf path corresponds to a conjunction of concept conditions. These paths are subsequently simplified into Boolean rules, providing a compact symbolic description of how combinations of the discovered motifs characterize the classifier's decisions.

5 Experiments

Datasets. We evaluate our framework on three binary classification datasets: WildJailbreak, GoT-Cities, and HC3. For each dataset, we use a fixed train–test split and repeat the graph-classifier training with multiple random seeds, reporting the mean and standard deviation across runs. Further details on the datasets and split construction are provided in Appendix A.1.

Transformer backbones. We construct graphs from four frozen pretrained transformer backbones: T5-small, Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, and Gemma-3-1B-IT, covering both encoder–decoder and decoder-only architectures. For T5-small, we construct graphs from the encoder layers, whereas for the decoder-only backbones we use the decoder layers. No transformer parameters are updated during graph construction. See Appendix A.2 for details.

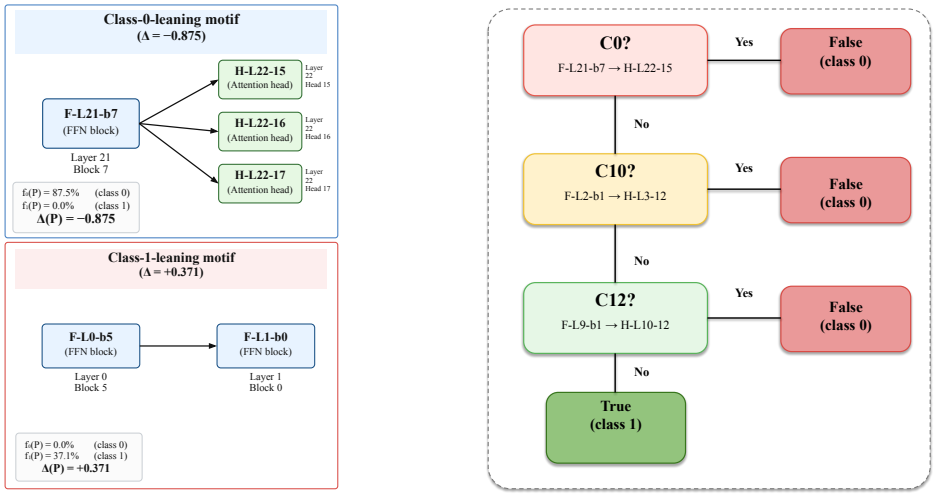
Graph construction and classification. We retain the top $K_h = 4$ attention heads and $K_f = 64$ FFN neurons per layer and use the pretrained-parameter edge weighting described in Section 4.1. We evaluate MLP, GCN, and GAT classifiers on the resulting graphs. We report accuracy, precision, recall, and macro-F1. Complete implementation and training details are provided in Appendix A.3.

5.1 Can transformer graphs distinguish behavioral classes?

Table 1 reports the classification performance across datasets, graph classifiers, and transformer backbones. The results show that transformer graphs contain substantial information about the behavioral class of the input. Across the three datasets, the best-performing settings achieve accuracies of 0.93 on WildJailbreak, 0.99 on GoT-Cities, and 0.99 on HC3. Performance depends strongly on the underlying transformer backbone. The Llama backbones generally perform strongest, with Llama-3B reaching 0.99 accuracy on GoT-Cities with both MLP and GAT. In contrast, T5 and Gemma generally underperform the Llama backbones. Interestingly, the MLP is competitive with, and in several settings outperforms, the message-passing GNNs. This indicates that the prompt-dependent component selection and node features alone already encode substantial class information. At the same time, GAT achieves strong performance across the Llama backbones.

5.2 Local Explanations and Motif Discovery

GNNExplainer setup. We apply GNNExplainer to each trained graph classifier to identify the most important component interactions to its predicted class. For each graph, we run the explainer with two random initializations for 100 optimization epochs using a learning rate of 10^{-2}. The resulting node and edge masks are independently converted to normalized ranks and averaged across the two runs. We retain the top 120 edges to form the local explanation graph for each sample.



(a) Class-discriminative motifs example. (b) Boolean rules example.

Figure 2: Representative class-discriminative motifs and Boolean rule from GoT-Cities using GAT with Llama-3B. The motifs are mined from training explanation graphs, and the selected concepts are combined into a compact Boolean rule that reproduces the GNN prediction.

Subgraph mining setup. We then search for recurring interaction motifs across the correctly classified training explanations using gSpan. Attention heads retain their individual identities, while FFN neurons are grouped into deterministic within-layer blocks of 1024 neurons. We use a minimum occurrence frequency of 10% and restrict mined motifs to at most six nodes. Motif discovery is performed only on the training explanations; the test explanations are not used during mining.

