Private Graph Extraction via Feature Explanations

Iyiola E. Olatunji
L3S Research Center
Hannover, Germany
iyiola@l3s.de

Mandeep Rathee
L3S Research Center
Hannover, Germany
rathee@l3s.de

Thorben Funke
L3S Research Center
Hannover, Germany
tfunke@l3s.de

Megha Khosla
TU Delft
Delft, Netherlands
m.khosla@tudelft.nl

ABSTRACT

Privacy and interpretability are two important ingredients for achieving trustworthy machine learning. We study the interplay of these two aspects in graph machine learning through graph reconstruction attacks. The goal of the adversary here is to reconstruct the graph structure of the training data given access to model explanations. Based on the different kinds of auxiliary information available to the adversary, we propose several graph reconstruction attacks. We show that additional knowledge of post-hoc feature explanations substantially increases the success rate of these attacks. Further, we investigate in detail the differences between attack performance with respect to three different classes of explanation methods for graph neural networks: gradient-based, perturbation-based, and surrogate model-based methods. While gradient-based explanations reveal the most in terms of the graph structure, we find that these explanations do not always score high in utility. For the other two classes of explanations, privacy leakage increases with an increase in explanation utility. Finally, we propose a defense based on a randomized response mechanism for releasing the explanations, which substantially reduces the attack success rate. Our code is available at https://github.com/iyempissy/graph-stealing-attacks-with-explanation.

KEYWORDS

privacy risk, model explanations, graph reconstruction attacks, private graph extraction, graph neural networks, attacks

1 INTRODUCTION

Graphs are highly informative, flexible, and natural ways of representing data in various real-world domains. Graph neural networks (GNNs) [19, 23, 37] have emerged as the standard tool to analyze graph data that is non-euclidean and irregular in nature. GNNs have gained state-of-the-art results in various graph analytical tasks ranging from applications in biology, and healthcare [2, 7] to recommending friends in a social network [12]. GNNs' success can be attributed to their ability to extract powerful latent features via complex aggregation of neighborhood aggregations [16, 46].

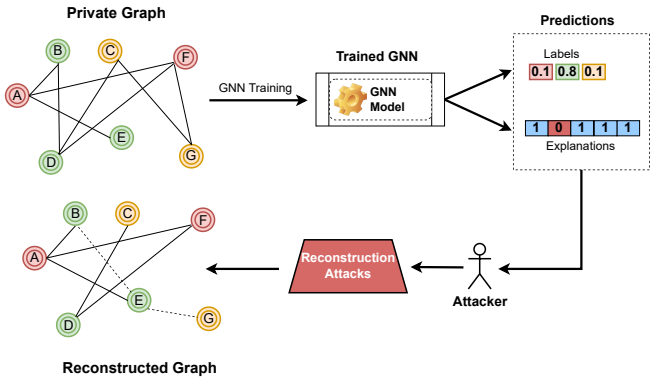


Figure 1: The importance scores for the features, as provided by an explanation, can be exploited to infer the graph structure. Here, we show an example of a binary explanation where a score of 1 indicates that the corresponding feature is part of the explanation.

However, these models are inherently black-box and complex, making it extremely difficult to understand the underlying reasoning behind their predictions. With the growing adoption of these models in various sensitive domains, efforts have been made to explain their decisions in terms of feature as well as neighborhood attributions. Model explanations can offer insights into the internal decision-making process of the model, which builds the trust of the users. Moreover, owing to the current regulations [28] and guidelines for designing trustworthy AI systems, several proposals advocate for deploying (automated) model explanations [17, 31].

Nevertheless, releasing additional information, such as explanations, can have adverse effects on the privacy of the training data. While the risk to privacy due to model explanations exists for machine learning models in general [34], it can have more severe implications for graph neural networks. For instance, several works [8, 27] have established the increased vulnerability of GNNs to privacy attacks due to the additional encoding of graph structure in the model itself. We initiate the *first investigation* of the effect of releasing feature explanations for graph neural networks on *the leakage of private information* in the training data.

To analyze the information leakage due to explanations, we take the perspective of an adversary whose goal is to infer the hidden connections among the training nodes. Consider a setting where

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
Proceedings on Privacy Enhancing Technologies 2023(2), 59–78
© 2023 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2023-0041

the user has access to node features and labels but not the graph structure among the nodes. For example, node features could be part of the public profile of various individuals. The graph structure among the nodes could be some kind of social network which is private. Now, let's say that the user wants to obtain a trained GNN on her data while providing the node features and labels to a central authority that has access to the private graph structure. We ask here the question: *how much do the feature-based explanations leak information about the graph structure used to train the GNN model?* We quantify this information leakage via several graph reconstruction attacks. A visual illustration is shown in Figure 1. Specifically, our threat model consists of two main settings: first, where the adversary has access only to feature explanations, and second, where the adversary has additional information on node features/labels. Note that we only focus on feature-based explanations for GNNs in this work. For other explanation types such as node or edge explanations, as the returned nodes and edges belong to the node's neighborhood, it becomes trivial to reconstruct the graph structure.

1.1 Our Contributions and Findings

Ours is the first work to analyze the risks of releasing feature explanations in GNNs on the privacy of the relationships/connections in the training nodes. We quantify the information leakage via explanations by the success rate of several graph reconstruction attacks. Our attacks range from a simple explanation similarity-based attack to more complex attacks exploiting graph structure learning techniques. Besides, we provide a thorough analysis of information leakage via feature-based explanations produced by three classes of GNN post-hoc explanation methods, including *gradient-based*, *perturbation-based*, and *surrogate model-based*.

To analyze the differences in the robustness of the explanation methods to our privacy attacks, we investigate the explanation utility in terms of *faithfulness* and *sparsity*. We find that the gradient-based methods are the most susceptible to graph reconstruction attacks even though the corresponding explanations are the least faithful to the model. In other words, they have low utility. This is an important finding as the corresponding explanations could release a large amount of private information without offering any utility to the user. The perturbation-based approach Zorro and its variant show the highest explanation utility as well as a high success rate of the attack, pointing to the expected trade-off between privacy and explanation utility.

We perform our study over three types of datasets with varying properties. For instance, our first dataset (Cora) has a large number of binary features, but the feature space is very sparse. Our second dataset (CoraML) has fewer but denser features. Our final dataset (Bitcoin) has a very small number of features. We find that the information leakage varies with explanation techniques as well as the feature size of the dataset. The dataset (Bitcoin) with the smallest feature size (8) is the most difficult to attack. Through experiments on three additional datasets (CiteSeer, PubMed, and Credit), we later show that although a small feature size might limit the information leakage, the correlation between node features with node connections plays a significant role. All baseline attacks which rely on knowledge of only features and labels perform no better than a random guess. In such a case, explanation-based attacks provide an improvement, though not very huge, in inferring private graph structure information. We also perform additional experiments to evaluate the effectiveness of the attacks on three other datasets. Also, we perform a subgraph reconstruction attack when only a subset of the explanations is available to the attacker.

Finally, we develop a perturbation-based defense for releasing feature-based explanations. Our defense employs a randomized response mechanism to perturb the individual explanation bits. We show that our defense reduces the attack to a random guess with a small drop in the explanation utility. Our code is available at https://github.com/iyempissy/graph-stealing-attacks-with-explanation.

2 PRELIMINARIES

2.1 Graph Neural Networks

Graph neural networks (GNNs) [19, 23, 37] are a special family of deep learning models designed to perform inference on graph-structured data. The variants of GNNs, such as graph convolutional network (GCN), compute the representation of a node by utilizing its feature representation and that of the neighboring nodes. That is, GNNs compute node representations by recursive aggregation and transformation of feature representations of its neighbors.

Let $G = (V, E)$ denote a graph where V is the node-set, and E represents the edges or links among the nodes. Furthermore, let $\boldsymbol{x}_i^{(\ell)}$ be the feature representation of node i at layer ℓ, $\mathcal{N}_i$ denote the set of its 1-hop neighbors and $\boldsymbol{\theta}$ is a learnable weight matrix. Formally, the ℓ-th layer of a graph convolutional operation can be described as

$$\boldsymbol{z}_i^{(\ell)} = \text{AGGREGATION}^{(\ell)}\left(\left\{\boldsymbol{x}_i^{(\ell-1)}, \left\{\boldsymbol{x}_j^{(\ell-1)} \mid j \in \mathcal{N}_i\right\}\right\}\right) \quad (1)$$

$$\boldsymbol{x}_i^{(\ell)} = \text{TRANSFORMATION}^{(\ell)}\left(\boldsymbol{z}_i^{(\ell)}\right) \quad (2)$$

Then a softmax layer is applied to the node representations at the last layer (say L) for the final prediction of the node classes C

$$\boldsymbol{y} \leftarrow \text{argmax}(\text{softmax}(\boldsymbol{z}_i^{(L)}\boldsymbol{\theta})), \quad (3)$$

GNNs have been shown to possess increased vulnerability to privacy attacks due to encoding of additional graph structure in the model [8, 27]. We further investigate the privacy risks of releasing post-hoc explanations for GNN models.

