Interpreting Graph Transformers for Long-Range Interactions

Ryan Hung ryhung@ucsd.edu
James Thai jqthai@ucsd.edu
Yusu Wang yusuwang@ucsd.edu
Gal Mishne gmishne@ucsd.edu

Abstract

In recent years, graph neural networks have gained widespread traction in being able to robustly model graph topology alongside input features. Neural network explainability, which focuses on unveiling the black-box behavior of deep neural networks, run parallel to this track of geometric learning. Understanding the limitations, fairness, ethicality, and adherence to the legal and regulatory policies of a model is crucial, especially in a deployment environment impacting end users and vital stakeholders. Existing graph neural network explainability methods, such as GNNExplainer (Ying et al. 2019), focus on message-passing variants and evaluate their computation graphs. However, with the rise of graph transformers, which leverage global attention to capture long-range interactions, current explainability techniques remain limited. In this paper, we leverage transformer attention mechanisms and positional encodings, deriving two explainability frameworks dedicated to graph transformers: a heuristic-based **AttentionExplainer** and an integrated gradient-based **IGExplainer** (Sundararajan, Taly and Yan 2017). In a graph transformer setting, we demonstrate that both our methods generate subgraph explanations that are quantifiably more sufficient and necessary than those of suboptimal explainer methods. By extension, our explanations bypass the over-smoothing and over-squashing limitations of message-passing neural networks (Alon and Yahav 2020; Rusch, Bronstein and Mishra 2023) that could diminish explanation fidelity.

Website: https://ryanhungry.github.io/Interpreting-Graph-Transformers-for-Long-Range-Interactions/
Code: https://github.com/RyanHUNGry/Interpreting-Graph-Transformers-for-Long-Range-Interactions

1 Introduction

Neural network explainability seeks to capture the decision-making process of a neural network model, achieving better model transparency for an otherwise black-box system. By improving model transparency, explainability techniques help build trust in AI systems, enabling stakeholders to assess model behavior, diagnose potential biases, and ensure alignment with ethical and regulatory standards. This is particularly important in high-stakes applications such as healthcare, finance, and criminal justice, where opaque decision-making can lead to unintended consequences. By fostering better understanding of such processes, neural network explainability supports accountability, facilitates model debugging, and drives crucial discussions related to ethicality, fairness, and societal impact.

Graph neural network (GNN) explainability extends to graph-structured data, which exhibit a non-Euclidean relational structure. Existing GNN explainability methods primarily analyze the computation graph induced by message-passing, where node representations are iteratively updated by aggregating information from neighbors. However, message-passing neural networks are prone to over-squashing and over-smoothing, which hinder performance in tasks requiring long-range interactions – a structural property in which distant node heavily influences a node's true label.

Graph transformers address this limitation by incorporating global self-attention, enabling more expressive embeddings that capture long-range dependencies. Despite this advancement, limited research explores the potential of global self-attention for input-level explanation. Inspired by natural language processing – the original domain of transformers – we leverage the inherent contextual awareness of self-attention and positional encodings to interpret model behavior. By examining how a node attends to others during the representation learning stages, we gain insight into the decision-making process of graph transformers, offering a novel perspective on GNN explainability.

1.1 Contribution

We propose **AttentionExplainer**, a heuristical explainability framework for graph transformers built on global self-attention. This framework is applicable to any graph transformer architecture variant that computes global self-attention using optimized query, key, and value matrices. We offer a set of tunable hyperparameters similar to other state-of-the-art edge mask explainers. Using integrated gradients, we also derive **IGExplainer**, an attribution-based explainability method built upon positional encodings.

We compare performances for **AttentionExplainer**, **IGExplainer**, GNNExplainer (Ying et al. 2019), and a random explanation generator against a synthetically generated network with attached motifs and a long-range interaction dataset. By benchmarking against these two distinct architectures, we demonstrate that our **AttentionExplainer** and **IGExplainer**, when applied to graph transformers, outperform GNNExplainer on message-passing neural networks. Our work bridges the explainability gap between message-passing and graph transformers, highlighting the latter's superior decision-making capabilities in long-range interaction settings.