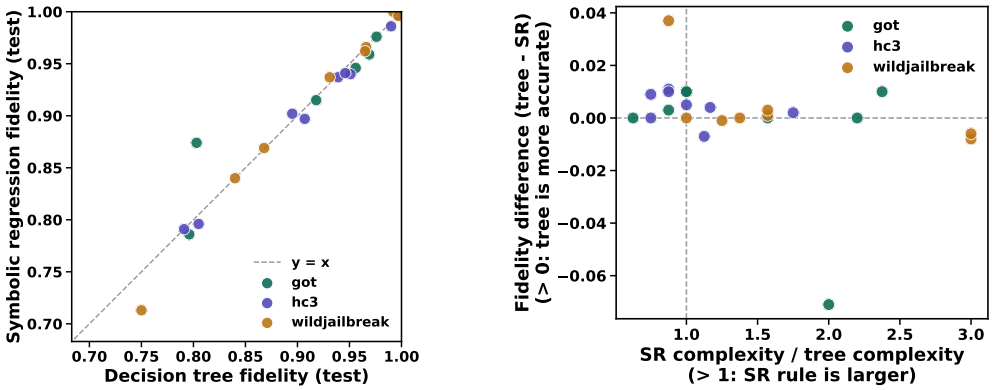
Representative motifs. Figure 2a shows representative class-discriminative motifs from GoT-Cities using GAT with Llama-3B. Here, F denotes an FFN block, H denotes an attention head, and $b0$, $b1$, etc. denote deterministic within-layer FFN block indices. The quantity $f_c(P)$ denotes the fraction of class-c explanation graphs containing motif P, and $\Delta(P) = f_1(P) - f_0(P)$ denotes the class-discriminative contrast. A class-0-leaning motif consists of an FFN block in layer 21, branching to attention heads 15, 16, and 17 in layer 22. This motif appears in 87.5% of class-0 explanations and is absent from class-1 explanations, yielding $\Delta = -0.875$. In contrast, the class-1-leaning motif F-L0-b5 $\rightarrow$ F-L1-b0 appears in 37.1% of class-1 explanations and is absent from class-0 explanations, yielding $\Delta = +0.371$. Examples from other datasets are provided in the Appendix A.4.2.

5.3 Global Rule Discovery

Rule construction. We next examine whether the recurring motifs can be combined into rules that reproduce the GNN's prediction. We retain the 15 most class-discriminative motifs, ranked by $|\Delta(P)|$, and treat them as candidate concepts. For each explanation graph, a concept is active if its corresponding motif is present and inactive otherwise. Because our goal is to explain the GNN rather than train a separate classifier, we fit a depth-3 decision tree on the concepts to reproduce the GNN's predictions on the training set. The resulting decision paths are then simplified into logical rules based on motif presence and absence. Motifs are mined from the training explanations, the resulting motifs are used to define concepts, and the decision tree is fit on the training split.

Representative rule. Figure 2b shows a representative rule from GoT-Cities using GAT with Llama-3B. The simplified rule depends on three concepts: C_0, corresponding to F-$L21$-$b7 \rightarrow H$-$L22$-15; C_{10}, corresponding to F-$L2$-$b1 \rightarrow H$-$L3$-12, and C_{12}, corresponding to F-$L9$-$b1 \rightarrow H$-$L10$-12. The resulting rule predicts class 1 when all three concepts are absent, i.e., $\hat{y}_{\text{rule}} = 1$ if $\neg C_0 \wedge \neg C_{10} \wedge \neg C_{12}$, and class 0 otherwise. Although multiple candidate motifs are provided to the rule learner, the final prediction can be summarized using only a few recurring component interactions.

Test Fidelity. We evaluate the rules using *fidelity*, the fraction of test graphs for which the logical rule prediction agrees with the GNN classifier: $\text{Fid} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{I}\left[\hat{y}^{(i)}_{\text{rule}} = \hat{y}^{(i)}_{\text{GNN}}\right]$. For the representative GoT-Cities example Figure 2, the three-concept rule achieves a test fidelity of 0.997. Table 2 summarizes rule complexity and test fidelity across datasets and transformer backbones.



(a) Test fidelity, decision tree vs. symbolic regression. Points below it indicate the tree is more accurate.

(b) Rule complexity ratio vs. fidelity difference, per run. Most runs fall right of the vertical line, indicating SR requires a larger expression.

Figure 3: Decision tree vs. symbolic regression as the rule-fitting method over mined concepts, evaluated across three datasets and the best-performing GAT-Llama backbones.

5.4 Robustness and Structural Analysis

We focus these analyses on GAT with Llama-1B and Llama-3B.

Choice of Rule Learner. To assess whether rule fidelity depends on the choice of symbolic learner, we compare decision tree with symbolic regression on the same 15 mined concepts and GNN predictions as targets. The symbolic regression search is restricted to Boolean-compatible operators and a comparable complexity budget. Figure 3 shows that both approaches achieve comparable test fidelity, while symbolic regression generally produces larger expressions. Also, decision-tree is effectively instantaneous in our experiment, whereas symbolic regression requires about 16–18 seconds per run. We therefore consider the decision tree as the default rule learner, as it provides comparable fidelity with interpretable Boolean rules.