2.2 Explaining Graph Neural Networks

GNNs are deep learning models which are inherently black-box or non-interpretable. This black-box behavior becomes more critical for applying them in sensitive domains like medicine, crime, and finance. Consequently, recent works have proposed post-hoc explainability techniques to explain the decisions of an already-trained model. In this work, we are concerned with the task of node classification, where the model is trained to predict node labels in a graph. For such a task, an instance is a single node. An explanation for a node usually consists of a subset of the most important features as well as a subset of its neighboring nodes/edges responsible for the model's prediction. Depending on the explanation method, the importance is usually quantified either as a continuous score (also referred to as a soft mask) or a binary score (also called a hard

mask). In this work, we consider three popular classes of explanation methods: *gradient-based* [3, 29, 32], *perturbation-based* [16, 46], and *surrogate* [21] methods.

Gradient-based Methods. These approaches usually employ the gradients of the target model's prediction with respect to input features as importance scores for the node features. We use two gradient-based methods in our study, namely **Grad**[35] and Grad-Input (**Grad-I**) [36]. For a given graph G and the trained GNN model $f(\boldsymbol{X}, G, \theta)$ (where θ is the set of parameters and $\boldsymbol{X}$ is the features matrix), Grad generates an explanation $\mathcal{E}_X$ by assigning continuous valued importance scores to the features. For node i and $\boldsymbol{x_i}$ is its features vector, The score is calculated by $\frac{\partial f}{\partial \boldsymbol{x_i}}$. Grad-I transforms Grad explanation by an element-wise multiplication with the input features ($\boldsymbol{x_i} \odot \frac{\partial f}{\partial \boldsymbol{x_i}}$).

Perturbation-based Methods. Perturbation-based methods obtain soft or hard masks over the features/nodes/edges as explanations by monitoring the change in prediction with respect to different input perturbations. We use two methods from this class: **Zorro** [16] and **GNNExp** [46]. Zorro learns discrete masks over input nodes and node features as explanations using a greedy algorithm. It optimizes a fidelity-based objective that measures how the new predictions match the original predictions of the model by fixing the selected nodes/features and replacing the others with random noise values. This returns a hard mask for the explanations. GNNExp learns soft masks over edges and features by minimizing the cross-entropy loss between the predictions of the original graph and the predictions of the newly obtained (masked) graph. We also utilize the Zorro variant that provides soft explanation masks called **Zorro-S**. Zorro-S relaxes the argmax in Zorro's objective with a softmax, such that the masked are retrievable with standard gradient-based optimization. Together with the regularization terms of GNNExp, Zorro-S learns sparse soft masks.

Surrogate Methods. Surrogate methods fit a simple and interpretable model to a sampled local dataset corresponding to the query node. For example, the sampled dataset can be generated from the neighbors of the given query node. The explanations from the surrogate model are then used to explain the original predictions. We use GraphLime[21], which we denote as **GLime** from this class. As an interpretable model, it uses the global feature selection method HSIC-Lasso [44]. The sampled dataset consists of the node and its neighborhood. The set of the most important features returned by HSIC-Lasso is used as an explanation for the GNN model.

Remark: Please note that in this work, we assume that only feature-based explanations are released to the user. In the presence of node/edge explanations, an adversary can trivially reconstruct large parts of the neighborhood as the returned nodes/edges are part of the node's original neighborhood.

2.2.1 Measuring Explanation Quality. We measure the quality of the explanation by its ability to approximate the model's behavior which is referred to as ***faithfulness***. As the groundtruth for explanations is not available, we use the *RDT-Fidelity* proposed by [16] to measure faithfulness. The corresponding fidelity score measure how the original and new predictions match by fixing the selected nodes/features and replacing the others with random noise values. Formally, the *RDT-Fidelity* of explanation $\mathcal{E}_X$ corresponding to explanation mask $M(\mathcal{E}_X)$ with respect to the GNN f and the noise distribution $\mathcal{N}$ is given by

$$\mathcal{F}(\mathcal{E}_X) = \mathbb{E}_{Y_{\mathcal{E}_X}|Z\sim\mathcal{N}}\left[\mathbb{1}_{f(X)=f(Y_{\mathcal{E}_X})}\right], \tag{4}$$

where the perturbed input is given by

$$\tilde{I}_{\mathcal{E}_X} = X \odot M(\mathcal{E}_X) + Z \odot (\mathbb{1} - M(\mathcal{E}_X)), Z \sim \mathcal{N}, \tag{5}$$

where $\odot$ denotes an element-wise multiplication, and $\mathbb{1}$ is a matrix of ones with the corresponding size.

Sparsity. We further note that by definition, the complete input is faithful to the model. Therefore, we further measure the sparsity of the explanation. A meaningful explanation should be sparse and only contain a small subset of features most predictive of the model decision. We use the entropy-based sparsity definition from [16] as it is applicable for both soft and hard explanation masks. Let p be the normalized distribution of explanation (feature) masks. Then the sparsity of an explanation is given by the entropy $H(p)$ over the mask distribution,

$$H(p) = -\sum_{f\in M} p(f)\log p(f).$$

Note that the entropy here is bounded by $\log(|M|)$, where M corresponds to the size of the feature set. The lower the entropy, the sparser the explanation. We will utilize these two metrics used in measuring explanation quality to argue about the differences in the attack performance.

3 THREAT MODEL AND ATTACK METHODOLOGY

3.1 Motivation

We consider the setting in which different data holders hold the features and graph (adjacency matrix) in practice. Specifically, the central trusted server has access to the graph structure and trains a GNN model using the node features and labels provided by the user. The user can further query the trained GNN model by providing node features. As the features are already known to the user, revealing *feature explanations* for the prediction might be considered a safe way to increase the user's trust in the model. We investigate such a scenario and uncover the increased privacy risks of releasing *feature explanations even if the original node features/labels are already known to the adversary.*

Our setting is inspired by previous works and practical scenarios [4, 10, 18, 45, 48]. We elaborate on one such scenario in the following. Consider a multinational company with several subdivisions. For simplicity, we consider two subdivisions (the marketing and product departments). The marketing department collects the interaction between users (edge information), while the product department collects data on user behavior (features). Under privacy law(e.g., GDPR), such user interactions cannot be shared because they are collected for different purposes. However, they aim to train a GNN model jointly since it would benefit both teams. The product department then sends the node features to the marketing department, which in addition, uses their edge information to train a predictive model and releases an API (similar to MLaaS). The API

returns both the prediction for the features and the corresponding feature explanations for such predictions. The product department, having observed how useful the exploratory analysis is, aims to recover the private edges using the information returned by the API (explanations) and their node features.

3.2 Threat Model

We consider two scenarios: **first**, in which the adversary has access to node features and/or labels and obtains additional access to feature explanations, and **second**, in which the adversary has access only to feature explanations. The goal of the adversary is to infer the private connections of the graph used to train the GNN model.

Corresponding to the above settings, we categorize our attacks as *explanation augmentation* (when explanations are augmented with other information to launch the attack) and *explanation-only* attacks. Through our five proposed attacks, we investigate the privacy leakage of the explanations generated by six different explanation methods.

Our simple similarity-based explanation-only attack already allows us to quantify the additional information that the feature-based explanation encodes about the graph structure. Our explanation augmentation attacks are based on the graph structure learning paradigm, which allows us to effectively integrate additional known information in learning the private graph structure. Besides, our explanation augmentation attacks also result in a successfully trained GNN model without the knowledge of the true graph structure, offering an additional advantage to the adversary.

3.3 Attack Methodologies

Here, we provide a detailed description of our two types of attacks. We commence with the *explanation-only attack* in which we utilize only the provided explanation to launch the attack, followed by the *explanation augmentation* attacks in which more information such as node labels or/and features are exploited in addition to the explanation. The taxonomy of our attacks based on the attacker's knowledge is presented in Table 1.

Explanation-only Attack. This is an unsupervised attack in which the attacker only has access to the explanations and does not have access to the features or the labels. The attacker measures the distance between each pair of explanation vectors and assigns an edge between them if their distance is small. The intuition is that the model might assign similar labels to connected nodes, which leads to similar explanations. We experimented with various distance metrics, but cosine similarity performs best across all datasets. We refer to this similarity-based attack as **ExplainSim**. This attack is illustrated in Figure 2. We report the performance using the area under the receiver operating characteristic curve (AUC) and average precision (AP). Rather than choosing one threshold which could be domain-dependent, these metrics allow us to provide a holistic evaluation across the complete range of thresholds.

Explanation Augmentation Attacks. Towards explanation augmentation attacks, we leverage the graph structure learning paradigm of [13]. In particular, we employ two *generator modules* for generating graph edges corresponding to features and explanations, respectively. The generators are trained using feature/explanation

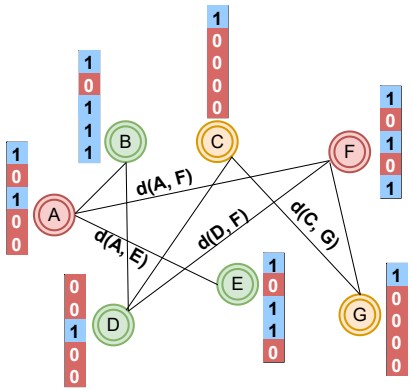


Figure 2: ExplainSim attack. Each node is assigned a feature explanation vector where blue (1) and red (0) indicate whether the feature is part of the explanation or not. The attacker then assigns an edge by computing the pairwise similarity represented as $d(node_i, node_j)$ between nodes' explanation vectors. We show the representation of Zorro's explanation for easier visualization.

reconstruction-based losses and node classification loss. We commence by describing the common architecture of the attack model followed by its concrete usage in the four attack variations in Section 3.3.1.