1.2 Related Work

Although there has been work in graph neural network explainability, focus has been on the message-passing neural network family of graph model architectures. In addition, these explainability methods are post-hoc in that they consider models as black-boxes and probe them for necessary information. Post-hoc interpretability methods such as GNNExplainer have shown robust results when evaluated against synthetic datasets, and are amenable to any graph neural network based upon the message-passing framework. By maximizing mutual information between an arbitrary model's prediction $\hat{Y}$ and a selected subgraph and feature explanation G_s and X_s, respectively, GNNExplainer captures influential neural message-passing pathways and feature dimensions during the information propagation steps of learning a new node embedding (Ying et al. 2019). PGExplainer builds upon GNNExplainer's mutual information optimization technique, capturing collective, global explainability by interpreting multiple predictions at once. Under the hood, the method learns a probabilistic graph generative model using various random network models such as Gilbert and Erdos–Rényi (Luo et al. 2020). GAT, though not an explainability method, localizes global self-attention to only a node's neighborhood, which in itself is a variant of message-passing. In doing so, the implicitly learned attention matrices can be visualized and used as an explanation if averaged across layers (Veličković et al. 2018). These architectures and explainability methods work at the edge-level, returning a subgraph G_s containing important edges within the graph vital to an individual node's prediction. NLP transformer explainability is quite analogue to graph transformer explainability methods. For instance, AllenNLP Interpret uses gradient-based saliency maps to quantify how important each input token is to an instance-level prediction (Wallace et al. 2019).

2 Methods

2.1 Graph Transformers

Our experiments focus on general, scalable, and powerful graph transformer architecture – GPS – a graph transformer variant using a blend of message-passing and global self-attention. In doing so, GPS prevents over-smoothing and over-squashing, as the interwoven attention layers allow information to spread across the full graph, resolving expressivity bottlenecks (Alon and Yahav 2020; Rusch, Bronstein and Mishra 2023). In addition, GPS provides a positional and structural encoding framework to capture graph topology during the self-attention process. Our experiments focus on Laplacian eigenvector and random walk matrix positional encoding, providing a node with context about its local cluster position and global graph position (Rampášek et al. 2023).

To start, the GPS transformer architecture combines node embeddings from a message-passing layer with a global self-attention layer via a multi-layer perceptron to inject information from the entire graph. Here, $A \in \mathbb{R}^{N\times N}$ represents the adjacency matrix with N nodes. $X^{\ell+1} \in \mathbb{R}^{N\times d_{(\ell+1)}}$ is the $(\ell+1)^{th}$ representational layer with $d_{(\ell+1)}$-dimensional features. Any positional or structural encoding is included as additional dimensions. We omit edge features in the formulation, as our benchmark and synthetic datasets do not possess them.

$$\mathbf{X}^{\ell+1} = \mathrm{GPS}^{\ell}(\mathbf{X}^{\ell}, \mathbf{A}), \tag{1}$$

$$\text{computed as } \mathbf{X}_M^{\ell+1} = \mathrm{MPNN}^{\ell}(\mathbf{X}^{\ell}, \mathbf{A}), \tag{2}$$

$$\mathbf{X}_T^{\ell+1} = \mathrm{GlobalAttn}^{\ell}(\mathbf{X}^{\ell}), \tag{3}$$

$$\mathbf{X}^{\ell+1} = \mathrm{MLP}^{\ell}(\mathbf{X}_M^{\ell+1} + \mathbf{X}_T^{\ell+1}). \tag{4}$$

During the global self-attention layer, the attention mechanism uses query, key, and value weight matrices $W^Q \in \mathbb{R}^{d\times d^k}$, $W^K \in \mathbb{R}^{d\times d^k}$, $W^V \in \mathbb{R}^{d\times d^k}$, respectively. These matrices are used to compute each node's query, key, and value where $Q \in \mathbb{R}^{n\times d^k}$, $K \in \mathbb{R}^{n\times d^k}$, $V \in \mathbb{R}^{n\times d^k}$. In a stacked graph transformer architecture, each GPS layer holds a learned attention weight matrix $A_w \in \mathbb{R}^{N\times N}$, where each $A_{w_{ij}}$ entry holds an attention weight reflecting the importance of the j-th node in contributing to the representation of the i-th node (Vaswani et al. 2017).