Table 2: Global rule statistics and test fidelity. Rule size denotes the number of unique concepts appearing in the Boolean rule and is reported as median (range) over four independent runs. Test fidelity is reported as mean ± standard deviation.

Dataset	Backbone	Rule size	Test fidelity
WildJailbreak	Llama-1B	3 (3–5)	0.95 ± 0.06
	Llama-3B	3 (3–3)	0.88 ± 0.08
GoT-Cities	Llama-1B	4.5 (3–5)	0.92 ± 0.07
	Llama-3B	3.5 (3–5)	0.94 ± 0.08
HC3	Llama-1B	3 (3–4)	0.94 ± 0.04
	Llama-3B	3 (3–3)	0.87 ± 0.07

Additional Results. We additionally measure the consistency of the motif structure across independent GNN training runs and structural composition of the mined motifs. In four of the six dataset–backbone settings, the same dominant layer pair is recovered in three of four runs. Motif topology varies across tasks and backbones: divergent motifs dominate four of the six configurations. Complete stability and motif results are provided in Appendix A.4.3

6 Conclusion

We have introduced Graph2Rule, a framework that represents prompt-induced transformer computation as directed weighted graphs and translates recurring graph patterns into symbolic rules. Across three classification tasks and multiple transformer backbones, classifiers operating only on the constructed graphs achieve strong performance, showing that the selected transformer components and their interactions retain substantial behavior-discriminative information. By explaining the trained GNNs and mining recurring motifs from their training explanations, we have identified class-discriminative interaction patterns that can be combined into small Boolean rules with high test fidelity. Additional analyses show that recurring layer-level structures can remain consistent across training seeds, while motif topology and edge composition vary across behavioral tasks and model backbones, with Llama-3B exhibiting particularly consistent divergent motif structures. Overall, these results show that transformer computation graphs can capture behavior-specific internal structures that can be summarized as compact human-interpretable rules.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, clean and reformat dataset, and to formulate mathematical claims. We have not used generative AI tools for the following tasks: generating synthetic data sets, helping develop theoretical models or conceptual frameworks, proposing or refine hypotheses, designing or providing feedback on research methodology or experiments, and interpreting results. The following tasks are not applicable to this work: providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, assisting with translation, and supporting qualitative and thematic data analysis. Additionally, we used generative AI tools for the tasks: suggesting experimental parameters, sourcing/searching for information, and editing a research paper to improve readability. We have reviewed all AI-assisted work. All AI-assisted text and code were carefully reviewed, verified, and, where necessary, modified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available datasets and does not involve new human-subject data collection. The WildJailbreak dataset contains harmful and adversarial prompts, which are used solely to study differences in transformer computation and are not used to develop or facilitate harmful behavior. All experiments were conducted for research purposes using publicly available pretrained models and datasets.

REPRODUCIBILITY STATEMENT

The experimental setup and implementation details needed to reproduce our results are documented throughout the paper and appendix. Dataset construction and fixed train–test splits are described in Appendix A.1, transformer backbones in Appendix A.2, graph construction and feature extraction in Appendix A.3.1, and graph-classifier architectures and training settings in Appendix A.3.2. Additional classification, motif, rule-discovery, and analysis results are provided in Appendices A.4. The complete source code, including graph extraction, classifier training, explanation, motif mining, and rule-discovery pipelines, together with scripts for reproducing the reported experiments, will be released in a public code repository. For now, we have added our code to this anonymous code repository: `https://anonymous.4open.science/r/graph2rule-245C/readme.md`

REFERENCES

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. *Advances in Neural Information Processing Systems*, 37:136037–136083, 2024.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, 2023.

Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. *arXiv preprint arXiv:2210.07147*, 2022.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, March 2022. doi: 10.1162/coli_a_00422. URL `https://aclanthology.org/2022.cl-1.7/`.

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

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Xin Chen, Junchao Wu, Shu Yang, Runzhe Zhan, Zeyu Wu, Ziyang Luo, Di Wang, Min Yang, Lidia S Chao, and Derek F Wong. Repreguard: Detecting llm-generated text by revealing hidden representation patterns. *Transactions of the Association for Computational Linguistics*, 13:1812–1831, 2025.

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

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597*, 2023.

Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jE8xbmvFin.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060, 2023.

Yinhan He, Wendy Zheng, Yushun Dong, Yaochen Zhu, Chen Chen, and Jundong Li. Towards global-level mechanistic interpretability: A perspective of modular circuits of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Stefan Heimersheim and Jett Janiak. A circuit for python docstrings in a 4-layer attention-only transformer. In *Alignment Forum*, 2023.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, et al. Wildteaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. *Advances in Neural Information Processing Systems*, 37:47094–47165, 2024.

Junsol Kim, James Evans, and Aaron Schein. Linear representations of political perspective emerge in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=rwqShzb9li.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=DeG07_TcZvT.

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.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=aajyHYjjsk.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024.

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

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 721–724. IEEE, 2002.

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, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5): 5782–5799, 2022.