Generators. The two generators take the node features/explanations as input and output two adjacency matrices. We employ the full parameterization (FP) approach to model the generators similar to those used in [13, 15]. In other words, each element of the adjacency matrix $\tilde{\boldsymbol{A}}$ is treated as a separate learnable parameter, i.e., the adjacency matrix is fully parameterized. We initialize these learnable parameters by the cosine similarity among the feature/explanation vectors of the corresponding nodes. Let the output adjacency matrix function be given as $\tilde{\boldsymbol{A}} = \mathrm{G}_{FP}(\boldsymbol{X}; \boldsymbol{\theta}_{\mathrm{G}}) = \boldsymbol{\theta}_{\mathrm{G}}$ where $\boldsymbol{\theta}_{\mathrm{G}} \in \mathbb{R}^{n \times n}$ and $\mathrm{G}_{FP}(\cdot;\cdot)$ denotes the generator function. To obtain a symmetric adjacency matrix with all positive elements, we perform the following transformation

$$\boldsymbol{A} = \boldsymbol{D}^{-\frac{1}{2}} \left(\frac{\mathrm{P}_{[0,1]}(\tilde{\boldsymbol{A}}) + \mathrm{P}_{[0,1]}(\tilde{\boldsymbol{A}})^T}{2} \right) \boldsymbol{D}^{-\frac{1}{2}},$$

where P is a non-negative function defined by

$$\mathrm{P}_{[0,1]}[x] = \begin{cases} 0 & x < 0, \\ 1 & x > 1, \\ x & otherwise. \end{cases}$$

The final adjacency matrix is computed by adding the matrices corresponding to the two generators. Any element greater than one is trimmed to 1. In other words, a cell in the adjacency matrix follows a Bernoulli distribution. The final graph is then generated by sampling from the learned Bernoulli distributions. More precisely, we select edge (i, j) with probability p where $p \in [0, 1]$ is the learnt weight for cell (i, j).

Training With Self-supervision and Node Classification Loss. We remark that only some nodes used to train the attack

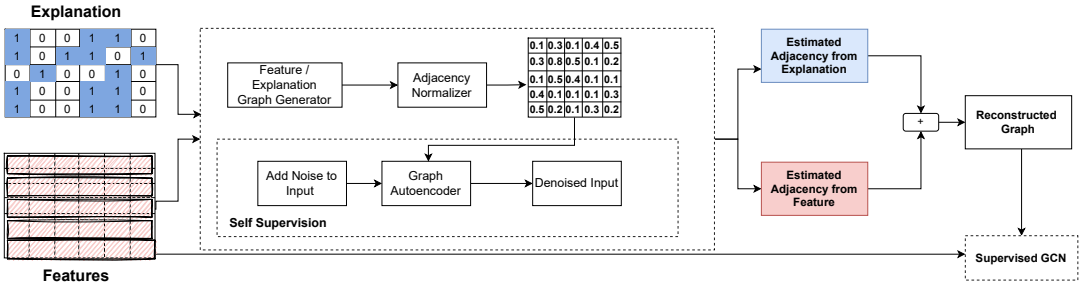


Figure 3: Overview of GSEF. The generator takes node features and explanations as input and outputs an adjacency matrix which may be non-normalized by the adjacency normalizer. The normalized adjacency matrix is used in predicting both the class labels and reconstructing the node features (explanations) by the denoising autoencoders. The final reconstructed adjacency is the one that minimizes the reconstruction error on each of the node features and explanations and the loss of the class label prediction.

model are labeled. To learn effective representations of both labeled and unlabeled nodes, we employ a self-supervision module in addition to training with the supervised classification loss. In particular, for the self-supervision module, we use *denoising graph autoencoders*, which takes noisy features/explanations and the graph sampled from the generator as input. The goal is to reconstruct the true node features and explanations.

Let DAE and $\text{DAE}_{\mathcal{E}_X}$ denote the feature and explanation reconstruction modules respectively. They take the noisy node features(explanations) as input and output a denoised version of the features (explanations) with the same dimension. To generate the noisy features/explanations, we first select random indices. We then add random noise to feature/explanation values corresponding to the selected indices. Concretely, for binary features/explanations, we randomly flip $r = 20\%$ of the index with original values 1 to 0. For features and explanations with continuous values, we add independent Gaussian noise to $r = 20\%$ percent of feature/explanation elements (corresponding to selected indices). To train our self-supervision module, we employ binary cross entropy loss for binary features/explanations and mean squared error for continuous features/explanations. Let $\mathcal{L}_{DAE}$ and $\mathcal{L}_{DAE_{\mathcal{E}_X}}$ denote the loss functions corresponding to feature and explanation reconstruction, respectively.

For supervised training, we employ a graph convolution network (GCN) which takes as input the node features and the graph sampled from the generator to predict node labels. Note that the GCN used here is not the target model. We use the cross-entropy loss (CE) between the predicted labels ($\tilde{Y}$) and the ground-truth labels (Y), $\mathcal{L}_C = CE(Y, \tilde{Y})$.

The final training loss for private graph extraction is then given by

$$\mathcal{L} = \mathcal{L}_{DAE} + \mathcal{L}_{DAE_{\mathcal{E}_X}} + \mathcal{L}_C.$$

We refer to the above attack framework as **G**raph **S**tealing with **E**xplanations and **F**eatures (**GSEF**), and the schematic diagram is given in Figure 3. Besides, we have three attack variations that employ a single generator module, as described below.

3.3.1 Attack Variations. Besides GSEF, we have three attack variations that employ explanations (i) **GSEF-CONCAT** in which we concatenate the node features and explanations and feed the concatenated input to a single generator module (ii) **GSEF-MULT** in which we perform element-wise multiplication between the features and the explanations and feed them into a single graph generator module. This is equivalent to assigning importance to the node features, which emphasize the essential characteristics of the nodes. Similar to GSEF-CONCAT, we reconstruct the adjacency matrix using one generator of Figure 3 and (iii) **GSE** in which the attacker only has access to the explanations and labels. Here, we also employ only one generator with explanations as input.

Table 1: Attack taxonomy based on attacker's knowledge of node features (X), labels (Y) and feature explanations ($\mathcal{E}_X$).

ATTACK	X	Y	$\mathcal{E}_X$
EXPLAINSIM	✗	✗	✓
GSEF	✓	✓	✓
GSEF-CONCAT	✓	✓	✓
GSEF-MULT	✓	✓	✓
GSE	✗	✓	✓

4 EXPERIMENTS

In this section, we present the experimental results to show the effectiveness of explanation-based attacks. Specifically, our experiments are designed to answer the following research questions:

RQ 1. *How does the knowledge of feature explanations influence the reconstruction of private graph structure?*

RQ 2. *What are the differences between explanation methods with respect to privacy leakage?*

RQ 3. *What is the additional advantage of the adversary (for example, in terms of the utility of the inferred information on a downstream task) on explanation augmentation attacks?*

RQ 4. *How much does the lack of knowledge about groundtruth node labels affect attack performance?*

4.1 Experimental Settings

4.1.1 Attack Baselines Without Explanations.

FeatureSim. In this unsupervised attack, the attacker computes the pairwise similarity between pairs of the actual features to reconstruct the graph. Specifically, an edge exists between two nodes if the distance between their feature representation is low. We use cosine similarity as a measure of similarity because it performs better than other distance metrics.

Lsa [20]. Lsa (Link stealing attack) is a black-box attack that assumes that the attacker has access to a dataset drawn from a similar distribution to the target data (shadow dataset). Additionally, Lsa knows the architecture of the target model and can train a corresponding shadow model that replicates the behavior of the target model. The goal of the attack is to infer sensitive links between nodes of interest. We compare our results with their proposed attack-2, where an attacker has access to the node features and labels. They trained a separate MLP model (reference model) using the available target attributes and their corresponding labels to obtain posteriors. Then, Lsa computes the pairwise distance between posteriors obtained from the target model and that of the reference model for the nodes of interest. We use cosine similarity as the distance metric.

GraphMI [48]. GraphMI is a white-box attack in which the attacker has access to the parameters of the target model, node features, all the node labels, and other auxiliary information like edge density. The goal of an attacker is to reconstruct the sensitive links or connections between nodes. The attack model uses the cross entropy loss between the true labels and the output posterior distribution of the target model, along with feature smoothness and adjacency sparsity constraints, to train a fully parameterized adjacency matrix. The graph is then reconstructed using the graph autoencoder module in which the encoder is replaced by learned parameters of the target model, and the decoder is a logistic function.

Slaps [13]. Since our attack model is built on top of the graph structure learning framework of SLAPS, we performed an experiment using the vanilla SLAPS. Given node features and labels, SLAPS aims to reconstruct the graph that works best for the node classification task.

4.1.2 Target GNN Model. We employ a 2-layer graph convolution network (GCN) [23] as our target GNN model. We used a learning rate of 0.001 and trained for 200 epochs.

4.1.3 Evaluation Metrics. Following the existing works [20, 48], we use the area under the receiver operating cost curve (AUC) and average precision (AP) to evaluate our attack. For all experiments (including baselines), we randomly sample an equal number of pairs of connected and unconnected nodes from the original graph to construct the test set. We measure the AUC and the AP on these randomly selected node pairs. We elaborate more in Appendix B. All our experiments were conducted for 10 different instantiations using PyTorch Geometric library [14] on 11GB GeForce GTX 1080 Ti GPU, and we report the mean values across all runs. The standard deviation of the results is in Appendix D.

