XInsight: Revealing Model Insights for GNNs with Flow-Based Explanations

Eli Laird(✉), Ayesh Madushanka, Elfi Kraka, and Corey Clark

Southern Methodist University, Dallas, TX, USA
{ejlaird,amahamadakalapuwage,ekraka,coreyc}@smu.edu

Abstract. Progress in graph neural networks has grown rapidly in recent years, with many new developments in drug discovery, medical diagnosis, and recommender systems. While this progress is significant, many networks are 'black boxes' with little understanding of the 'what' exactly the network is learning. Many high-stakes applications, such as drug discovery, require human-intelligible explanations from the models so that users can recognize errors and discover new knowledge. Therefore, the development of explainable AI algorithms is essential for us to reap the benefits of AI.

We propose an explainability algorithm for GNNs called eXplainable Insight (XInsight) that generates a distribution of model explanations using GFlowNets. Since GFlowNets generate objects with probabilities proportional to a reward, XInsight can generate a diverse set of explanations, compared to previous methods that only learn the maximum reward sample. We demonstrate XInsight by generating explanations for GNNs trained on two graph classification tasks: classifying mutagenic compounds with the MUTAG dataset and classifying acyclic graphs with a synthetic dataset that we have open-sourced. We show the utility of XInsight's explanations by analyzing the generated compounds using QSAR modeling, and we find that XInsight generates compounds that cluster by lipophilicity, a known correlate of mutagenicity. Our results show that XInsight generates a distribution of explanations that uncovers the underlying relationships demonstrated by the model. They also highlight the importance of generating a diverse set of explanations, as it enables us to discover hidden relationships in the model and provides valuable guidance for further analysis.

Keywords: Explainable AI · Graph Neural Networks · GFlowNets

1 Introduction

Graph neural networks (GNNs) have emerged as a popular and effective machine learning algorithm for modeling structured data, particularly graph data. As GNNs continue to gain popularity, there is an increasing need for explainable GNN algorithms. Explainable AI refers to machine learning algorithms that can provide understandable and interpretable results. Explainable AI algorithms

L. Longo (Ed.): xAI 2023, CCIS 1902, pp. 303–320, 2023.
https://doi.org/10.1007/978-3-031-44067-0_16

have the ability to uncover hidden relationships or patterns that deep learning models use in making their decisions. This means that researchers can use these methods to understand why a model arrived at a certain decision. In the case of GNNs, the need for explainability arises from the fact that they are often used in applications where the decision-making process needs to be transparent and easily understood by humans. For example, in the field of drug discovery, GNNs are used to predict the efficacy of a drug by analyzing its molecular structure [49]. In this case, it is crucial to understand how the GNN arrived at its prediction, as it can have significant implications for patient health and safety.

Explainable AI algorithms also uncover erroneous correlations in deep learning models. For instance, in a study by Narla et al. [25], the researchers found that their model had incorrectly learned that images with rulers were more likely to be cancerous. The use of explainable AI methods helped them uncover this error and highlighted the need for methods that can explain the underlying relationships that deep learning models rely on to make predictions.

In response to this need, we propose a novel GNN explainability algorithm, *eXplainable Insight (XInsight)*, that generates diverse model-level explanations using Generative Flow Networks (GFlowNets) [3]. XInsight represents the first application of GFlowNets to explain graph neural networks. Unlike previous model-level algorithms, that only learn the maximum reward sample, XInsight generates objects with probabilities proportional to a reward. We demonstrate the effectiveness of XInsight by applying it to GNNs trained on two graph classification tasks: classifying mutagenic compounds with the MUTAG dataset and classifying acyclic graphs with a synthetic dataset. In our experiments, we demonstrate that XInsight's explanations for the MUTAG dataset [24] can be analyzed using data mining techniques and QSAR [14] modeling to uncover hidden relationships in the model. For instance, when analyzing the compounds generated by XInsight, we found that they clustered by lipophilicity, which is a known correlate of mutagenicity. Our results demonstrate that XInsight generates a distribution of explanations that enables the discovery of hidden relationships in the model.

The key contributions of this paper are summarized below:

(i) We proposed eXplainable Insight (XInsight), an explainability algorithm for Graph Neural Networks (GNNs) that uses GFlowNets to generate a distribution of model explanations.
(ii) We applied XInsight to explain two classification tasks, one of which was a newly open-sourced synthetic dataset, and the other was a real-world molecular compound dataset.
(iii) We analyzed XInsight's generated explanations using a clustering method and chemical analysis tool, which helped us to discover important underlying patterns and relationships of the examined model.

2 Related Work

2.1 Graph Neural Networks

Graph neural networks (GNNs) have emerged as a popular deep learning technique to model structured data that can be represented as graphs. Unlike traditional neural networks that operate on structured data like images and sequences, GNNs operate on non-Euclidean data, such as social networks [9,34], chemical molecules [5,11,12,49], and 3D point clouds [10,31]. GNNs typically use a message-passing approach [12], where the feature representations of nodes, edges, and the overall graph are iteratively updated by aggregating the features of their neighbors and combining them with the learned features from the previous step. This message-passing process is repeated for a fixed number of iterations or until convergence. Expanding upon traditional message-passing GNNs, many other GNN architectures have been proposed, such as Graph Convolutional Networks (GCNs) [48] that use convolutional operations similar to Euclidean Convolutional Neural Networks, Graph Isomorphism Networks (GINs) [41] that employ multilayer perceptrons to aggregate neighboring features, and Graph Attention Networks (GATs) [36] that apply an attention mechanism to weigh contributions of neighboring nodes/edges based on their importance. With the development of GNNs, we can now model and make predictions based on structured data in a way that was not possible before.