A Appendix

A.1 Datasets

We use the three publicly available datasets, summarized in Table 3. For each dataset, we construct a single train–test split and use the same split across all model configurations. We repeat each experiment with four random seeds and report the mean and standard deviation across runs.

- WildJailbreak (Jiang et al., 2024): This is a large scale dataset containing 262K vanilla and adversarial prompts, including benign and harmful queries. While harmful queries carry malicious intent, benign queries are phrased similarly, but do not contain any harmful intent. We use the vanilla benign and vanilla harmful subsets for binary classification. After preprocessing and retaining unique samples, our fixed split contains 2,983 training samples and 743 test samples.
- Human ChatGPT Comparison Corpus (HC3)(Guo et al., 2023): We use the English subset of HC3, which contain questions from open-domain QA, finance, medicine, computer science and Reddit ELI5. It includes both human-written and ChatGPT-generated answers. We select one human answer and one ChatGPT answer for each question, and split the data at the question level so that no question appears in both the training and test sets. The resulting fixed split contains 3,110 training samples and 798 test samples.
- GoT-Cities (Marks & Tegmark, 2024): We use the Cities dataset from *The Geometry of Truth*, which consists of simple factual statements following the template "The city of [city] is in [country]." Each city is paired with one true statement using the correct country and one false statement using an incorrect one, yielding a balanced true/false classification task. Our fixed split contains 1,176 training samples and 294 test samples, with the test set evenly divided between true and false examples.

Table 3: Summary of the datasets used in our experiments.

Dataset	Classes	Train	Test
WildJailbreak	Benign / Harmful	2983	743
HC3	Human / ChatGPT	3110	798
GoT-Cities	True / False	1176	294

A.2 Transformer backbones

In our experiments, we utilize *four transformer backbones* to evaluate performance across different architectural paradigms, specifically comparing encoder-decoder and decoder-only structures.

As **Encoder-Decoder architecture** we use **T5-small** which is a standard sequence-to-sequence transformer baseline that utilizes both an encoder to process the input and a decoder to generate the response. The **Decoder-Only architectures** are as follows.

- **Gemma-3-1B-It**: A lightweight, instruction-tuned autoregressive model from Google.
- **Llama-3.2-1B-Instruct**: An instruction-optimized autoregressive model from Meta.
- **Llama-3.2-3B-Instruct**: An instruction-optimized autoregressive model from Meta.

A.3 Additional Experiment Details

A.3.1 Graph Construction

We construct one computation graph for each input sample using a single forward pass through a frozen pretrained transformer. Inputs are tokenized with truncation and padding to a maximum sequence length of 256 and processed in batches of 8. Padding positions are removed before computing component statistics. For decoder-only models, we explicitly extract per-head attention probabilities during the forward pass. For gated FFN architectures, we compute the intermediate activation by combining the gate and up-projection branches as described in Section 3.

Table 4: Graph size for each transformer backbone. The number of nodes and edges is fixed by the backbone depth as $K_h = 4$ and $K_f = 64$ components are retained per layer.

Backbone	Layers L	Nodes $\lvert V\rvert$	Edges $\lvert E\rvert$
T5-small encoder	6	408	23,376
Llama-3.2-1B	16	1,088	69,616
Gemma-3-1B	26	1,768	115,856
Llama-3.2-3B	28	1,904	125,104

We exclude the first-token attention sink when scoring attention heads. For each layer, we retain the $K_h = 4$ highest-scoring attention heads and $K_f = 64$ highest-scoring FFN neurons. Node features include the component type, layer and component indices, component score, positional statistics, and a 32-dimensional sinusoidal encoding of the activation center of mass.

For each retained component, the positional statistics summarize how its activity is distributed across the input sequence. We record the maximum activation magnitude, the normalized center of mass of the activation profile, the fraction of activity concentrated in the final 30% of the sequence, and the position of the maximum activation. For attention heads, these statistics are computed from the sink-excluded and renormalized attention profile averaged over query positions, while for FFN neurons they are computed from the neuron activation profile across tokens. The normalized center of mass is additionally represented using a 32-dimensional sinusoidal positional encoding. No token identities or residual-stream hidden states are included in the node features.

We retain within-layer head-to-FFN edges and consecutive-layer head-to-head, FFN-to-head, and FFN-to-FFN edges. Cross-layer head-to-FFN edges are not included. Unless otherwise stated, edge weights are computed from the frozen pretrained parameters using the structural weighting. Every input therefore produces one directed weighted graph with its corresponding dataset label.