$$\mathrm{Attention}(Q, K, V) = A_w V \tag{1}$$

$$A_w = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2}$$

2.2 AttentionExplainer

For **AttentionExpainer**, we expose each trained attention matrix A_w^{ℓ} where ℓ denotes each GPS layer. In the case of multi-headed self-attention with h heads, the individual heads are averaged per layer ℓ: $A_w^{\ell} = \frac{1}{h}\sum_{i=1}^{h} A_{w_h}^{\ell}$. **AttentionExplainer** takes each trained matrix, applies Min-Max normalization, and averages them, acquiring a final attention score representation in which explanation subgraphs can be generated from (**Algorithm 1**). We propose two explanation generation algorithms of differing computational complexities: greedily selecting the top K attention scores from the set of immediate neighbors (**Algorithm 2**), and computing shortest path to the nodes with the top K attention scores of a current node to explain (**Algorithm 3**). Both methods use a greedy heuristic, as higher attention corresponds to stronger influence. **Algorithm 2** is less computationally expensive as edge masks can be directly computed from the attention matrices without any further processing. However, this lends less explanation expressivity, as the method only perceives immediate neighbors rather than the potential larger neighborhood representing actual decision-making. **Algorithm 3** requires running a graph traversal algorithm like breadth-first search to reach a target node. This edge mask generation algorithm is also limited by disconnected graphs, as two nodes in separate components cannot be reached via path. However, it offers better explanation expressivity by capturing potentially distant nodes that heavily influence the prediction of a source node.

```
Algorithm 1: AttentionExplainer score extraction
  1. Acquire trained attention matrices $\{A_w^1, ..., A_w^\ell\}$
  2. Compute the final attention score representation
       $\alpha_w^\ell = \frac{A_w^\ell - \min(A_w^\ell)}{\max(A_w^\ell) - \min(A_w^\ell)}$   (1)
       $A_{final} = \frac{1}{\ell} \sum_{k=1}^{\ell} \alpha_w^k$   (2)
  3. Choose explanation subgraph computation Algorithm 2 or Algorithm 3
```

```
Algorithm 2: Greedy Neighbor Subgraph Explanation
Require: $x_i$, the node to explain
Require: $A_{final}$, the attention matrix
Require: $A$, the adjacency matrix
Require: $k$, the number of top attention neighbors
Ensure: $G_s$, the extracted subgraph
  Compute the top-$k$ immediate neighbors for node $x_i$ using existing edge mask:
    $\text{TopK}_{immediate_i} = \text{mask}[\arg\max_{j_1,...,j_k} \left(A_{final_{ij_1}}, \ldots, A_{final_{ij_k}}\right)]$
  Initialize subgraph $G_s \leftarrow \emptyset$
  $G_s = \text{compute}_{G_s}(\text{TopK}_{immediate_i}, A)$
```

2.3 IGExplainer

We base **IGExplainer** (**Algorithm** 4) on integrated gradients, which computes edge attribution to a node-level prediction. Attribution can be viewed as a measure of importance or influence to a prediction. Integrated gradients operate upon the axiom that the gradients of a model's output with respect to its input are a natural analog to the model's coefficients. As such, analyzing this gradient is a natural starting point for generating attributions for explainability (Sundararajan, Taly and Yan 2017). For instance, in a basic binary classification task with a prediction function $F : \mathbb{R}^n \rightarrow \{0, 1\}$ that represents a deep network, we assign attribution scores for each feature $\mathbf{I} \in R^n$ where $\mathbf{I} = \{\text{IntegratedGrads}_1(x), ..., \text{IntegratedGrads}_n(x)\}$. Specifically, let $x \in \mathbb{R}^n$ be the input at hand, and $x' \in \mathbb{R}^n$ be the baseline input. For image data, the baseline could be the black image, while for text models it could be the zero