2.2 Explaining Graph Neural Networks

Graph neural networks (GNNs) are widely used in various domains such as drug discovery [5,11,12,49], recommendation systems [39,40,43], and medical diagnosis [1,18,19]. However, as with other machine learning models, GNNs are often considered to be 'black boxes', providing little insight into how they make predictions. Therefore, explainable AI algorithms for GNNs have gained increasing attention in recent years.

There are several approaches to developing explainable GNN algorithms that can conveniently be categorized as *instance-level* and *model-level* approaches. Instance-level algorithms provide explanations for individual predictions of the GNN and include methods that utilize the gradients of the features to determine input importance, such as sensitivity analysis, Guided BP, and Grad-CAM [2,30,33], perturb inputs to observe changes in output as in GNNExplainer and PGExplainer [20,42], and learn relationships between the input and its neighbors using surrogate models [15,38]. While there are several instance-level explainability methods for GNNs, there is still a lack of effective model-level explainability methods [46].

Model-level explanations help identify how the GNN approaches the task at hand, and can reveal patterns and structures that may not be immediately evident from the graph data alone. They also help identify when a model is not performing well on the given task, or when it is exhibiting unwanted behavior. Outside of the graph-learning world, input optimization is a popular model-level

approach for image classification models, where the goal is to generate an image that maximizes the predicted class label [8,21,26–28,32]. In contrast, model-level explanations for GNNs have received relatively less attention. One of the most prominent model-level explainability methods for GNNs is XGNN (eXplainable Graph Neural Networks) [45]. XGNN leverages reinforcement learning techniques to generate graphs that provide insights into how the GNN is making predictions. XGNN generates a graph explanation that maximizes a target prediction, thereby revealing the optimized class pattern learned by the model.

To gain more insight into the model, it is often necessary to analyze a diverse distribution of examples that cover different scenarios and edge cases. Furthermore, generating a distribution of explanations opens the door to applying statistical analysis and data mining techniques, such as dimensionality reduction or t-tests, to uncover hidden relationships in the data. For instance, in this paper we use dimensionality techniques to uncover clusters within XInsight explanations. When then used these clusters to verify that the model correctly learned a known correlation within the data.

2.3 XGNN

XGNN, which stands for eXplainable Graph Neural Networks, is a novel model-level explainability framework introduced by Yuan et al. in 2020 [45]. The goal of XGNN is to generate a graph that maximizes a specific target class of a graph classification model. XGNN employs a reinforcement learning approach to iteratively build a graph using actions that add nodes or edges to the graph at each time step. During each time step, the model calculates the reward based on the probability of the target class, which encourages the algorithm to select actions that generate graphs of a particular class. This process is repeated until the model converges or until a maximum number of time steps is reached.

Like Graph Convolutional Policy Networks [44], XGNN learns a generator model using a policy gradient. The generator produces a graph that contains patterns that maximize the target class in question. In contrast to instance-level explainability methods that identify subgraphs that contribute to the model's output, XGNN focuses on the entire graph and the relationships between its nodes and edges. XGNN is currently the only model-level explanation method that has been proposed for GNNs, according to a recent survey [46].

While XGNN is a powerful model-level explainability method for GNNs, it generates a single maximum reward explanation, which limits its ability to explain the full extent of the model's behavior. XGNN is also limited in its utility to discover hidden insights related to the classification task due to the inability to perform a more detailed analysis of the explanations, such as clustering. Due to these limitations, there is no way of directly comparing XGNN to techniques that generate a distribution of explanations, such as XInsight.

2.4 GFlowNets

Generative Flow Networks (GFlowNets) are a type of generative model that generate a diverse set of objects by iteratively sampling actions proportional to

a reward function [3,4]. The objective of GFlowNets is to learn to sample from a distribution of diverse and high-reward samples instead of generating a single sample to maximize a reward function. GFlowNets can be viewed as Markov Decision Processes (MDP) represented by a directed acyclic graph (DAG), where the edges represent the actions that can be taken in the states. The flows coming into a state represent the actions that can be taken to reach that state, while the flows leaving a state represent the actions that can be taken in that state to reach the next state. The DAG is traversed iteratively by sampling flows, which generates a flow trajectory that ends when a terminal state is reached. The flow entering a terminal state is the total flow of the trajectory and is equal to the reward function assigned to that state.

Trajectory Balance Objective. In [22], Malkin et al. introduced the *Trajectory Balance Constraint*, shown in Eq. 1, which ensures that the flow of the trajectory leading to a state is equal to the flow of the trajectory leaving that state and terminating at a terminal state. Satisfying this constraint allows the GFlowNet to sample objects with probability proportional to its reward.

$$Z \prod_t P_F(s_{t+1}|s_t) = R(\tau) \prod_t P_B(s_t|s_{t+1}) \tag{1}$$

where Z is the 'total flow'. The right side of Eq. 1 represents the fraction of the total reward going through the trajectory, while the left represents the fraction of the total flow going through the trajectory. This constraint can be turned into the *Trajectory Balance Objective* [22] for training a GFlowNet, shown below:

$$L_{TB}(\tau) = \left(\log \frac{Z \prod_t P_F(s_{t+1}|s_t)}{R(\tau) \prod_t P_B(s_t|s_{t+1})} \right)^2 \tag{2}$$

Applications of GFlowNets. GFlowNets have been applied to many generative applications, including molecular sequence generation [3,16,22] and MNIST image generation [47]. And due to their ability to generate diverse samples, GFlowNets trained to generate model explanations, as in XInsight, provide the machine learning user a greater breadth of human-readable explanations of what their models are learning from the data.