Graph size. Since we retain $K_h = 4$ attention heads and $K_f = 64$ FFN neurons per transformer layer, each layer contributes $N_{\text{layer}} = K_h + K_f = 68$ nodes. Thus, for a backbone with L transformer layers, the total number of nodes is $|V| = L(K_h+K_f)$. The edge count follows directly from the construction in Section 4.1. Within each layer, all retained heads are connected to all retained FFN neurons, giving K_hK_f edges. Across each pair of consecutive layers, we include head-to-head, FFN-to-head, and FFN-to-FFN connections, giving $K_h^2 + K_fK_h + K_f^2$ edges per layer boundary. Therefore,

$$|E| = LK_hK_f + (L-1)\left(K_h^2 + K_fK_h + K_f^2\right).$$

With $K_h = 4$ and $K_f = 64$, this becomes

$$|V| = 68L, \qquad |E| = 256L + 4368(L-1).$$

Table 4 lists the resulting sizes. Although the graph size is fixed for a given backbone, the identities of the retained components and their node features are prompt-dependent, so different inputs produce different computation graphs.

A.3.2 Graph Classifier Training

We evaluate MLP, GCN, and GAT graph classifiers. The GCN and GAT models use two message-passing layers followed by global mean pooling and a linear classification head. The GAT uses four attention heads in each graph attention layer. The hidden representation has dimension 128.

The training set of each run is further divided into training and validation subsets using a stratified split, with 15% of the training samples used for validation. Models are optimized using AdamW with a learning rate of 10^{-3} and weight decay of 10^{-4}. We use a batch size of 16, dropout of 0.2, and train for 50 epochs. The checkpoint achieving the highest validation macro-F1 is used for test evaluation. We report accuracy, precision, recall, and macro-F1 on the held-out test set. All reported results are summarized as mean $\pm$ standard deviation across the four independent runs.

A.4 Additional Experimental results

A.4.1 Detailed Classification Results

Tables 5–7 report the complete classification results, including Gemma-1B, across the three datasets and random seeds.

WildJailbreak. The three classifiers achieve their strongest performance with the Llama backbones. MLP reaches its highest accuracy of 0.93 ± 0.01 with Llama-3B, while GCN reaches 0.91 ± 0.00 with both Llama-1B and Llama-3B models. GAT performs best with Llama-1B, obtaining 0.92 ± 0.01 accuracy. In comparison, Gemma-1B gives lower performance, with accuracies between 0.59 and 0.66.

GoT-Cities. GoT-Cities shows the strongest dependence on the transformer backbone. Graphs extracted from T5 and Gemma-1B yield performance close to chance across all three classifiers. In contrast, the Llama backbones provide substantially more discriminative graph representations. With Llama-1B, MLP and GAT achieve 0.94 ± 0.01 and 0.92 ± 0.02 accuracy, respectively. With Llama-3B, both MLP and GAT reach 0.99 accuracy, whereas GCN reaches 0.97 ± 0.01.

HC3. HC3 gives the balanced overall classification performance. MLP achieves 0.95 ± 0.00, 0.98 ± 0.01, and 0.99 ± 0.00 accuracy for T5, Llama-1B, and Llama-3B, respectively. GAT also improves with the Llama backbones, reaching 0.97 ± 0.01 with Llama-3B, while GCN reaches 0.93 ± 0.01. Gemma-1B produces lower and relatively similar performance across the three classifiers, with accuracies between 0.68 and 0.70.

Table 5: Complete classification results on WildJailbreak. Each entry reports mean $\pm$ standard deviation over four runs.

Method	Backbone	Accuracy	Precision	Recall	Macro-F1
MLP	T5	0.91 ± 0.01	0.91 ± 0.01	0.91 ± 0.01	0.91 ± 0.01
	Llama-1B	0.92 ± 0.01	0.92 ± 0.00	0.92 ± 0.01	0.92 ± 0.01
	Llama-3B	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$
	Gemma-1B	0.59 ± 0.05	0.59 ± 0.05	0.59 ± 0.05	0.58 ± 0.05
GCN	T5	0.77 ± 0.01	0.77 ± 0.01	0.77 ± 0.01	0.77 ± 0.01
	Llama-1B	$\mathbf{0.91 \pm 0.00}$	$\mathbf{0.91 \pm 0.00}$	$\mathbf{0.91 \pm 0.00}$	$\mathbf{0.91 \pm 0.00}$
	Llama-3B	0.91 ± 0.00	0.91 ± 0.00	0.91 ± 0.00	0.91 ± 0.00
	Gemma-1B	0.66 ± 0.01	0.67 ± 0.01	0.66 ± 0.01	0.66 ± 0.01
GAT	T5	0.85 ± 0.02	0.85 ± 0.02	0.85 ± 0.02	0.85 ± 0.02
	Llama-1B	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$
	Llama-3B	0.92 ± 0.00	0.92 ± 0.00	0.92 ± 0.00	0.92 ± 0.00
	Gemma-1B	0.61 ± 0.03	0.62 ± 0.03	0.61 ± 0.03	0.61 ± 0.03

A.4.2 Additional Motif and Rule Examples

Figure 4 and 5 show additional representative motif and rule examples for WildJailbreak and HC3, respectively, using GAT with Llama-3B. We use the same notation and definitions as in Sections 5.2–5.3.