Algorithm 3: Shortest Path Subgraph Explanation

```
Require: $x_i$, the node to explain
Require: $A_{final}$, the attention matrix
Require: $A$, the adjacency matrix
Require: $k$, the number of top attention neighbors
Ensure: $G_s$, the extracted subgraph
  Compute the top-$k$ attention scores for node $x_i$:
      $\text{TopK}_i = \arg\max_{j_1,\dots,j_k} \left(A_{final_{ij_1}}, \dots, A_{final_{ij_k}}\right)$
  Initialize subgraph $G_s \leftarrow \emptyset$
  for each $x_j \in \text{TopK}_i$, do:
    Find shortest path $P_{ij}$ from $x_i$ to $x_j$ using $A$
    Merge path $P_{ij}$ into subgraph $G_s$
```

embedding vector. Starting from the baseline input, we gradually interpolate towards the original input for each feature using an interpolation factor α. At each step, we compute the gradient of the model output with respect to each input feature, $\frac{\partial F}{\partial x_i}$, and accumulate the effect of these gradients across the interpolation path through integration.

$$\text{IntegratedGrads}_i(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} \, d\alpha$$

We extend the work of AllenNLP Interpret (Wallace et al. 2019), which applies integrated gradients in the context of explaining predictive NLP tasks through attribution scores by interpolating zero embedding tokens. Since graph transformers are quite analogous to NLP transformers, we modify the formulation to work with graph topology. Specifically, we generate a baseline zero edge mask $\mathbf{v}' \in \mathbb{R}^E = \{0, ..., 0\}$ where E is the number of edges in the graph G and $\mathbf{v}_i'$ represents the i^{th} edge corresponding to PyTorch Geometric's COO format graph connectivity tensor. The input edge mask, $\mathbf{v} \in \mathbb{R}^E = \{1, ..., 1\}$, consists of all edges in the graph. However, for a graph transformer model to work with integrated gradients, we retain graph topology, which is lost in the edge mask interpolation steps, by training the model with positional encodings (Dwivedi et al. 2021). As a reuslt, we retrieve the attribution of each edge to the prediction made at the node-level. These attributions are directly used as weighted edge masks, but can be tuned through explainer hyperparameters.

2.4 Datasets

We benchmark **AttentionExplainer** and **IGExplainer** against BAShapes, a synthetically generated graph, and PascalVOC-SP, a computer vision graph suitable for global self-attention architectures due to the presence of long-range interactions. Both graphs are node-level classification tasks.

Algorithm 4: IGExplainer

```
Require: $v = \{1, ..., 1\}$, the input edge mask
Require: $v' = \{0, ..., 0\}$, the baseline edge mask
Require: $X$, the feature matrix
Require: $A$, the adjacency matrix
Ensure: $G_s$, the extracted subgraph
  Compute positional encodings (random walk example shown)
```

$$p_i^{\text{RWPE}} = \left[\text{RW}_{ii}, \text{RW}_{ii}^2, \ldots, \text{RW}_{ii}^k\right] \in \mathbb{R}^k \text{ where } RW = AD^{-1} \quad (1)$$

$$PE = \begin{bmatrix} p_1^{\text{RWPE}} \\ \vdots \\ p_n^{\text{RWPE}} \end{bmatrix} \in \mathbb{R}^{n \times k} \quad (2)$$

```
  Concatenate feature and positional encodings $X' = [X \ PE]$
  Train GPS model $GPS(X', A)$
  for each edge index $i \in \mathbf{v}$, do:
```

$$\text{IntegratedGrads}_i(v) ::= (v_i - v_i') \times \int_{\alpha=0}^{1} \frac{\partial \text{GPS}(v' + \alpha \times (v - v'))}{\partial v_i} \, d\alpha$$

```
  Assemble attribution vector $\mathbf{I} = \{\text{IntegratedGrads}_1(\mathbf{v}), ..., \text{IntegratedGrads}_n(\mathbf{v})\}$
  Initialize subgraph $G_s \leftarrow \emptyset$
  $G_s = \text{compute}_{G_s}(I, \ A)$
```

For BAShapes, we generate a graph against a Barabási–Albert network model with 25 central nodes, 5 edges per node, and attach 10 house motifs. We fix the feature matrix $X \in \mathbb{R}^{N \times 1}$ with a constant, arbitrary 1-dimensional feature, and then attach 2-dimensional Laplacian eigenvector positional encoding. The classes $Y = \{0, 1, 2, 3, 4\}$ represent the 4 roles in the house motifs, and lack thereof. As such, we obtain an explanation truth edge mask by looking at the 1-hop neighborhood of a specific node we hope to explain (Ying et al. 2019).