Table 2: Dataset statistics. $|V|$ and $|E|$ denotes the number of nodes and edges respectively, C, $\mathbf{X}_d$, and deg denotes the number of classes, size of feature dimension and the average degree of the corresponding graph dataset.

	Cora	CoraML	Bitcoin
$\|V\|$	2708	2995	3783
$\|E\|$	10858	8226	28248
$\mathbf{X}_d$	1433	300	8
C	7	7	2
deg	4.00	2.75	7.50

4.1.4 Datasets. We use three commonly used datasets chosen based on varying graph properties, such as their feature dimensions and structural properties. The task on all datasets is node classification. We present the data statistics in Table 2. We also performed additional experiments using PubMed, CiteSeer, and Credit datasets (See Appendix E.1).

Cora. The Cora dataset [33] is a citation dataset where each research article is a node, and there exists an edge between two articles if one article cites the other. Each node has a label that shows the article category. The features of each node are represented by a 0/1-valued word vector, which indicates the word's presence or absence from the article's abstract.

CoraML. In CoraML dataset [33], each node is a research article, and its abstract is available as raw text. In contrast to the above dataset, the raw text of the abstract is transformed into a dense feature representation. We preprocess the text by removing stop words, web page links, and special characters. Then, we generate Word2Vec [25] embedding of each word. Finally, we create the feature vector by taking the average over the embedding of all words in the abstract.

Bitcoin. The Bitcoin-Alpha dataset[24] is a signed network of trading accounts. Each account is represented as a node, and there is a weighted edge between any two accounts, which represents the trust between accounts. The maximum weight value is +10, indicating total trust, and the lowest is -10, which means total distrust. Each node is assigned a label indicating whether or not the account is trustworthy. The feature vector of each node is based on the rating by other users, such as the average positive or negative rating. We follow the procedures in [38] for generating the feature vectors.

4.2 Result Analysis

4.2.1 ***Analysing the Information Leakage by Explanations (RQ 1).*** The detailed results of different attacks are provided in Table 3. Our results show that the explanation-only (ExplainSim) and explanation augmentation (GSEF) attacks for all explanation methods other than GLime and GNNExp outperform all baseline methods and by far reveal the most information about the private graph structure. We attribute the superior performance of GSEF to the multi-task learning paradigm that aims to reconstruct both the features and explanations. Our results also support our assumption

Table 3: Attack performance and baselines. The best performing attack(s) on each explanation method is(are) highlighted in bold, and the second best attack(s) is(are) underlined.

Exp	Attack	Cora		CoraML		Bitcoin	
		AUC	AP	AUC	AP	AUC	AP
Baseline	FeatureSim	0.799	0.827	0.706	0.753	0.535	0.478
	Lsa [20]	0.795	0.810	0.725	0.760	0.532	0.500
	GraphMI [48]	0.856	0.830	0.808	0.814	0.585	0.518
	Slaps [13]	0.736	0.776	0.649	0.702	0.597	0.577
Grad	GSEF-concat	0.734	0.773	0.640	0.705	0.527	0.515
	GSEF-mult	0.678	0.737	0.666	0.730	0.264	0.383
	GSEF	0.948	0.953	**0.902**	0.833	**0.700**	**0.715**
	GSE	0.924	0.939	0.699	0.768	0.229	0.365
	ExplainSim	**0.984**	**0.978**	0.890	**0.891**	0.681	0.644
Grad-I	GSEF-concat	0.734	0.775	0.674	0.734	0.525	0.527
	GSEF-mult	0.691	0.742	0.717	0.756	0.252	0.380
	GSEF	0.949	0.950	0.887	0.832	**0.709**	**0.723**
	GSE	0.903	0.923	0.717	0.781	0.256	0.380
	ExplainSim	**0.984**	**0.979**	**0.903**	**0.899**	0.681	0.644
Zorro	GSEF-concat	0.823	0.860	0.735	0.786	0.575	0.529
	GSEF-mult	0.723	0.756	0.681	0.697	0.399	0.449
	GSEF	**0.884**	**0.880**	0.776	0.820	0.537	0.527
	GSE	0.779	0.810	0.722	0.777	**0.596**	**0.561**
	ExplainSim	0.871	0.873	**0.806**	**0.829**	0.427	0.485
Zorro-S	GSEF-concat	0.907	0.922	0.747	0.791	**0.601**	**0.590**
	GSEF-mult	0.794	0.815	0.712	0.740	0.490	0.491
	GSEF	**0.918**	0.923	**0.776**	**0.819**	0.598	0.565
	GSE	0.893	0.915	0.742	0.784	0.571	0.564
	ExplainSim	0.908	**0.934**	0.732	0.787	0.484	0.496
GLime	GSEF-concat	0.643	0.710	0.610	0.652	0.473	0.493
	GSEF-mult	0.516	0.522	0.517	0.528	0.264	0.371
	GSEF	**0.730**	**0.773**	**0.681**	**0.740**	**0.542**	**0.525**
	GSE	0.558	0.571	0.540	0.555	0.236	0.361
	ExplainSim	0.505	0.524	0.520	0.523	0.504	0.512
GNNExp	GSEF-concat	0.614	0.650	0.653	0.705	0.467	0.489
	GSEF-mult	0.724	0.760	0.637	0.692	0.390	0.454
	GSEF	**0.762**	**0.796**	**0.700**	**0.695**	**0.590**	**0.563**
	GSE	0.517	0.552	0.490	0.508	0.386	0.451
	ExplainSim	0.537	0.541	0.484	0.508	0.551	0.543

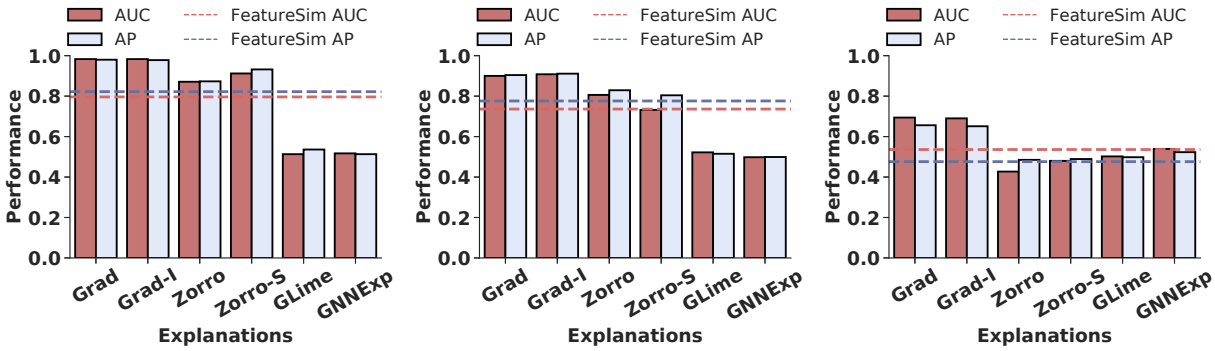
Summary of attack comparisons

- The amount of information in the explanation alone for the graph structure can be quantified using the ExplainSim attack.
- Explanation only (ExplainSim) and explanation augmentation (GSEF) attacks for all explanation methods other than GLime and GNNExp outperform all baseline methods.
- Among the baseline approaches, the white-box access-based attack, GraphMI, performs the best, followed by FeatureSim.
- The relatively good performance of FeatureSim in datasets points to a high correlation of node features with node connections in all datasets other than Bitcoin.
- The information leakage for Bitcoin is limited by small feature size. Our results on additional datasets with varying feature dimensions, as discussed in Section 4.2.2, support our observation about the effect of small feature size on the attack's success.
- For GLime and GNNExp, we observe that the explanation contains little information about the graph structure. The reason behind this is further revealed in the fidelity-sparsity analysis of the obtained explanations.

that a graph structure that is good for predicting the node labels is also suitable for predicting the node features and explanations.

In the following, we provide an in-depth result analysis.

- **Baseline Methods.** Among the baseline methods, GraphMI is the best-performing attack. This is not very surprising as GraphMI has white-box access to the target GNN in addition to access to node features and labels. On the contrary, the FeatureSim attack, which only uses node features, shows competitive performance. This highlights the fact that the features alone are very informative of the graph structure of the studied datasets (except Bitcoin in which all baseline attacks almost fail with an AUC score of close to 0.5).
- **Comparison of the Privacy Leakage via Explanations With That of Features.** Figure 4 compares the performance of two similarity-based attacks using explanations (ExplainSim) and features (FeatureSim), respectively. Note that these attacks do not use any other information except explanations and features, respectively. Hence, allowing us to compare their information content. We note that except for GLime and GNNExp, ExplainSim outperforms FeatureSim for all datasets except Bitcoin. Moreover, for Bitcoin, both attacks fail (with AUC close to 0.5) except for gradient-based explanation methods.
- **Explanation Augmentation With Features and Labels.** Next, we compare the explanation augmentation attack GSEF with the



(a) Cora (b) CoraML (c) Bitcoin

Figure 4: Performance of explanation-only attack (ExplainSim) on the different datasets. The adopted baseline is FeatureSim which performs the pairwise similarities using the true node features.

vanilla graph structure learning approach Slaps, which only uses node features and labels. GSEF outperforms Slaps (Figure 6), which points to the added utility of using explanations to reconstruct the graph structure.