3 eXplainable Insight (XInsight)

3.1 Explaining Graph Neural Networks

Graph classification networks can be difficult for humans to interpret since graph structures can be less intuitive to humans compared to visual features which humans are naturally equipped to interpret. Therefore when seeking to understand a graph classification model, a quality explainability algorithm should take

advantage of the natural pattern matching capabilities of its human users by producing concise explanations that highlight patterns that are easily interpreted by humans. To make it even easier for its users, a quality explainability algorithm should produce a distribution of explanations in order to provide the user with multiple perspectives into the model; however, most algorithms to date lack one or both of these qualities.

XInsight not only produces concise explanations that highlight important patterns but also produces a distribution of explanations that allows the user to develop a more robust understanding of what the examined model is learning from the data. XInsight trains a GFlowNet to generate a diverse set of model-explanations for a graph classification model. The explanations that XInsight generates highlight general patterns that the classification model attributes to specified target class in question.

In the context of model explanations as a whole, the explanations that XInsight produces are particularly useful for discovering relationships within the trained model. For example, they can help determine if a model incorrectly associates an artifact in the data with the target class, like rulers with skin cancer as discussed in [25]. XInsight empowers users to do this by generating a distribution of explanations, which can then be passed through traditional data mining techniques, such as clustering, to uncover what the model is learning from the data.

3.2 Generating Graphs with XInsight

XInsight employs a GFlowNet that it is trained to generate graphs with probabilities proportional to their likelihood of belonging to a target class. Specifically, the GFlowNet generates a graph by iteratively sampling actions that determine whether to add a new node or edge to the existing structure. It is important to note that the likelihood of a sample belonging to a particular class is defined by the trained model that is being explained. Therefore, the distribution of generated samples is dependent upon the trained model and not the true class distribution, which in the context of explaining a trained model is desirable since the goal is to understand the model itself.

Action Space. The action space, $\mathcal{A}$, is split into two flows: the first selecting a starting node and the second selecting the ending node. The starting node is selected from the set of nodes N in the current incomplete graph G_t. The ending node is selected from the union of the same $\mathcal{N}$, excluding the starting node and a set of building blocks $\mathcal{B}$. Together, the starting and ending nodes form the combined action $\mathcal{A}(n_s, n_e)$ sampled from the forward flow P_F. Taking this action generates a new graph G_{t+1} as shown below:

$$G_{t+1} \sim P_F(\mathcal{A}(n_s, n_e)|G_t) \tag{3}$$

$$p_{start}(n_s \in N|G_t) \tag{4}$$

$$p_{end}(n_e \in N \cup \mathcal{B}; n_e \neq n_s | G_t) \tag{5}$$

3.3 Proxy

The proxy f in classical GFlowNets is used to generate the reward for a generated object. For example, in [3] Bengio et al. used a pretrained model as their proxy to predict the binding energy of a generated molecule to a protein target. In XInsight, we use the model to be explained as the proxy since the generated objects are treated as explanations of the model.

Reward. The reward in XInsight guides the underlying GFlowNet to generate graphs that explain the proxy. In XInsight, we define the reward as the proxy's predicted probability that the generated graph belongs to the target class c, as shown in Eq. 6. To encourage the generation of objects explaining the target class, we define the reward to be zero if the generated object is classified as the opposite class. In addition, we add a scalar multiplier α to magnify the reward for the target class, where $\alpha > 0$.

$$R(G_t) = \begin{cases} \alpha * softmax(f(G_t)) & \text{if } argmax\{f(G_t)\} = Target\ Class \\ 0 & \text{if } argmax\{f(G_t)\} \neq Target\ Class \end{cases} \tag{6}$$

3.4 Training XInsight

We train XInsight using the trajectory balance objective, following [22]. We define the trajectory balance objective for a complete graph G generated over a trajectory τ actions in Eq. 7.

$$L_{TB}(G) = \left(\log \frac{Z \prod_t^\tau P_F(G_{t+1}|G_t)}{R(G_t) \prod_t^\tau P_B(G_t|G_{t+1})} \right)^2 \tag{7}$$

The training loop consists of sampling trajectories (i.e. generating graphs), calculating forward and backward flows and the reward, and updating the underlying GFlowNet parameters until convergence. We highlight the in-depth steps of XInsight's training loop in Algorithm 1.

For every epoch in the training loop, we start by initializing the GFlowNet and creating an initial graph G_0. Then we generate a graph by iteratively sampling actions from the forward policy P_F that add nodes or edges to the graph at each step G_t. Once a trajectory is complete, either by the reaching the $MAX\_ACTIONS$ limit or by sampling a *stop* action, the reward is computed for G_t using the $Proxy$. Finally, we calculate the trajectory balance loss, update the GFlowNet's parameters and repeat.