A.4.3 Additional Analyses results

Stability Across Runs and Backbones To assess the stability of the discovered motifs across independent GNN training runs, we characterize where the discriminative motif weight is concentrated in the transformer. For each run, we take the 15 motifs with the largest $|\Delta(P)|$ and assign each motif's $|\Delta(P)|$ weight to the layer pair spanned by its edges. For motifs spanning multiple layer pairs, the weight is divided equally among the corresponding pairs. We then identify the layer pair receiving the largest total weight and report its fraction of the total motif weight. Figure 6 reports the dominant layer pair for each dataset, backbone, and random seed. In four of the six dataset–backbone settings, the same dominant layer pair is recovered in three of the four runs. In particular,

Table 6: Complete classification results on GoT-Cities. Each entry reports mean ± standard deviation over four runs.

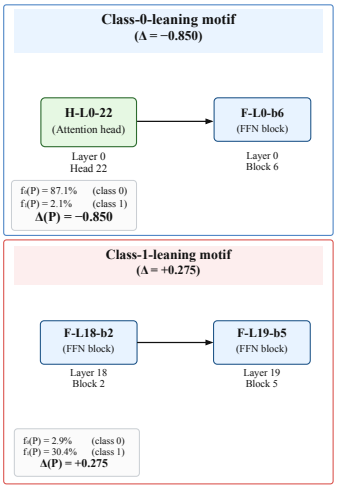
Method	Backbone	Accuracy	Precision	Recall	Macro-F1
MLP	T5	0.55 ± 0.01	0.56 ± 0.01	0.55 ± 0.01	0.55 ± 0.01
	Llama-1B	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01
	Llama-3B	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$
	Gemma-1B	0.49 ± 0.02	0.49 ± 0.02	0.49 ± 0.02	0.49 ± 0.02
GCN	T5	0.51 ± 0.01	0.51 ± 0.01	0.51 ± 0.01	0.51 ± 0.01
	Llama-1B	0.85 ± 0.01	0.85 ± 0.01	0.85 ± 0.01	0.85 ± 0.01
	Llama-3B	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$
	Gemma-1B	0.51 ± 0.02	0.51 ± 0.02	0.51 ± 0.02	0.50 ± 0.02
GAT	T5	0.51 ± 0.02	0.51 ± 0.02	0.51 ± 0.02	0.50 ± 0.02
	Llama-1B	0.92 ± 0.02	0.92 ± 0.02	0.92 ± 0.02	0.92 ± 0.02
	Llama-3B	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$
	Gemma-1B	0.51 ± 0.01	0.52 ± 0.01	0.51 ± 0.01	0.51 ± 0.01

Table 7: Complete classification results on HC3. Each entry reports mean ± standard deviation over four runs.

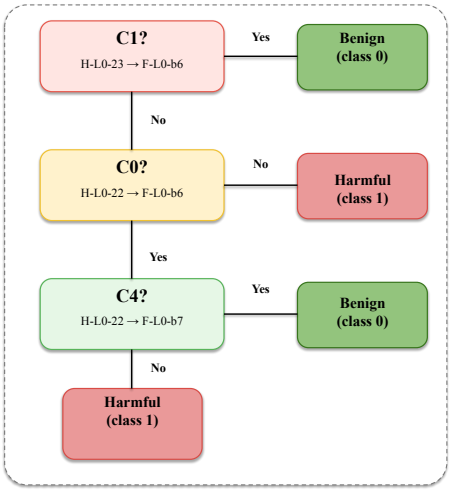
Method	Backbone	Accuracy	Precision	Recall	Macro-F1
MLP	T5	0.95 ± 0.00	0.95 ± 0.00	0.95 ± 0.00	0.95 ± 0.00
	Llama-1B	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01
	Llama-3B	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$
	Gemma-1B	0.68 ± 0.05	0.69 ± 0.04	0.68 ± 0.05	0.68 ± 0.05
GCN	T5	0.76 ± 0.01	0.76 ± 0.01	0.76 ± 0.01	0.76 ± 0.01
	Llama-1B	0.85 ± 0.01	0.85 ± 0.01	0.85 ± 0.01	0.85 ± 0.01
	Llama-3B	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$
	Gemma-1B	0.70 ± 0.01	0.72 ± 0.01	0.70 ± 0.01	0.70 ± 0.01
GAT	T5	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01	0.89 ± 0.01
	Llama-1B	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01
	Llama-3B	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$	$\mathbf{0.97 \pm 0.01}$
	Gemma-1B	0.70 ± 0.01	0.71 ± 0.01	0.70 ± 0.01	0.70 ± 0.00

GoT-Cities with Llama-1B repeatedly emphasizes layers 1–2, HC3 with Llama-3B emphasizes the final layer, WildJailbreak with Llama-1B emphasizes layers 14–15, and WildJailbreak with Llama-3B predominantly emphasizes layer 0. These results suggest that the dominant layer-level structure varies across behavioral tasks and model backbones, while remaining consistent across different seed runs within several dataset–backbone settings.