- **Explanation Augmentation Attack Variants.** In Figure 5, we compare the performance of GSE, which have no access to the true features, with Slaps, which utilizes the true features. We observe that on most datasets, the attack significantly outperforms Slaps. This emphasizes that explanations encode the feature information albeit the importance. Comparing the performance of GSEF-concat and GSEF-mult with GSEF on all datasets shows that independently extracting the adjacency matrix from the features and explanations respectively and then combining the two adjacency matrices is better than combining the features and explanations at the input stage.

- **Attack on Bitcoin.** We observe that all baseline attacks fail for Bitcoin. We attribute this to the very small feature size (=8) and the small number of labels (=2). The attacks are not able to exploit the little information exposed by a small set of features/labels. Explanation-based attacks, especially in the case of gradient-based explanations, are more successful than baselines but less than for other datasets.

4.2.2 Results on Additional Datasets. We further investigate our observation of the impact of feature dimension and disasno on the attack success by performing experiments on the additional datasets (namely CiteSeer, PubMed, and Credit) with varying graph properties (details on datasets are provided in Section E.1). The detailed result is in the Appendix in Table 12. We observe that the result on CiteSeer and PubMed supports our observation of the effect of the feature size on the attack success on Cora and CoraML, respectively. First, a dataset with a high feature dimension is most vulnerable to the attack, as seen in the relative performance of CiteSeer on all explanation methods and attacks. Secondly, on the Grad and Grad-I explanations, simply augmenting backpropagation-based explanations with the actual feature via concatenation (GSEF-concat) or element-wise multiplication (GSEF-mult) does not encode the necessary information for launching a successful attack. Instead, it introduces new noise to the reconstructed graph because the explanations are generated such that larger gradients, which only reflect the sensitivity between input and output, may not accurately indicate the importance of the feature. We also observe this phenomenon on other explanations methods, including Zorro, Zorro-S, and GLime, where GSE, which uses only explanations as input, performs better than GSEF-concat and GSEF-mult on the Cora, CoraML, CiteSeer, and PubMed datasets. Thirdly, explanation methods that are robust to input perturbation, such as Zorro and Zorro-S, achieve better attack performance than backpropagation-based explanations on all explanation augmentation attacks. Finally, GLime and GNNExp explanations are more robust to the attacks except on GSEF attack, which achieves a high AUC and AP scores.

Among all the additional datasets, the results of the baseline attacks (FeatureSim, Lsa, GraphMI, and Slaps, which all rely only on feature information) on the Credit dataset are the least. Recall that the Credit dataset has only 13 features. This supports our observation that small feature size affects the attack's success. Notably, the best performing baseline (GraphMI), which utilizes white-box access to the trained model, has a very low AUROC of 0.59 and 0.51 on the Bitcoin and Credit datasets, respectively. This is in contrast to the performance of GraphMI on Cora, CiteSeer, PubMed, and CoraML with AUROC of 0.86, 0.79, 0.80, and 0.81, respectively.

Furthermore, in comparison with Bitcoin, which has only 8 features, we observe that the performance of FeatureSim on the Credit dataset is 34% higher (0.54 on Bitcoin and 0.72 on Credit), which points to high connectivity among similar nodes on the Credit dataset. This, in turn, boosts our attack performance and allows better reconstruction of the private graph. Therefore, the poor performance of our attacks on the Bitcoin dataset is due to disassociativity on the Bitcoin dataset. Moreover, it is worth noting that our explanation augmentation attack is highly successful on the Credit dataset.

4.2.3 ***Differences in Privacy Leakage (RQ 2).*** All explanation method leaks significant information via the reconstruction attacks except for GLime and GNNExp. We observe that for GLime and GNNExp, the explanation-based attacks do not perform better than the baselines, which do not utilize any explanation. Moreover, gradient-based methods are most vulnerable to privacy leakage. To understand the reason behind these observations, we investigate the explanation quality. We measure the goodness of the explanation by its ability to approximate the model's behavior which is also referred to as *faithfulness*. As the groundtruth for explanations is not available, we use the RDT-Fidelity proposed by [16] to measure faithfulness. The results are shown in Table 4. We further note that by definition, the complete input is faithful to the model. Therefore, in addition, we measure the sparsity of the explanation. A meaningful explanation should be sparse and only contain a small subset of features most predictive of the model decision. We use the entropy-based sparsity definition from [16] as it is applicable for both soft and hard explanation masks. The results are shown in Table 4. We analyze the tradeoffs of privacy and explanation goodness in the following.

- **First**, we observe that GNNExp has the lowest sparsity (the higher the entropy, the more uniform the explanation mask distribution, i.e., higher the explanation density). In other words, almost all features are marked equally important. Hence, it is not surprising that it shows a high fidelity. This is the main reason why ExplainSim fails because there is no distinguishing power contained in the explanations.
- **Second**, we observe that gradient-based explanations (Grad and Grad-I) contain the most information about the graph structure though they have low fidelity, i.e., they do not reflect the model's decision process. It appears that these two methods provide the most similar explanations for connected nodes. This is really the worst case when the explanations are not useful but leak maximum private information about the graph structure.
- **Third**, GLime has the highest sparsity and lowest fidelity. GLime runs the HSIC-Lasso feature selection method over a local dataset created using the node and its neighborhood. HSIC-Lasso is known to output a very small set of most predictive features [6] when used for global feature selection. But for the current setting of instance-wise feature selection, i.e., finding the most predictive features for decision over an instance/node, GLime's explanation turns out to be too short, which is neither faithful to the model nor contains any predictive information about the neighborhood.
- **Finally**, the explanations of Zorro and Zorro-S show the highest fidelity and intermediate sparsity, pointing to their high quality. The GSEF attack also obtains high AUC scores for two datasets pointing to the expected increased privacy risk with an increase in explanation utility.

4.2.4 ***Adversary's Advantage in Terms of Trained GNN Model (RQ 3).*** Here, we formalize a quantitative advantage measure that captures the privacy risk posed by the different attacks. The attacker is at an advantage if she can train a well-performing model (on a downstream task) using the reconstructed graph. As the attack models based on graph structure learning implicitly train a GNN on the reconstructed graph, we quantify the attacker's advantage by the performance on the downstream task of node classification.

Table 4: RDT-Fidelity and sparsity (entropy) of different explanation methods. For fidelity, the higher the better. For sparsity, the lower the better

Exp	Cora		CoraML		Bitcoin	
	Fidelity	Sparsity	Fidelity	Sparsity	Fidelity	Sparsity
Grad	0.23	3.99	0.22	5.24	0.83	0.64
Grad-I	0.19	3.99	0.20	5.30	0.82	0.64
Zorro	0.89	1.83	0.96	3.33	0.99	0.37
Zorro-S	0.98	2.49	0.84	2.75	0.95	0.96
GLime	0.19	0.88	0.20	0.98	0.82	0.13
GNNExp	0.74	7.27	0.55	5.70	0.90	2.05

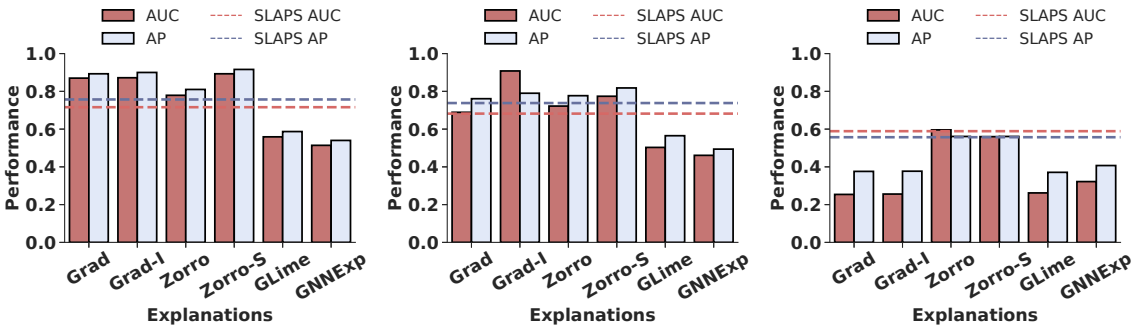
Hypothesis 1. *If the explanations and the reconstructed graph can perform better on a downstream classification task with high confidence, then the reconstructed adjacency is a valid representation of the graph structure. Hence, the attacker has an advantage quantified by Equation 6.*

We define the attacker's advantage as

$$Advantage = \mathcal{R}(f(\mathcal{E}_X; Adj_{rec}; \theta_W), y), \tag{6}$$

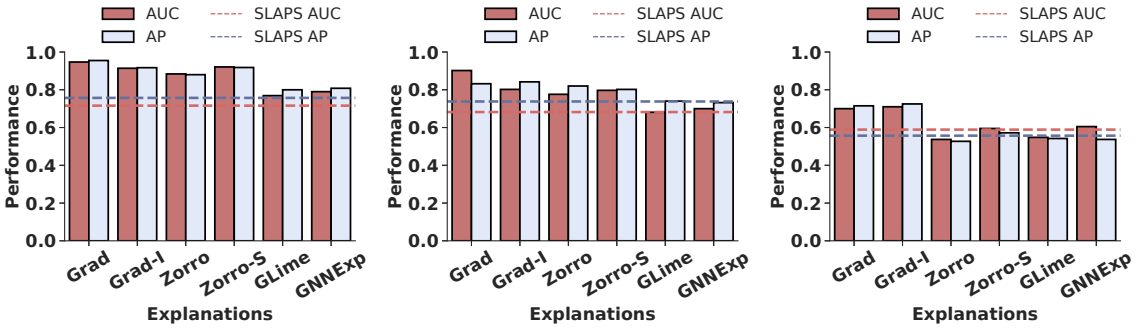
where f is a 2-layer GCN model parameterized by θ_W, $\mathcal{E}_X$ is the explanation matrix, Adj_{rec} is the reconstructed graph by the attacker, y is the groundtruth label and $\mathcal{R}$ is an advantage measure that compares the predictions on f and the groundtruth label. We use accuracy as the choice of $\mathcal{R}$.