Algorithm 1. XInsight Training Loop

Input: $EPOCHS$, $Proxy(\cdot)$, $TARGET\_CLASS$, $MAX\_ACTIONS$

```
XI(·;θ) ← GFlowNet
for epoch in EPOCHS do
    actions ← 0
    G_t ← G_0                                   ▷ Initialize new graph
    τ ← ∅ + {G_t}
    repeat
        P_F, P_B ← XI(G_t; θ)                   ▷ Generate flows
        n_s, n_e ~ P_F                          ▷ Sample start & end node
        if n_s = stop then
            STOP ← True                         ▷ Stop if stop action sampled
        end if
        G_new ← T(n_s, n_e)                     ▷ Add node/edge
        P_F, P_B ← XI(G_new; θ)                 ▷ Recompute flows
        G_t = G_new
        τ ← τ + {G_t}                           ▷ Append G_t to trajectory
        actions + +
    until actions > MAX_ACTIONS or STOP
    Reward = softmax(Proxy(G_t))_TARGET_CLASS   ▷ Calculate reward
    θ ← θ − η∇Loss_TB(Reward, log_Z, τ)         ▷ Update parameters
end for
```

4 Experiment Design

4.1 Datasets

The Acyclic Graph dataset includes 2405 synthetically generated graphs labeled as either acyclic or cyclic. We generated graphs using graph generation functions from the NetworkX software package [13]. To improve the diversity of the dataset, we trained a GFlowNet with a brute-force cycle checker as a reward function to generate acyclic and cyclic graphs to add to the dataset. The code used to generate this dataset can be found in [17].

The MUTAG dataset [24], included in Pytorch Geometric, contains 188 graphs representing chemical compounds used in an Ames test on the S. Typhimurium TA98 bacteria with the goal of measuring the mutagenic effects of the compound. This dataset was used in a study to measure the correlation between the chemical structure of the compounds and their mutagenic activity [7]. The nodes and edges in the graphs in MUTAG represent 7 different atoms (Carbon, Nitrogen, Oxygen, Fluorine, Iodine, Chlorine, and Bromine) and their chemical bonds. In the graph learning community, the dataset is used as a benchmark dataset for graph classification models labeling each graph as 'Mutagenic' or 'Non-Mutagenic'.

4.2 Verifying XInsight's Generative Abilities Setup

To validate that XInsight can generate graphs belonging to a target class, we trained XInsight to generate acyclic graphs because of their simple and human-

interpretable form. For the proxy, we trained a graph convolutional neural network (GCN) to classify acyclic graphs using the Acyclic Graph dataset and node degree as the node features, achieving 99.58% accuracy. This GCN is composed of three graph convolutional layers (GCNConv) with 32, 48, 64 filters, respectively, a global mean pooling layer, and two fully connected layers with 64 and 32 hidden units. We also used dropout and the ReLU activation function. The GFlowNet was also a GCN made up of three GCNConv layers with 32, 64, and 128 filters, two fully connected layers with 128 and 512 hidden units, and a scalar parameter representing $log(Z)$ from the reward function, see Sect. 3.3. The building blocks used for action selection consisted of a single node of degree 1.

4.3 Revealing MUTAG Relationships Setup

Due to their highly qualitative nature, there is no established method for evaluating model-level explanation methods for graphs, particularly for methods that generate a distribution of explanations. Despite this barrier, we demonstrate XInsight's explanatory abilities by applying it to the task of knowledge discovery within the mutagenic compound domain. Particularly, we evaluated XInsight for its ability to uncover meaningful relationships learned by a graph neural network trained to classify mutagenic compounds and verify that these relationships exist in the ground truth data.

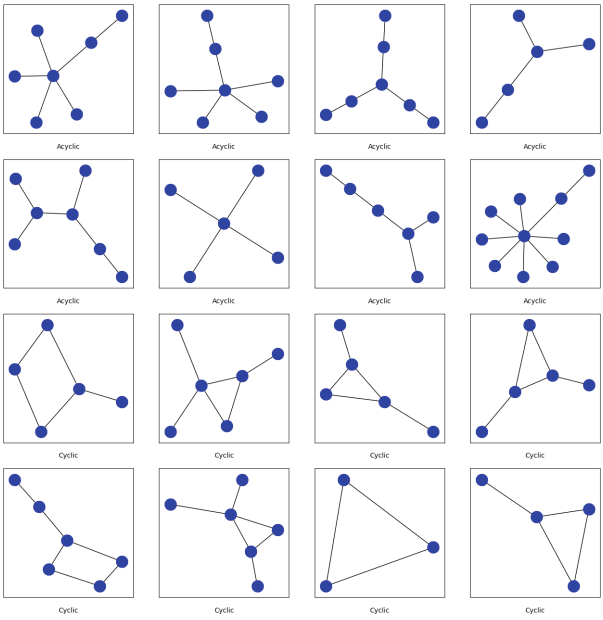

Fig. 1. Generated graphs (8 with cycles and 8 without cycles) to verify XInsight's ability to generate graphs of a specified target class.

For the proxy, we trained a graph convolutional neural network (GCN) to classify mutagenic compounds using the MUTAG dataset. Following [45], we used node features which were seven-dimensional one-hot encoded vectors encoding the seven different atoms in the dataset. The architecture of this GCN mirrored that used for the Acyclic classification task, with the addition of another GCNConv layer with 64 filters and LeakyReLU as the activation function. With this architecture, we achieved 89% accuracy on the MUTAG classification task. The GFlowNet architecture was also the same as the one used for the Acyclic classification task, except we used seven nodes representing the different atoms as the building blocks. The initial graph G_0, used in training the GFlowNet, was set to a single node graph with the feature value set to carbon, as in [45].

5 Results

5.1 Verifying XInsight's Generative Abilities Using the Acyclic Dataset

To verify that XInsight is capable of generating graphs of a particular class defined by a classification model, we conducted an experiment in which we trained XInsight to generate graphs from a graph convolutional network, previously trained on the Acyclic Graphs dataset. This synthetic dataset contains two classes, acyclic and cyclic, and is described in detail in Sect. 4.1.