We use PascalVOC-SP, a node-classification graph dataset originating from semantic segmentation in computer vision. Here, nodes represent superpixels extracted from a source image, requiring more robust models due to the presence of long-range interactions (Dwivedi et al. 2023). There are multiple graphs in this dataset, so we choose an arbitrary one for node-level, binary classification. During data loading, we map node classes to ensure consecutive ordering for PyTorch Geometric's binary cross-entropy loss compatibility, and also create train, test, and validation masks. We then attach 5-dimensional random walk matrix positional encoding. Because structural semantics vary across graphs, there is no ground-truth for PascalVOC-SP.

Table 1: Comparison of BAShapes and PascalVOC-SP datasets.

	BAShapes	**PascalVOC-SP**
Nodes	75	499
Edges	~150	1413
Classes	5	2

2.5 Metrics

2.5.1 Explanation Accuracy, Recall, Precision, and F1-score

We use various metrics to quantify explanation robustness: explanation accuracy, recall, and precision. These metrics apply only to BAShapes, as generated explanations on this dataset can be compared against an explanation truth edge mask.

Explanation accuracy measures how accurate our generated node-level explanations are. The explanation accuracy against the BAShapes dataset determines an explainer's ability to capture the corresponding house motifs roles. PascalVOC-SP is not a synthetically generated graph, we cannot use explanation accuracy since we are unable to determine an infallible, oracle-like truth about the relationships between nodes.

We also use the recall metric, which measures the proportion of relevant nodes captured by the generated explanation. If an explanation captures all influential nodes for a given prediction, the recall score will be 1. The recall formula is given below, where TP denotes True Positives and FN denotes False Negatives.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

We also use the precision metric, which measures the proportion of selected nodes that are truly influential in the prediction. If all nodes included in an explanation are relevant to the outcome, the precision score will be 1. The precision formula is given below, where TP denotes True Positives and FP denotes False Positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

We also use the F1-score, which provides a balanced measure of precision and recall. It is the harmonic mean of precision and recall, ensuring that both metrics are considered when evaluating the quality of an explanation. The F1-score reaches its best value at 1 when both precision and recall are maximized, and its worst value at 0 when either precision or recall is zero. The formula for the F1-score is given.

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

A higher F1-score indicates a better balance between precision and recall, making it especially useful when there is an uneven distribution of positive and negative instances.

2.5.2 Fidelity and Characterization Scores

Fidelity is another metric that we use to quantify the robustness of our explanations. Since our explanations are subgraph structures, fidelity measures how efficiently vital (Amara et al. 2024) a subgraph is in determining a certain prediction. Fidelity scores range from 0-1.

There are two types of fidelity scores – Fid+ and Fid-. Fid+ quantifies a ”necessary” explanation. An explanation is deemed necessary if the model's prediction changes when removed from the initial graph. A high score is 1 and a low score is 0. The formula for Fid+ is as follows.

$$fid+ = 1 - \frac{1}{N} \sum i = 1^N \mathbb{1}\left(\hat{y}_i^{G \setminus S} = \hat{y} i\right)$$

Fid- quantifies a ”sufficient” explanation. An explanation is sufficient if the explanation's prediction (the subgraph) leads to the same outcome as the initial outcome of a prediction. A high score is 0 and a low score is 1. The formula for Fid- is as follows.

$$fid- = -\frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left(\hat{y}_i^{G^S} = \hat{y}_i\right)$$

We use both these measures to quantify the explanation robustness. However, a better ensemble statistic is characterization score – the weighted harmonic mean of Fid+ and Fid-

. This score allows us to tune the weights based on the fidelity measure we value the most. The formula for characterization score is as follows.

$$charact = \frac{w+ + w-}{\frac{w+}{fid+} + \frac{w-}{1-fid-}} = \frac{(w+ + w-) \times fid + \times (1 - fid-)}{w+ \cdot (1 - fid-) + w- \cdot fid+}$$

3 Experiments

3.1 Setup

We benchmark our **AttentionExplainer** and **IGExplainer** against GNNExplainer and a dummy explainer on BAShapes, a synthetically-generated dataset, and PascalVOC-SP, a computer vision dataset.