We compare the results to that of Slaps that uses the actual features and groundtruth label for learning the graph structure and the original performance (denoted by *Max*) in which the model is trained with true features, labels, and graph structure. We analyze the attacker's advantage corresponding to four attacks;
GSEF-concat, GSEF-mult, GSEF, and GSE. The intuition is that if the attacker's advantage is not better than Slaps, then the best advantage an attacker can have is similar to having the actual feature and performing graph structure learning. Also, if the attacker's advantage is greater or equal to *Max*, then the attacker has an equivalent advantage as she would have by possessing the actual feature and graph. An example use case of the attacker's advantage is shown in Figure 7. Specifically, if a model trained with, say, Jane's full data (true features and graph) and another trained only with her explanations (no graph or true features), both models will make the same prediction about Jane. The detailed results for the attacker's advantage are plotted in Figure 8. We observe that on Cora, the attacker obtains the highest advantage for Grad, Grad-I, Zorro, and Zorro-S explanations. On CoraML, the highest advantage is obtained with Grad and Zorro-S explanations. Usually, the attacker's advantage is positively correlated with the success rate of corresponding attacks for both Cora and CoraML. On Bitcoin, the attacker's advantage for all explanation methods is usually high. This is surprising as the success rate for attacks is relatively lower than for other datasets. This might imply that the reconstructed



(a) Cora (b) CoraML (c) Bitcoin

Figure 5: Average AUC and AP of GSE attack on the different datasets. The adopted baseline is Slaps which use the true node features.



(a) Cora (b) CoraML (c) Bitcoin

Figure 6: Average AUC and AP of GSEF attack on the different datasets. The adopted baseline is Slaps which use the true node features.

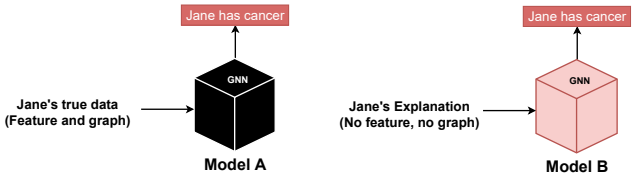


Figure 7: An example use case of the attacker's advantage

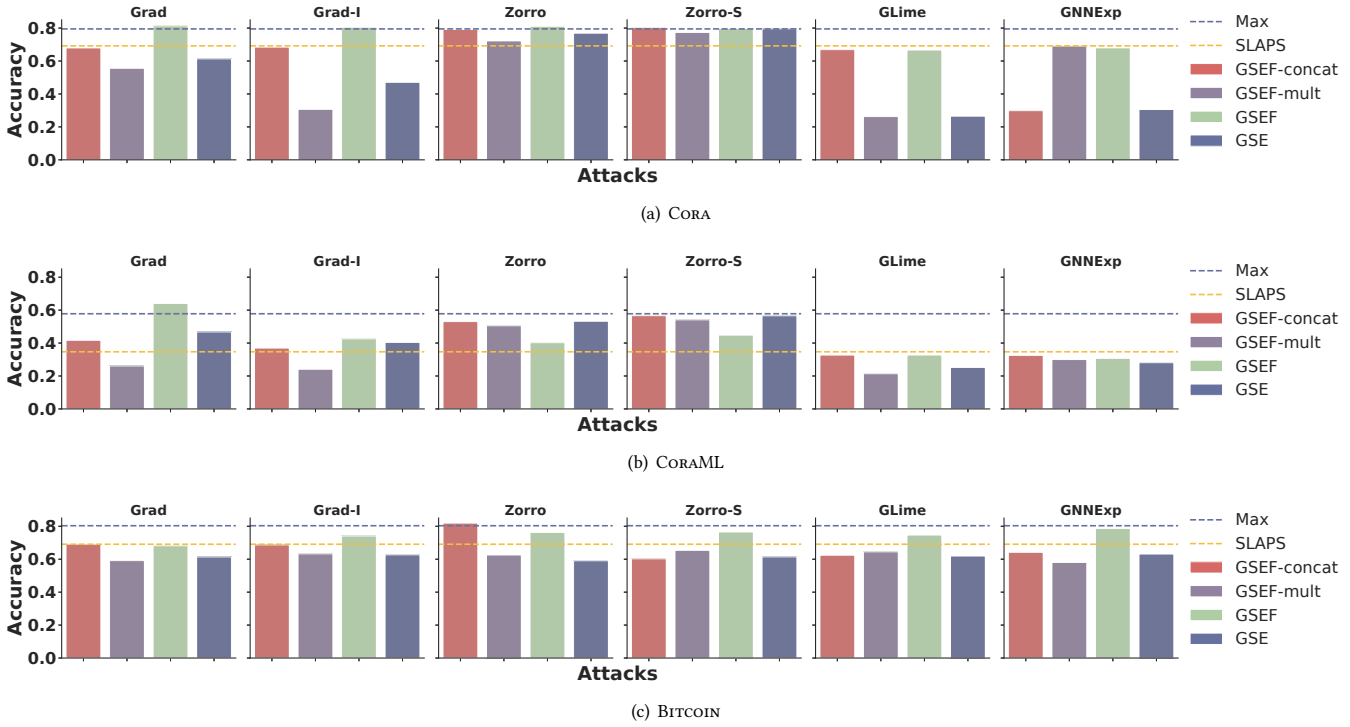
graph has the same semantics as the true graph, if not the exact structure.

4.2.5 ***Lack of Groundtruth Labels (RQ 4).*** We relax the assumption that the attacker has access to groundtruth labels. Instead, she has black-box access to the target model. This is made possible with the popularity of machine learning as a service (MLaaS), where a user can input a query and get the predictions as output. Therefore, the "groundtruth" label is the one obtained from the target model. As representative explanations, we show the performance on Grad and Zorro on all datasets.

As shown in Table 5, on the Grad explanation, we observe a 3% gain in attack performance on GSEF-concat when the attacker has access to the target model on the Cora and CoraML dataset. On the Bitcoin dataset, we observe a decrease of about 3% in AUC and an increase of 6% in AP. The corresponding performance on Zorro follows the same with no significant change in the performance on CoraML and a 5% decrease in performance on Cora.

The performance of GSEF-mult and GSEF attack decreases across all datasets, with Bitcoin having the worst performance reduction of up to 29% on Grad. However, on Zorro, there is a 2% gain on Cora, a 4% decrease on CoraML, and up to 36% decrease on Bitcoin. We observe performance drop on GSEF and GSE on all datasets across both explanations except for GSE on Grad, which has up to 5% gain in performance on the Cora and CoraML datasets. It is important to note that the Bitcoin dataset has the least performance across all attacks, even when the true groundtruth label is used. Therefore, the large disparity in performance when the label is generated from the trained black-box model is not surprising.

Summary. In the absence of groundtruth labels, we used model predictions. Intuitively, for incorrectly predicted nodes, the model explanations which are based on incorrectly predicted labels should



(a) Cora

(b) CoraML

(c) Bitcoin

Figure 8: Accuracy of the reconstructed graph on a downstream node classification task by all models on different datasets. The blue line is the original accuracy using the true features and edges, while the yellow line is the Slaps accuracy.

lead to lower performance in the presence of groundtruth labels. For Cora and CoraML datasets, on GSEF-concat and GSE attacks, having black-box access to the target model performs better than the attacker having access to the groundtruth label. For GSEF-mult and GSEF attacks, it is better to have access to groundtruth label to achieve the best attack success rate. For the Bitcoin dataset, the groundtruth labels perform better than the black-box access to the target model on all attacks.

4.3 Attack Performance When Explanations for Partial Node-set Are Available

In this section, we assume that the attacker has access to explanations of only a subset of the nodes. The attacker is interested in reconstructing the subgraph induced on the partial node set. We perform the experiment on representative datasets Cora and Credit. For each of the datasets, we only use 30% of the nodes to obtain explanations. The result is shown in Table 6.

Unsurprisingly, we observe a drop in attack performance scores when explanations for only a node subset are available. However, we observe that explanation augmentation attacks perform best on all explainers and datasets. Recall that when the full explanations are available on Grad, the augmentation attacks rank second on the Cora dataset. However, with explanations for the partial node set, augmentation attacks rank first with AUROC and AP of 0.91 and 0.92. Moreover, on the Cora and Credit datasets, GSEF-concat and GSEF-mult perform competitively with other attacks. Lastly, as observed in the experiments, when the full explanations are available, attacks on the Grad and Zorro-S explanations are highly successful, while attacks on the GNNExp are less successful. Nonetheless, we observe AUC and AP > 0.76 on both datasets for GNNExp.