Following XInsight training, we generated a distribution of 16 graphs (8 acyclic and 8 cyclic), shown in Fig. 1. The results of the experiment indicate that XInsight is indeed capable of generating acyclic graphs, which is consistent with the nature of the dataset. This provides evidence that XInsight is capable of generating graphs guided by the predictions of a simple classification model.

5.2 Revealing Distinct Relationships Learned by the MUTAG Classifier

Generating Explanations. In our second experiment, we trained XInsight to explain a GCN trained on the MUTAG dataset. Our objective was to uncover hidden patterns and relationships that the trained GCN classifier associates with the mutagenic class. To achieve this, we used XInsight to generate a distribution of 16 compounds, illustrated in Fig. 2, and then fed the generated graphs through the trained GCN to produce graph embeddings. In order to visualize the 32-dimensional graph embeddings we used the UMAP dimensionality reduction algorithm, which preserves global and local structure of the data [23], to project the embeddings onto a 2-dimensional plane. From this visualization, we identified five distinct groupings of compounds that we hypothesize group by an unknown factor related to mutagenicity. To uncover the factor behind these groupings, we continued our analysis by analyzing the chemical properties of the generated compounds using QSAR modeling [14] (Fig. 3).

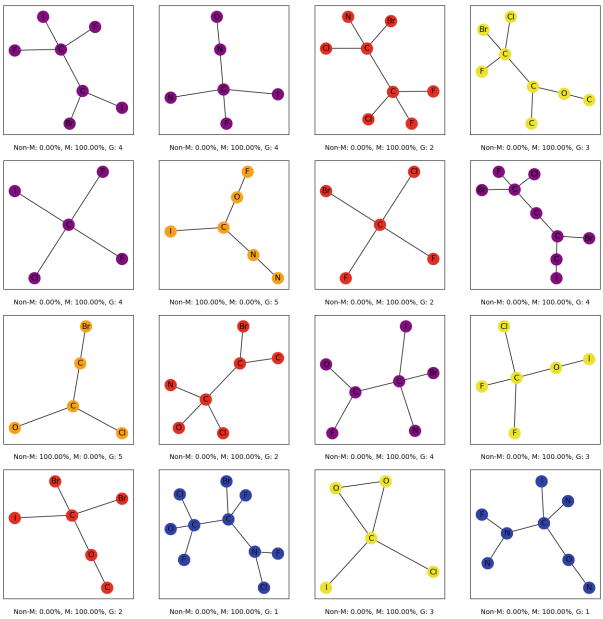

Fig. 2. Distribution of explanations for the *Mutagenic* classifier generated by the trained XInsight model, with MUTAG class probabilities according to the trained proxy. Colors represent UMAP clusters of graph embeddings for the generated compounds. Blue: Group 1, Red: Group 2, Yellow: Group 3, Purple: Group 4, Orange: Group 5. (Color figure online)

Knowledge Discovery. Quantitative Structure-Activity Relationship (QSAR) modeling is a well-established methodology that is used to differentiate between mutagenic and non-mutagenic compounds, which have been identified by the Ames test [14]. Various features of typical drugs, such as lipophilicity, polarizability, hydrophilicity, electron density, and topological analysis, have been utilized in the literature to establish QSAR models for mutagenicity [35]. Among these features, lipophilicity has been identified as a major contributing factor for mutagenicity, as it facilitates the penetration of lipophilic compounds through cellular membranes.

To establish a relationship between the clusters of compounds generated by XInsight and their mutagenicity, we calculated the lipophilicity of all the generated structures using the XLOGP3 method [6], samples shown in Fig. 4. This method has been shown to provide reliable results that are comparable to those obtained using the calculation of the octanol water partition coefficient for *logP* [37]. It is essential to note that we added hydrogens to O (-OH) and N (-NH2) groups to represent the aqueous environment within the human body, since hydrogen atoms were not included in the building blocks for the MUTAG dataset.

In Fig. 5 we see that in general the lipophilicity value is higher for the generated mutagenic compounds compared to the non-mutagenic compounds. We

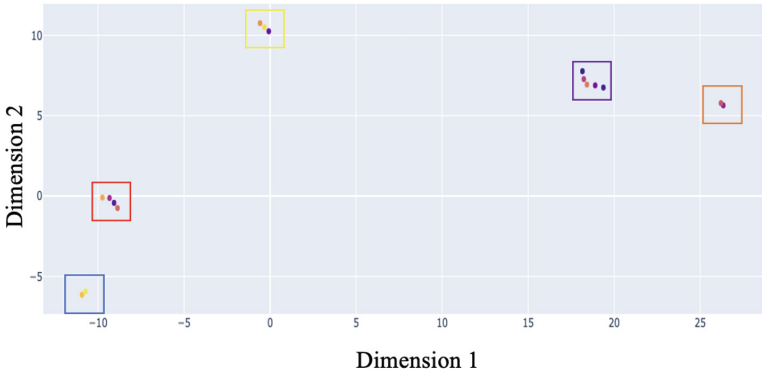

Fig. 3. Generated graph embeddings projected onto 2-dimensional plane using UMAP. UMAP was fit using the cosine similarity metric, 2 neighbors, and a minimum distance of 0.1.