3.1.1 Explainer Hyperparameters

All explainers within our benchmark support the following thresholding hyperparameters that modify the output explanation subgraph:

1. **topk**: Limits the number of edges in the explanation to k.
2. **hard**: Removes edges within a mask that have values below a threshold v, while setting others to 1.
3. **topk_hard**: Similar to **topk**, but all retained elements are set to 1.

3.1.2 GNNExplainer

BAShapes In BAShapes, GNNExplainer is set up on both GPS and GCN. The model parameters are as follows:

GPS

1. **pe_channels**: GNNExplainer on GPS has two positional encoding channels
2. **num_layers**: 4 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 1 attention head
5. **observe_attention**: True

GNNExplainer utilizes the **topk** thresholding hyperparameter, where k is 10, for GPS.

GNNExplainer is trained on 400 epochs.

GCN

1. **num_layers**: 4 total layers
2. **hidden_channels**: 20 hidden channels

GNNExplainer utilizes the **topk** thresholding hyperparameter, where *k* is 10, for GCN.

GNNExplainer is trained on 4000 epochs.

PascalVOC-SP The model parameters for the dataset PascalVOC-SP are as follows:

GPS

1. **pe_channels**: 2 positional encoding channels
2. **num_layers**: 4 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 4 attention heads
5. **observe_attention**: True

topk is set to 10. GNNExplainer is trained on 400 epochs for PascalVOC-SP for GPS

GCN

1. **num_layers**: 20 total layers
2. **hidden_channels**: 20 hidden channels

The hyperparameter **topk** is set to 10. GNNExplainer is trained on 2000 epochs for PascalVOC-SP for GPS.

3.1.3 DummyExplainer

DummyExplainer generates a random subgraph explanation for every prediction, and the metrics it produces are used for comparison purposes.

The model parameters for GPS and GCN are as follows:

BAShapes

GPS

1. **pe_channels**: 2 positional encoding channels
2. **num_layers**: 4 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 1 attention head
5. **observe_attention**: True

GNNExplainer utilizes the **topk** thresholding hyperparameter, where k is 10, for GPS. GNNExplainer is trained on 4000 epochs.

GCN

1. **num_layers**: 3 total layers
2. **hidden_channels**: 20 hidden channels

For GCN, **topk** is also utilized and k is set to 10. It is also trained on 4000 epochs.

3.1.4 AttentionExplainer

The model parameters for **AttentionExplainer** are as follows (only GPS):

BAShapes :

1. **pe_channels**: 2 positional encoding channels
2. **num_layers**: 4 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 1 attention head
5. **observe_attention**: True

AttentionExplainer is trained on 400 epochs.

PascalVOC-SP

1. **pe_channels**: 5 positional encoding channels
2. **num_layers**: 2 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 4 attention heads
5. **observe_attention**: True

AttentionExplainer is trained on 400 epochs for PascalVOC-SP.

3.1.5 IGExplainer

For **IGExplainer**, the model setup is:

BAShapes

1. **pe_channels**: 2 positional encoding channels
2. **num_layers**: 4 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 1 attention head
5. **observe_attention**: True

6. **integrated_gradients**: True

PascalVOC-SP For **PascalVOC-SP**, the set up is:

1. **pe_channels**: 5 positional encoding channels
2. **num_layers**: 2 total layers
3. **hidden_channels**: 4 hidden channels
4. **num_attention_heads**: 4 attention heads
5. **observe_attention**: True
6. **integrated_gradients**: True

The threshold configuration is **topk**, where k is 5.

IGExplainer is trained on 400 epochs.

3.2 Results

For each explainer, model, and dataset combination, we perform hyperparameter tuning to achieve the highest characterization score.