Motif Structure. We further analyze the topology and edge composition of the motifs mined from the training explanation graphs. We group multi-edge motifs into divergent, convergent, chain, and mixed structures according to their connectivity pattern, and separately summarize the prevalence of FFN→FFN, FFN→head, head→FFN, and head→head edges. Figure 7 shows that divergent branching structures dominate most dataset–backbone settings. GoT-Cities is strongly divergent for both Llama-1B and Llama-3B, while WildJailbreak with Llama-3B is entirely divergent. In contrast, HC3 with Llama-1B is dominated by convergent motifs. The edge composition also varies across settings: GoT-Cities with Llama-1B is primarily FFN→FFN, whereas several other settings are dominated by FFN→head or head→FFN interactions. This analysis considers only motifs with at least two edges, since single-edge patterns have no topology. For one GoT-Cities/Llama-3B run, gSpan returned only single-edge patterns, so no multi-edge motif was available for topology classification. These results suggest that motif topology depends on both the behavioral task and model backbone, with Llama-3B exhibiting a particularly consistent divergent structure across tasks.

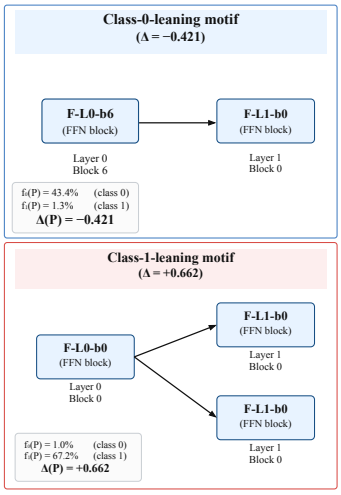


(a) Class-discriminative motifs example.

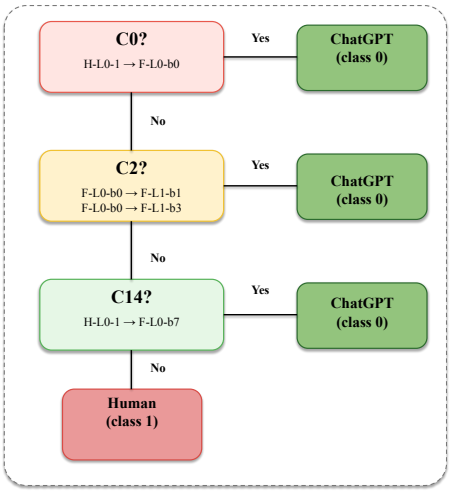


(b) Boolean rules example.

Figure 4: Representative class-discriminative motifs and Boolean rule from WildJailbreak using GAT with Llama-3B. The motifs are mined from training explanation graphs, and the selected concepts are combined into a compact Boolean rule that reproduces the GNN prediction.