4.4 When Do Augmentation Attacks Work?

To summarize, augmentation attacks perform best on all explanation models except on Grad and Grad-I across all datasets. Nonetheless, augmentation attack GSEF on the Grad and Grad-I explanations perform comparable to or better than GSE attack, which only uses explanations as input on CiteSeer, CoraML, Bitcoin, and Credit datasets.

On explainers other than Grad and Grad-I, augmentation attacks perform the best. For instance, on the CiteSeer dataset, augmentation attack GSEF achieved AUROC and AP of 0.94 and 0.95 respectively on Zorro-S explanation, which is far better than all baselines and explanation-only attacks. The multi-task learning paradigm that GSEF employs gives it an edge over other augmentation attacks, baselines, and explanation-only attacks.

Overall, the strength of augmentation attacks depends on the choice of the explanation method and the strategy of using the additional feature information. For instance, GSEF-concat and GSEF-mult augmentation attacks, which combine the additional feature information by concatenation and element-wise multiplication, perform worst on the Grad, Grad-I, Zorro, and Zorro-S explanations. However, they perform best on surrogate method

Table 5: Performance comparison of relaxing the availability of groundtruth labels (Y) assumption. Here, the attacker has black-box access to the target model ($\mathcal{M}$). We perform the experiment on all datasets' Grad and Zorro explanation methods. Δ is the percentage difference. A negative value implies that the groudtruth labels are preferred over black-box access.

Dataset	Attack	Y_{Grad}		$\mathcal{M}_{Grad}$		Δ_{Grad}		Y_{Zorro}		$\mathcal{M}_{Zorro}$		Δ_{Zorro}	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Cora	GSEF-concat	0.734	0.773	0.757	0.784	3.1	1.4	0.823	0.860	0.779	0.810	-5.3	-5.8
	GSEF-mult	0.678	0.737	0.658	0.700	-2.9	-5.0	0.723	0.756	0.740	0.772	2.4	2.0
	GSEF	0.948	0.953	0.927	0.932	-2.2	-2.2	0.884	0.880	0.871	0.881	-1.5	0.1
	GSE	0.924	0.939	0.946	0.966	2.4	2.9	0.779	0.810	0.814	0.849	4.5	4.8
CoraML	GSEF-concat	0.640	0.705	0.661	0.734	3.3	4.1	0.735	0.786	0.738	0.792	0.4	0.8
	GSEF-mult	0.666	0.730	0.650	0.693	-2.4	-5.1	0.681	0.697	0.653	0.692	-4.1	-0.7
	GSEF	0.902	0.833	0.808	0.852	-10.4	2.3	0.776	0.820	0.751	0.796	-3.2	-2.9
	GSE	0.699	0.768	0.735	0.795	5.2	3.5	0.722	0.777	0.713	0.759	-1.2	-2.3
Bitcoin	GSEF-concat	0.527	0.515	0.512	0.546	-2.7	6.0	0.575	0.529	0.523	0.517	-9.0	-2.3
	GSEF-mult	0.264	0.383	0.241	0.367	-8.7	-4.2	0.399	0.449	0.255	0.369	-36.1	-17.8
	GSEF	0.700	0.715	0.500	0.560	-28.6	-21.7	0.537	0.527	0.352	0.438	-34.5	-16.9
	GSE	0.229	0.365	0.210	0.352	-8.3	-3.6	0.596	0.561	0.491	0.503	-17.6	-10.3

Table 6: Attack performance with explanations over partial node set for Cora and Credit datasets. The best performing attack(s) on each explanation method is(are) highlighted in bold, and the second best attack(s) is(are) underlined.

Exp	Attack	Cora		Credit	
		AUC	AP	AUC	AP
Grad	GSEF-concat	0.683 ± 0.02	0.700 ± 0.03	0.821 ± 0.03	0.851 ± 0.03
	GSEF-mult	0.674 ± 0.02	0.705 ± 0.02	0.806 ± 0.03	0.839 ± 0.02
	GSEF	**0.916 ± 0.03**	**0.920 ± 0.03**	**0.842 ± 0.04**	**0.875 ± 0.04**
	GSE	0.865 ± 0.02	0.871 ± 0.02	0.840 ± 0.02	0.870 ± 0.03
	ExplainSim	0.881 ± 0.01	0.913 ± 0.01	0.835 ± 0.01	0.854 ± 0.01
Zorro-S	GSEF-concat	0.863 ± 0.04	0.896 ± 0.04	0.825 ± 0.03	0.842 ± 0.02
	GSEF-mult	0.720 ± 0.03	0.757 ± 0.04	0.819 ± 0.02	0.840 ± 0.02
	GSEF	**0.901 ± 0.03**	**0.905 ± 0.03**	**0.835 ± 0.02**	**0.847 ± 0.03**
	GSE	0.881 ± 0.04	0.893 ± 0.01	0.762 ± 0.03	0.801 ± 0.02
	ExplainSim	0.887 ± 0.02	0.901 ± 0.02	0.701 ± 0.01	0.721 ± 0.01
GNNExp	GSEF-concat	0.585 ± 0.04	0.600 ± 0.04	**0.829 ± 0.03**	**0.840 ± 0.02**
	GSEF-mult	0.680 ± 0.03	0.699 ± 0.03	0.794 ± 0.02	0.802 ± 0.02
	GSEF	**0.762 ± 0.04**	**0.791 ± 0.06**	0.782 ± 0.02	0.799 ± 0.02
	GSE	0.500 ± 0.06	0.520 ± 0.05	0.688 ± 0.04	0.715 ± 0.04
	ExplainSim	0.508 ± 0.02	0.515 ± 0.02	0.615 ± 0.02	0.660 ± 0.02

GLime and perturbation-based method GNNExp. Moreover, augmentation attacks show the best performance when explanations only for a partial node set are available.

5 DEFENSE

Explanation Perturbation. To limit the information leakage by the explanation, we perturb each explanation bit using a randomized response mechanism [22, 40]. Specifically for 0/1 feature (explanation) mask as in Zorro, we flip each bit of the explanation with a probability that depends on the privacy budget ϵ as follows

$$Pr(\mathcal{E}'_{x_i} = 1) = \begin{cases} \frac{e^{\epsilon}}{e^{\epsilon}+1}, & \text{if } \mathcal{E}_{x_i} = 1, \\ \frac{1}{e^{\epsilon}+1}, & \text{if } \mathcal{E}_{x_i} = 0, \end{cases} \tag{7}$$

where $\mathcal{E}_{x_i}$ and $\mathcal{E}'_{x_i}$ are true and perturbed i^{th} bit of explanation $\mathcal{E}_x$ respectively. Note that our defense mechanism satisfies $d\epsilon$-local differential privacy.

Lemma 1. *For an explanation with d dimensions, the explanation perturbation defense mechanism in Equation 7 satisfies $d\epsilon$-local differential privacy.*

Proof. Note that for an explanation corresponding to two graph datasets D and D' differing in a single edge, the ratio of probabilities of obtaining a certain explanation can be bounded as follows.

$$\frac{Pr[\mathcal{E}_X(D) = S]}{Pr[\mathcal{E}_X(D') = S]} = \prod_{i=1}^{d} \frac{Pr[\mathcal{E}_{x_i}(D) = S_i]}{Pr[\mathcal{E}_{x_i}(D') = S_i]} \leq \prod_{i=1}^{d} \frac{\frac{e^\epsilon}{e^\epsilon+1}}{\frac{1}{e^\epsilon+1}} = e^{d\epsilon}.$$

□

Defense Evaluation. We evaluate our defense mechanism on ExplainSim attack as it best quantifies the information leakage due to the explanations alone. All other attacks assume the availability of other information, such as features and labels. We use two datasets: Cora and CoraML. As evaluation metrics, we use the AUC score and AP to compute the attack success rate after the defense. Besides, we measure the utility of the perturbed explanation in terms of fidelity, sparsity, and the percentage of 1 bit that is retained from the original explanation (intersection).

Defense Results. As shown in Figure 9, the explanation perturbation based on the randomized response mechanism clearly defends against the attack. For instance, at a very high privacy level $\epsilon = 0.0001$, which gives $d\epsilon = 0.14$, the attack performance drastically dropped to 0.56 in AUC and 0.59 in AP, which is about 36% decrease over the non-private released explanation. As expected, the attack performance decreases significantly with increase in the amount of noise (ϵ decreases). In Table 7, we analyze the change in explanation utility due to our perturbation mechanism. We observe that on Cora, with the lowest privacy loss level, there is a drop of 5.61% in the fidelity when the attack is already reduced to a random guess. The entropy of the mask distribution increases. In other words, the explanation sparsity decreases. For Zorro, this implies that more bits are set to 1 than in the true explanation mask. Even though this decreases explanation utility to some extent, we point out that 74.68% of true explanation is still retained. Moreover, the sparsity is still lower than achieved by GNNExp explanations, even without any perturbations.

While quantitatively, the change in explanation sparsity is acceptable, more application-dependent qualitative studies are required to evaluate the change in the utility of explanations. Nevertheless, we provide a promising first defense for future development and possible improvements. We obtain similar results for CoraML, which are provided in Appendix C.