Table 2: Explainer Accuracy for BAShapes

Method	GNN Model	Dataset	Explanation Accuracy ↑	Recall ↑	Precision ↑	F1 ↑
GNNExplainer	GPS	BAShapes	94.0 ± 1.1	10.0 ± 1.3	10.0 ± 1.5	16.0 ± 1.3
	GCN		95.3 ± 0.8	**100.0** ± 0.7	36.3 ± 1.7	**57.7** ± 1.2
DummyExplainer	GPS	BAShapes	94.2 ± 1.2	0.0 ± 0	1.0 ± 0	3.0 ± 1.2
	GCN		91.3 ± 1.0	0.0 ± 0	0.0 ± 0	0.0 ± 0
AttentionExplainer*	GPS	BAShapes	93.35 ± 1.3	25.0 ± 2.5	11.76 ± 1.5	15.99 ± 1.4
IGExplainer*	GPS	BAShapes	**96.35** ± 1.9	94.3 ± 2.1	**36.76** ± 1.1	52.9 ± 1.6

Table 3: Explainer Fidelity for BAShapes (Maximized Against Characterization Score)

Method	GNN Model	Dataset	Fid+ ↑	Fid- ↓	Characterization Score ↑
GNNExplainer	GPS	BAShapes	66.6 ± 3.1	66.6 ± 0.9	44.4 ± 2.8
	GCN		25.3 ± 0.3	61.4 ± 0.9	30.6 ± 0.5
DummyExplainer	GPS	BAShapes	46.6 ± 1.5	73.5 ± 0.3	34.0 ± 0.9
	GCN		45.3 ± 1.1	**48.0** ± 2.5	48.4 ± 1.5
AttentionExplainer*	GPS	BAShapes	61.33 ± 1.4	57.33 ± 0.5	50.32 ± 1.9
IGExplainer*	GPS	BAShapes	**91.73** ± 3.1	51.31 ± 2.3	**65.06** ± 2.7

3.3 Discussion

In **Table 2**, explanation accuracy is high among all explainer algorithms. We attribute this to the limited size of the explanation truth edge mask. By simply limiting the number of

Table 4: Explainer Fidelity for PascalVOC-SP (Maximized Against Characterization Score)

Method	GNN Model	Dataset	Fid+ ↑	Fid- ↓	Characterization Score ↑
GNNExplainer	GPS	PascalVOC-SP	12.0 ± 1.5	56.0 ± 1.8	19.3 ± 1.2
	GCN		28.2 ± 0.9	68.2 ± 1.3	15.3 ± 1.7
AttentionExplainer*	GPS	PascalVOC-SP	42.3 ± 0.7	**2.4** ± 1.3	58.7 ± 1.1
IGExplainer*	GPS	PascalVOC-SP	**48.0** ± 1.2	6.0 ± 1.9	**65.3** ± 1.4

edges an explainer returns, high accuracy is achieved even if the explanation edge mask does not contain the explanation truth's edge mask. As such, a better indicator of an explainer's ability to represent truth is recall, precision, and F1-score. This is because recall and precision capture how well the explainer retrieves the relevant edges while avoiding irrelevant ones. Recall measures the proportion of relevant edges correctly identified, while precision evaluates the proportion of retrieved edges that are actually relevant. F1-score provides a balanced measure, considering both recall and precision, making it a more reliable metric for assessing an explainer's effectiveness. We found that GNNExplainer consistently achieves the highest recall, precision, and F1-score. This can be attributed to its ability to maximize mutual information while directly evaluating the computation graph formed during message-passing layers, such as those in GCN. However, when applied to GPS, GNNExplainer performs poorly on these metrics, as the self-attention mechanism in GPS does not generate a computation graph. Notably, **IGExplainer** with GPS closely follows GNNExplainer in performance across these metrics, making it a strong alternative for explaining graph transformer models.