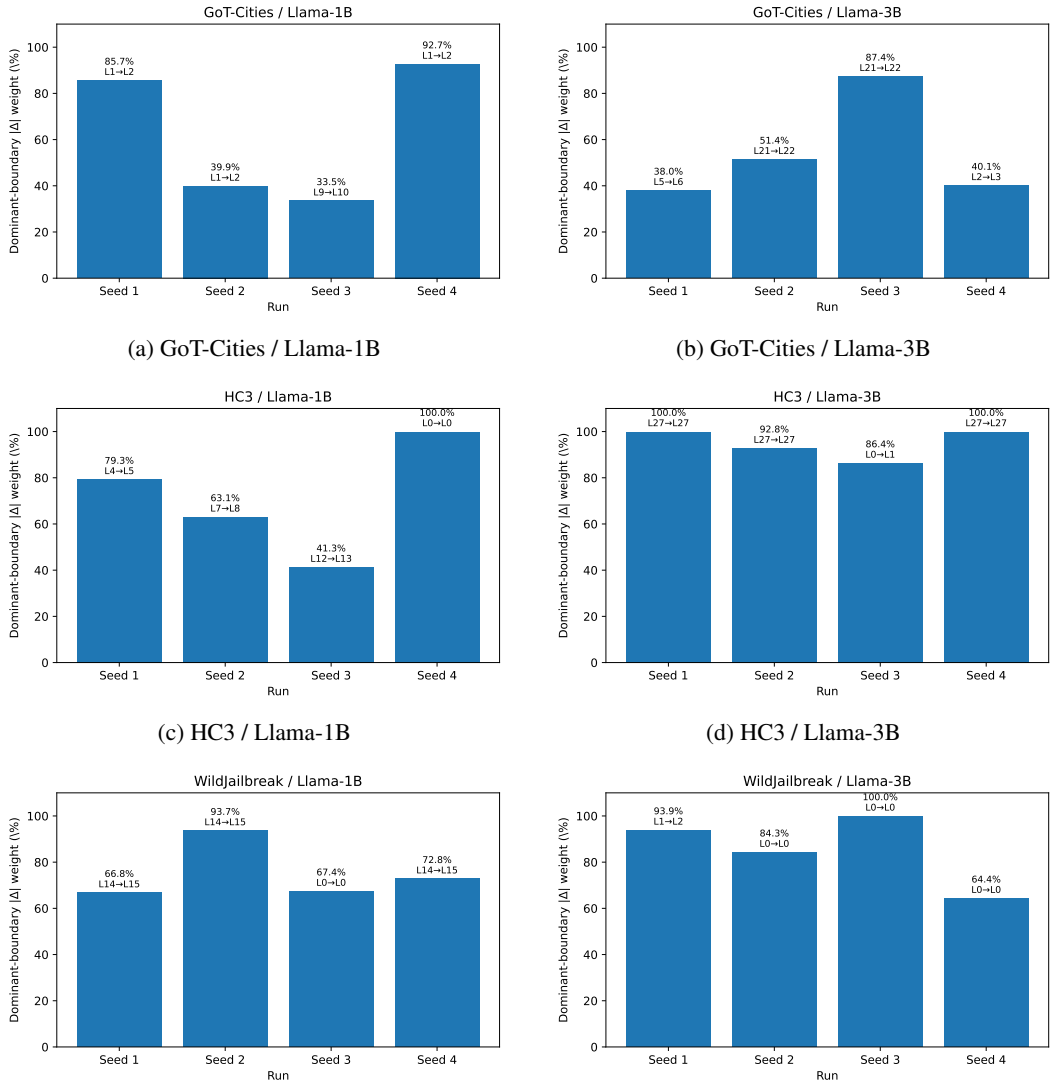


(a) Class-discriminative motifs example.



(b) Boolean rules example.

Figure 5: Representative class-discriminative motifs and Boolean rule from HC3 using GAT with Llama-3B. The motifs are mined from training explanation graphs, and the selected concepts are combined into a compact Boolean rule that reproduces the GNN prediction.



(a) GoT-Cities / Llama-1B

(b) GoT-Cities / Llama-3B

(c) HC3 / Llama-1B

(d) HC3 / Llama-3B

(e) WildJailbreak / Llama-1B

(f) WildJailbreak / Llama-3B

Figure 6: Dominant-boundary concentration across four GNN seeds for each dataset/backbone setting. Each bar shows the percentage of total discriminative motif weight concentrated on the single most dominant layer pair among the top-15 mined motifs ranked by $|\Delta|$. The boundary label above each bar indicates the dominant layer transition for that seed.

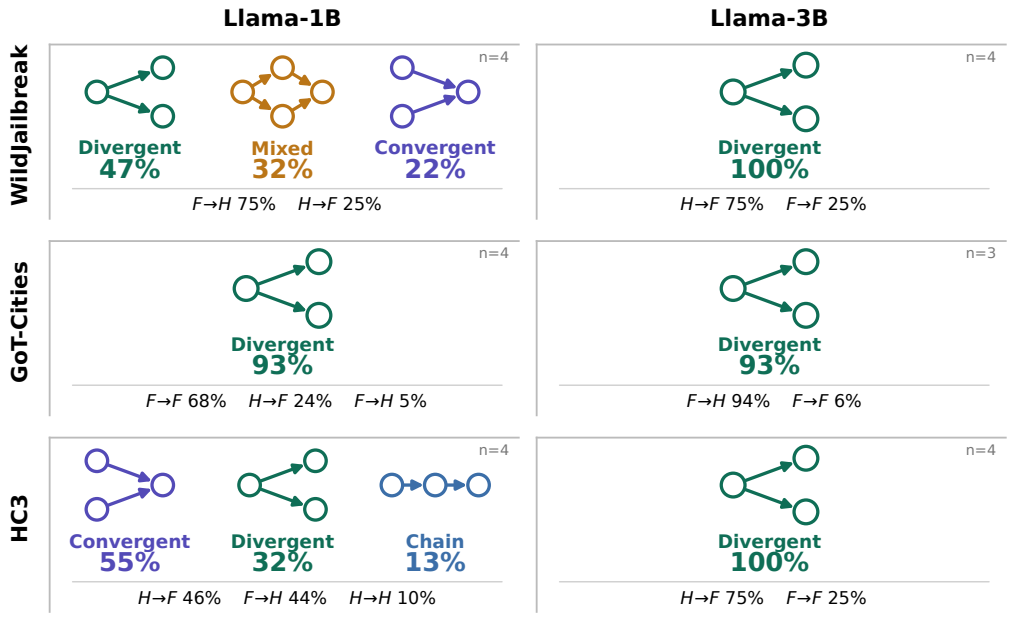


Figure 7: Motif structure across datasets and model scales. The edge-type summaries below each configuration report the relative prevalence of FFN–FFN ($F \to F$), FFN–head ($F \to H$), head–FFN ($H \to F$), and head–head ($H \to H$) interactions. Branching, particularly divergent, structures dominate most settings, whereas HC3 with Llama-1B shows a stronger convergent pattern. Here, n denotes the number of runs. For one GoT-Cities/Llama-3B seed, gSpan found only single-edge patterns, so no multi-edge motifs were available for structural analysis (n=3).