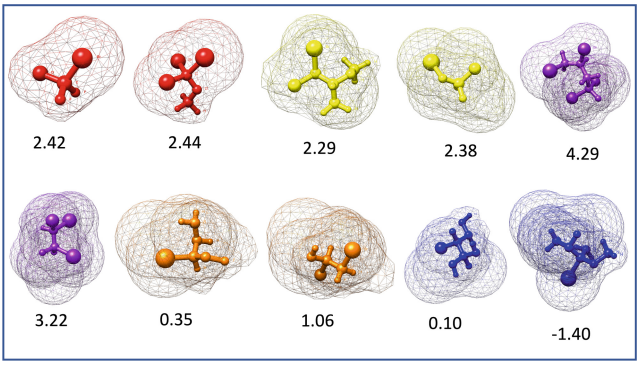

Fig. 4. Lipophilicity calculations for 10 of the clustered compounds generated by XInsight using the XLOGP3 method. Surface mesh and 3 dimensional structures were generated by Chimera visualization software [29].

observed that the highest lipophilicity was associated with compounds of Group 4 (purple), followed by those of Group 2 (red) and Group 3 (yellow). The purple cluster exhibited significant differences in lipophilicity when compared to the red and yellow clusters, which explains why purple is a distinct cluster. However, groups 1 (blue) and 5 (orange) showed lower levels of lipophilicity values but still exhibited significant differences. Thus, lipophilicity appears to be a factor related to the mutagenicity of these compounds.

Knowledge Verification. To verify that the discovered relationship between lipophilicity and mutagenicity is valid, we randomly sampled 32 compounds from the MUTAG dataset, with 16 compounds for each class, and calculated lipophilicity for each. We then performed a t-test to determine whether there is a statistically significant difference in lipophilicity for mutagenic and non-mutagenic compounds. In Table 1 we show a statistically significant difference between the mean lipophilicity values for the mutagenic and non-mutagenic classes, thus verifying that the relationship uncovered using XInsight's generated distribution is a true relationship exhibited in the training data. Additionally, this shows how XInsight can be used to discover knowledge about the model.

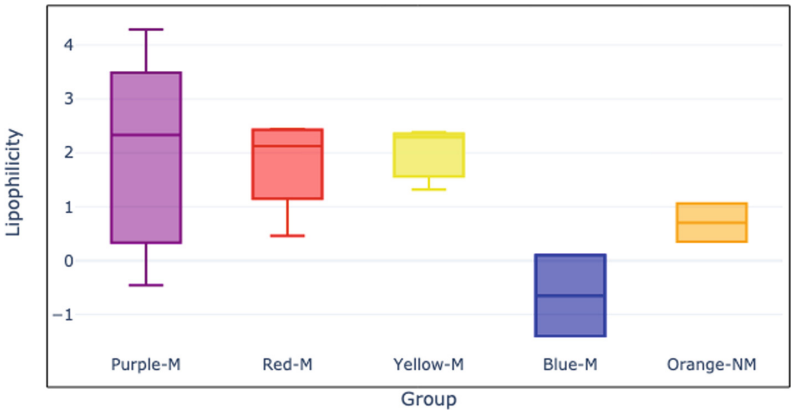

Fig. 5. XLOGP3 Lipophilicity values for UMAP clustered compounds colored by group with classified labels, Mutagenic: *M*, Non-Mutagenic: *NM*. (Color figure online)

Further Insights. The distribution of explanations provided us with another significant insight, which is that the compounds in Group 1 (blue) and Group 5 (orange) have low lipophilicity, even though Group 1 is classified as mutagenic and Group 5 as non-mutagenic. This raises two possible assumptions: first, the classifier might be incorrectly classifying compounds that are similar to those in Groups 1 as mutagenic, or second, there might be another underlying factor that is responsible for the hydrophilic nature of these compounds. Furthermore, as mentioned earlier, lipophilicity is not the only factor determining the mutagenicity of the compounds. To explain the clustering of the Group 1, additional quantum-mechanical calculations are necessary.

This analysis underscores the considerable advantages of generating a distribution of explanations, as opposed to a single explanation that maximizes the reward. By having a distribution of explanations, we can uncover hidden insights into what the classification model associates with the target class. Without a distribution of explanations, we are restricted in the types of analysis we can perform to more effectively explain the model being examined.

Table 1. *t-test* results showing a statistically significant difference between Mutagenic and Non-Mutagenic lipophilicity values for 32 randomly sampled compounds from the MUTAG dataset, $\alpha = 0.05$.

	Mutagenic	Non-Mutagenic
Mean	4.1444	2.1750
Variance	0.6812	1.3963
t-statistic	−5.2917	
p-value	0.00001022	

6 Conclusion

In this paper, we proposed XInsight, a novel explainability algorithm for graph neural networks, that generates a diverse set of model explanations using Generative Flow Networks. Our approach is designed to provide human-understandable explanations for GNNs that uncover the hidden relationships of the model. We demonstrated the effectiveness of XInsight by generating explanations for GNNs trained for two graph classification tasks, including the classification of acyclic graphs and the classification of mutagenic compounds. Our results indicate that XInsight uncovers underlying relationships and patterns demonstrated by the model, and provides valuable guidance for further analysis.

Our findings emphasize the importance of generating a diverse set of explanations, as it enables us to discover hidden relationships in the model and identify important features in the data. Furthermore, we show that the generated explanations from XInsight can be used in combination with data mining and chemical analysis methods to uncover relationships within the model. For instance, we analyzed the generated compounds from XInsight using QSAR modeling, and we observe that XInsight generates compounds that cluster by Lipophilicity, a known correlate of mutagenicity.

Overall, XInsight provides a promising direction for developing explainable AI algorithms for graph-based applications, with implications for many real-world domains. We believe that XInsight has the potential to make a significant impact in various real-world domains, particularly in high-stakes applications, such as drug discovery, where interpretability and transparency are essential.