In **Table 3**, we evaluate explainer algorithms using fidelity measures on BAShapes, which we argue provide a more robust assessment of explanation quality (Amara et al. 2024). While explanation accuracy and its derived metrics can indicate whether an explainer captures the explanation truth, their perceived effectiveness is highly dependent on the underlying model architecture. This dependency creates an undesirable entanglement between model decision-making and explanation generation, where an explanation may not align with the true underlying rationale but still faithfully represent how the model arrived at its decision. We see this phenomenon occur with **AttentionExplainer** on GPS, which achieves a low F1-score but possesses a high characterization score on BAShapes. This means **AttentionExplainer** effectively produces an explanation that is both necessary and sufficient when describing GPS' decision-making, even if these decisions do not conform with the explanation truth. Notably, **IGExplainer** receives high scores for both fidelity and accuracy metric sets. We attribute this to the gradient accumulation phase, which captures the change in model output with respect to each feature across an interpolated space. We argue this gradient-based method offers better explanation robustness than the shortest path or greedy heuristics used in **AttentionExplainer**.

Similarly in **Table 4**, which is evaluated on PascalVOC-SP, both **IGExplainer** and **AttentionExplainer** outperform GNNExplainer in terms of fidelity measures. This is notable, as there is no explanation truth available for PascalVOC-SP, given that the data is not synthetically generated. Additionally, PascalVOC-SP includes long-range interactions, which are more effectively captured by graph transformers than by message-passing architectures (Alon and Yahav 2020; Rusch, Bronstein and Mishra 2023). This unique structural property evidences the superior performance of **IGExplainer** and **AttentionExplainer**, as GNNExplainer struggles to capture long-range dependencies using GCN or fails to optimally explain GPS. The latter is hindered by the absence of computation subgraphs, which are essential for GNNExplainer's probabilistic optimization of mutual information.

In essence, our results show that our **AttentionExplainer** and **IGExplainer** effectively and efficiently capture decision-making for graph transformers across simple pattern recognition tasks and long-range interaction settings.

4 Conclusion

The importance of neural network explainability has grown alongside the increasing complexity of both data and model architectures. Specifically, graph neural network explainability aims to shed light on the decision-making processes of models that incorporate both features and relational structures during training. However, there is a notable gap in the literature regarding the explainability of graph transformers. To address this, we propose two methods: **AttentionExplainer** and **IGExplainer**. The former utilizes trained attention weights to greedily generate subgraph explanations, while the latter applies integrated gradients to compute edge attribution for each edge in the graph. Our approach is directly applicable to any graph transformer model that employs self-attention, offering a more appropriate explanation framework for this architecture compared to existing explainer techniques that are primarily designed for message-passing neural networks. Our explainer algorithms provide efficient and vital interpretations of graph transformer decision-making, providing critical model transparency, fairness, ethicality, and legal adherence.

References

Alon, Uri, and Eran Yahav. 2020. "On the Bottleneck of Graph Neural Networks and its Practical Implications." *CoRR* abs/2006.05205. [Link]

Amara, Kenza, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. 2024. "GraphFramEx: Towards Systematic Evaluation of Explainability Methods for Graph Neural Networks." [Link]

Dwivedi, Vijay Prakash, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. "Graph Neural Networks with Learnable Structural and Positional Representations." *CoRR* abs/2110.07875. [Link]

Dwivedi, Vijay Prakash, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2023. "Long Range Graph Benchmark." [Link]

Luo, Dongsheng, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. "Parameterized Explainer for Graph Neural Network." *CoRR* abs/2011.04573. [Link]

Rampášek, Ladislav, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaiani. 2023. "Recipe for a General, Powerful, Scalable Graph Transformer." [Link]

Rusch, T. Konstantin, Michael M. Bronstein, and Siddhartha Mishra. 2023. "A Survey on Oversmoothing in Graph Neural Networks." [Link]

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. 2017. "Axiomatic Attribution for Deep Networks." *CoRR* abs/1703.01365. [Link]

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." *CoRR* abs/1706.03762. [Link]

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. "Graph Attention Networks." [Link]

Wallace, Eric, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. "AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models." 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): System Demonstrations*. Hong Kong, China Association for Computational Linguistics. [Link]

Ying, Rex, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. "GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks." *CoRR* abs/1903.03894. [Link]

Appendices

A.1 Proposal

Our original proposal for this project is here.

A.2 Acknowledgments

We thank Professor Yusu Wang and Professor Gal Mishne for their persistent guidance and mentorship.