References

1. Ahmedt-Aristizabal, D., Armin, M.A., Denman, S., Fookes, C., Petersson, L.: Graph-based deep learning for medical diagnosis and analysis: past, present and future. Sensors **21**(14), 4758 (2021). https://doi.org/10.3390/s21144758. https://www.mdpi.com/1424-8220/21/14/4758
2. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Mueller, K.R.: How to explain individual classification decisions (2009). https://doi.org/10.48550/arXiv.0912.1128. http://arxiv.org/abs/0912.1128. arXiv:0912.1128

3. Bengio, E., Jain, M., Korablyov, M., Precup, D., Bengio, Y.: Flow network based generative models for non-iterative diverse candidate generation (2021). https://doi.org/10.48550/arXiv.2106.04399. http://arxiv.org/abs/2106.04399. arXiv:2106.04399
4. Bengio, Y., Lahlou, S., Deleu, T., Hu, E.J., Tiwari, M., Bengio, E.: GFlowNet Foundations (2022). https://doi.org/10.48550/arXiv.2111.09266. arXiv:2111.09266
5. Bongini, P., Bianchini, M., Scarselli, F.: Molecular graph generation with Graph Neural Networks. Neurocomputing **450**, 242–252 (2021). https://doi.org/10.1016/j.neucom.2021.04.039. http://arxiv.org/abs/2012.07397, arXiv:2012.07397
6. Cheng, T., et al.: Computation of octanol-water partition coefficients by guiding an additive model with knowledge. J. Chem. Inf. Model. **47**(6), 2140–2148 (2007). https://doi.org/10.1021/ci700257y. https://pubs.acs.org/doi/10.1021/ci700257y
7. Debnath, A.K., Lopez De Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. J. Med. Chem. **34**(2), 786–797 (1991). https://doi.org/10.1021/jm00106a046
8. Dosovitskiy, A., Brox, T.: Inverting visual representations with convolutional networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4829–4837. IEEE, Las Vegas, NV, USA (2016). https://doi.org/10.1109/CVPR.2016.522
9. Fan, W., et al.: Graph Neural Networks for Social Recommendation (2019). https://doi.org/10.48550/arXiv.1902.07243. http://arxiv.org/abs/1902.07243. arXiv:1902.07243
10. Gao, J., et al.: VectorNet: encoding HD maps and agent dynamics from vectorized representation (2020). https://doi.org/10.48550/arXiv.2005.04259. arXiv:2005.04259
11. Gasteiger, J., Groß, J., Günnemann, S.: Directional message passing for molecular graphs (2022). https://doi.org/10.48550/arXiv.2003.03123. http://arxiv.org/abs/2003.03123. arXiv:2003.03123
12. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry (2017). https://doi.org/10.48550/arXiv.1704.01212. arXiv:1704.01212
13. Hagberg, A., Swart, P., S Chult, D.: Exploring network structure, dynamics, and function using networkx (2008). https://www.osti.gov/biblio/960616
14. Honma, M., et al.: Improvement of quantitative structure-activity relationship (QSAR) tools for predicting Ames mutagenicity: outcomes of the Ames/QSAR International Challenge Project. Mutagenesis **34**(1), 3–16 (2019). https://doi.org/10.1093/mutage/gey031. https://academic.oup.com/mutage/article/34/1/3/5142926
15. Huang, Q., Yamada, M., Tian, Y., Singh, D., Yin, D., Chang, Y.: GraphLIME: local interpretable model explanations for graph neural networks (2020). https://doi.org/10.48550/arXiv.2001.06216. http://arxiv.org/abs/2001.06216. arXiv:2001.06216
16. Jain, M., et al.: Biological sequence design with GFlowNets (2022). https://arxiv.org/abs/2203.04115v2
17. Laird, E.: Acyclic graph dataset. https://github.com/elilaird/acyclic-graph-dataset
18. Li, Y., Qian, B., Zhang, X., Liu, H.: Graph neural network-based diagnosis prediction. Big Data **8**(5), 379–390 (2020). https://doi.org/10.1089/big.2020.0070. https://www.liebertpub.com/doi/10.1089/big.2020.0070

19. Lu, H., Uddin, S.: A weighted patient network-based framework for predicting chronic diseases using graph neural networks. Sci. Rep. **11**(1), 22607 (2021). https://doi.org/10.1038/s41598-021-01964-2. https://www.nature.com/articles/s41598-021-01964-2
20. Luo, D., et al.: Parameterized explainer for graph neural network. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS 2020, Curran Associates Inc., Red Hook, NY, USA (2020)
21. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5188–5196. IEEE, Boston, MA, USA (2015). https://doi.org/10.1109/CVPR.2015.7299155
22. Malkin, N., Jain, M., Bengio, E., Sun, C., Bengio, Y.: Trajectory balance: improved credit assignment in GFlowNets (2022). https://doi.org/10.48550/arXiv.2201.13259. http://arxiv.org/abs/2201.13259. arXiv:2201.13259
23. McInnes, L., Healy, J., Melville, J.: UMAP: uniform manifold approximation and projection for dimension reduction (2020). https://doi.org/10.48550/arXiv.1802.03426. arXiv:1802.03426
24. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: TUDataset: a collection of benchmark datasets for learning with graphs (2020). https://doi.org/10.48550/arXiv.2007.08663. arXiv:2007.08663
25. Narla, A., Kuprel, B., Sarin, K., Novoa, R., Ko, J.: Automated classification of skin lesions: from pixels to practice. J. Invest. Dermatol. **138**(10), 2108–2110 (2018). https://doi.org/10.1016/j.jid.2018.06.175. https://linkinghub.elsevier.com/retrieve/pii/S0022202X18322930
26. Nguyen, A., Clune, J., Bengio, Y., Dosovitskiy, A., Yosinski, J.: Plug & play generative networks: conditional iterative generation of images in latent space. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3510–3520. IEEE, Honolulu, HI (2017). https://doi.org/10.1109/CVPR.2017.374
27. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 427–436. IEEE, Boston, MA, USA (2015). https://doi.org/10.1109/CVPR.2015.7298640
28. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill **2**(11), e7 (2017). https://doi.org/10.23915/distill.00007
29. Pettersen, E.F., et al.: UCSF chimera?A visualization system for exploratory research and analysis. J. Comput. Chem. **25**(13), 1605–1612 (2004). https://doi.org/10.1002/jcc.20084. https://onlinelibrary.wiley.com/doi/10.1002/jcc.20084
30. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. Int. J. Comput. Vis. **128**(2), 336–359 (2020). https://doi.org/10.1007/s11263-019-01228-7. http://arxiv.org/abs/1610.02391. arXiv:1610.02391
31. Sheng, Z., Xu, Y., Xue, S., Li, D.: Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving. IEEE Trans. Intell. Transp. Syst. **23**(10), 17654–17665 (2022). https://doi.org/10.1109/TITS.2022.3155749. http://arxiv.org/abs/2109.12764. arXiv:2109.12764
32. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps (2014). https://doi.org/10.48550/arXiv.1312.6034. arXiv:1312.6034
33. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: the all convolutional net (2015). https://doi.org/10.48550/arXiv.1412.6806. http://arxiv.org/abs/1412.6806. arXiv:1412.6806

34. Tan, Q., Liu, N., Hu, X.: Deep representation learning for social network analysis. Front. Big Data **2**, 2 (2019). https://doi.org/10.3389/fdata.2019.00002.https://www.frontiersin.org/article/10.3389/fdata.2019.00002/full
35. Tuppurainen, K.: Frontier orbital energies, hydrophobicity and steric factors as physical QSAR descriptors of molecular mutagenicity. A review with a case study: MX compounds. Chemosphere **38**(13), 3015–3030 (1999). https://doi.org/10.1016/S0045-6535(98)00503-7.https://linkinghub.elsevier.com/retrieve/pii/S0045653598005037
36. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lió, P., Bengio, Y.: Graph attention networks (2018). https://doi.org/10.48550/arXiv.1710.10903. arXiv:1710.10903
37. Viana, R.D.S., Aquino, F.L.T.D., Barreto, E.: Effect of trans -cinnamic acid and p -coumaric acid on fibroblast motility: a pilot comparative study of in silico lipophilicity measure. Nat. Prod. Res. **35**(24), 5872–5878 (2021). https://doi.org/10.1080/14786419.2020.1798664. https://www.tandfonline.com/doi/full/10.1080/14786419.2020.1798664
38. Vu, M.N., Thai, M.T.: PGM-explainer: probabilistic graphical model explanations for graph neural networks (2020). https://doi.org/10.48550/arXiv.2010.05788.arXiv:2010.05788
39. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 165–174 (2019). https://doi.org/10.1145/3331184.3331267. http://arxiv.org/abs/1905.08108. arXiv:1905.08108
40. Wu, J., et al.: Self-supervised graph learning for recommendation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 726–735 (2021). https://doi.org/10.1145/3404835.3462862. http://arxiv.org/abs/2010.10783. arXiv:2010.10783
41. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How Powerful are Graph Neural Networks? (2019). https://doi.org/10.48550/arXiv.1810.00826. arXiv:1810.00826
42. Ying, R., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: GNNExplainer: generating explanations for graph neural networks. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA (2019)
43. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 974–983 (2018). https://doi.org/10.1145/3219819.3219890. http://arxiv.org/abs/1806.01973. arXiv:1806.01973
44. You, J., Liu, B., Ying, R., Pande, V., Leskovec, J.: Graph convolutional policy network for goal-directed molecular graph generation. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6412–6422. NIPS 2018, Curran Associates Inc., Red Hook, NY, USA (2018)
45. Yuan, H., Tang, J., Hu, X., Ji, S.: XGNN: towards model-level explanations of graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 430–438. ACM, Virtual Event CA USA (2020). https://doi.org/10.1145/3394486.3403085
46. Yuan, H., Yu, H., Gui, S., Ji, S.: Explainability in graph neural networks: a taxonomic survey. IEEE Trans. Pattern Anal. Mach. Intell. 45, 1–19 (2022). https://doi.org/10.1109/TPAMI.2022.3204236. https://ieeexplore.ieee.org/document/9875989/

47. Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., Bengio, Y.: Generative flow networks for discrete probabilistic modeling (2022). https://arxiv.org/abs/2202.01361v2
48. Zhang, H., Lu, G., Zhan, M., Zhang, B.: Semi-supervised classification of graph convolutional networks with Laplacian rank constraints. Neural Process. Lett. **54**(4), 2645–2656 (2022). https://doi.org/10.1007/s11063-020-10404-7
49. Zhang, Z., et al.: Graph neural network approaches for drug-target interactions. Curr. Opin. Struct. Biol. **73**, 102327 (2022). https://doi.org/10.1016/j.sbi.2021.102327. https://linkinghub.elsevier.com/retrieve/pii/S0959440X2100169X