Defense Variant for Soft Explanation Masks. Note that Equation 7 only applies to explanations that return binary values. For explanations with continuous values, we can adapt the defense as follows. We keep the original value ($\mathcal{E}_{\mathbf{x}i}$) when the flipped coin lands heads, but when it lands tail, we replace ($\mathcal{E}_{\mathbf{x}i}$) with $\mathcal{E}_{x_i'}$ where $\mathcal{E}_{x_i'}$ is a random number drawn from a normal distribution ($\mathcal{E}_{\mathbf{x}_i'} \sim \mathcal{N}(0, 1)$).

6 CONCLUSION

We initiate the first investigation on the privacy risks of releasing post-hoc explanations of graph neural networks. Concretely, we quantify the information leakage of explanations via our proposed *five* graph reconstruction attacks. The goal of the attacker is to reconstruct the private graph structure information used to train a GNN model. Our results show that even when the explanations alone are available without any additional auxiliary information, the attacker can reconstruct the graph structure with an AUC score of more than 90%. Our explanation-based attacks outperform all baseline methods pointing to the additional privacy risk of releasing explanations. We propose a perturbation-based defense mechanism that reduces the attack to a random guess. The defense leads to a slight decrease in fidelity. At the lowest privacy loss, the perturbed explanation still contains around 75% of the true explanation. While quantitatively, the change in explanation sparsity seems to be acceptable, more application-dependent qualitative studies would be required to evaluate the change in the utility of explanations.

We emphasize that we strongly believe in the transparency of graph machine learning and acknowledge the need to explain trained models. At the same time, our work points out the associated privacy risks which cannot be ignored. We believe that our work would encourage future work on finding solutions to balance the complex trade-off between privacy and transparency.

Table 7: Fidelity, sparsity and percentage of 1 bits in the true explanation that is retained in the perturbed explanation (intersection) after defense for different ϵ on the Cora dataset for Zorro explanation. ∞ implies no privacy.

ϵ	Fidelity	Sparsity	Intersection
0.0001	0.84	5.91	74.68
0.001	0.84	5.91	74.70
0.01	0.84	5.89	75.03
0.1	0.84	5.80	75.10
0.2	0.83	5.71	75.60
0.4	0.82	5.49	76.45
0.6	0.81	5.25	77.16
0.8	0.81	5.00	78.66
1	0.81	4.73	80.10
∞	0.89	1.83	100

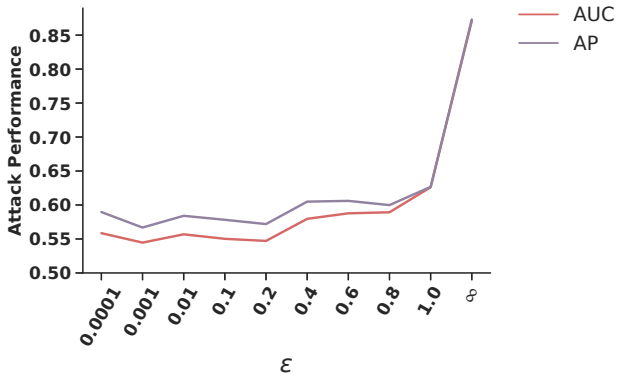


Figure 9: Privacy budget and corresponding attack performance of ExplainSim for Zorro explanation on the Cora dataset. ∞ implies that no perturbation is performed.

ACKNOWLEDGMENTS

This work is partly funded by the Lower Saxony Ministry of Science and Culture under grant number ZN3491 within the Lower Saxony "Vorab" of the Volkswagen Foundation and supported by the Center for Digital Innovations (ZDIN), and the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor (grant number 01DD20003). The authors are grateful to the anonymous reviewers for providing insights that further improved our paper.

REFERENCES

[1] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. 2021. Towards a unified framework for fair and stable graph representation learning. In *Uncertainty in Artificial Intelligence*. PMLR, 2114–2124.

[2] David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. 2021. Graph-Based Deep Learning for Medical Diagnosis and Analysis: Past, Present and Future. *Sensors (Basel, Switzerland)* 21, 14 (July 2021). https://doi.org/10.3390/s21144758

[3] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).

[4] Anitesh Barua, Suryanarayanan Ravindran, and Andrew B Whinston. 2007. Enabling information sharing within organizations. *Information Technology and Management* 8, 1 (2007), 31–45.

[5] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371* (2018).

[6] Ngan Thi Dong and Megha Khosla. 2020. Revisiting Feature Selection with Data Complexity. In *2020 IEEE 20th International Conference on Bioinformatics and Bioengineering (BIBE)*. 211–216. https://doi.org/10.1109/BIBE50027.2020.00042

[7] Thi Ngan Dong, Stefanie Mucke, and Megha Khosla. 2022. MuCoMiD: A Multitask graph Convolutional Learning Framework for miRNA-Disease Association Prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2022).

[8] Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. 2018. Towards explanation of dnn-based prediction with guided feature inversion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1358–1367.

[9] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying Privacy Leakage in Graph Embedding. *arXiv preprint arXiv:2010.00906* (2020).

[10] Aleksander Fabijan, Helena Holmström Olsson, and Jan Bosch. 2016. The Lack of Sharing of Customer Data in Large Software Organizations: Challenges and Implications. In *Agile Processes, in Software Engineering, and Extreme Programming*, Helen Sharp and Tracy Hall (Eds.). Springer International Publishing, Cham, 39–52.

[11] Wenqi Fan, Wei Jin, Xiaorui Liu, Han Xu, Xianfeng Tang, Suhang Wang, Qing Li, Jiliang Tang, Jianping Wang, and Charu Aggarwal. 2021. Jointly Attacking Graph Neural Network and its Explanations. *arXiv preprint arXiv:2108.03388* (2021).

[12] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[13] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. *Advances in Neural Information Processing Systems* 34 (2021).

[14] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[15] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*. PMLR, 1972–1982.

[16] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. 2022. ZORRO: Valid, Sparse, and Stable Explanations in Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[17] Bryce Goodman and Seth Flaxman. 2017. European Union regulations on algorithmic decision-making and a "right to explanation". *AI magazine* 38, 3 (2017), 50–57.

[18] Danielle Greenstein. 2020. Why you should share data with other in your organization. https://www.lotame.com/sharing-data-within-organization/

[19] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[20] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing links from graph neural networks. In *30th USENIX Security Symposium (USENIX Security 21)*. 2669–2686.

[21] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. Graphlime: Local interpretable model explanations for graph neural networks. *arXiv:2001.06216* (2020).

[22] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*. PMLR, 2436–2444.

[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[24] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 221–230.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[26] Iyiola E Olatunji, Thorben Funke, and Megha Khosla. 2021. Releasing Graph Neural Networks with Differential Privacy Guarantees. *arXiv preprint arXiv:2109.08907* (2021).

[27] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership Inference Attack on Graph Neural Networks. In *IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2021*.

[28] European Parliament and Council of the European Union. 2016. General Data Protection Regulation. *Official Journal of the European Union* (2016). https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32016R0679

[29] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. 2019. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10772–10781.

[30] Sina Sajadmanesh and Daniel Gatica-Perez. 2020. Locally Private Graph Neural Networks. *arXiv preprint arXiv:2006.05535* (2020).

[31] Andrew Selbst and Julia Powles. 2018. "Meaningful Information" and the Right to Explanation. In *Conference on Fairness, Accountability and Transparency*. PMLR, 48–48.

[32] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[34] Reza Shokri, Martin Strobel, and Yair Zick. 2021. On the privacy risks of model explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 231–241.

[35] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034* (2013).

[36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *PMLR*.

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[38] Minh Vu and My T Thai. 2020. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *Advances in neural information processing systems* 33 (2020), 12225–12235.

[39] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2023–2040.

[40] Yue Wang, Xintao Wu, and Donghui Hu. 2016. Using Randomized Response for Differential Privacy Preserving Data Collection.. In *EDBT/ICDT Workshops*, Vol. 1558. 0090–6778.

[41] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. 2020. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization. *arXiv preprint arXiv:2010.12751* (2020).

[42] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2021. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis. *arXiv preprint arXiv:2108.06504* (2021).

[43] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610* (2019).

[44] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. 2014. High-dimensional feature selection by feature-wise kernelized lasso. *Neural computation* 26, 1 (2014), 185–207.

[45] Tung-Mou Yang and Terrence A Maxwell. 2011. Information-sharing in public organizations: A literature review of interpersonal, intra-organizational and inter-organizational success factors. *Government information quarterly* 28, 2 (2011), 164–175.

[46] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNN Explainer: A tool for post-hoc explanation of graph neural networks. *arXiv:1903.03894* (2019).

[47] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2020. Backdoor attacks to graph neural networks. *arXiv preprint arXiv:2006.11165* (2020).
[48] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. 2021. GraphMI: Extracting Private Graph Data from Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). 3749–3755.
[49] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2847–2856.
[50] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412* (2019).

APPENDIX

Organization. The Appendix is organized as follows. We first present related works in private graph extraction attacks and other attacks on GNN models. Next, we elaborated on the data sampling used for evaluation in Section B followed by the results for defense on the CoraML dataset in Section C. The performance and the corresponding standard deviation of our main results are provided in Section D. In Section E, we present the results of the additional experiments on the CiteSeer, PubMed, and Credit datasets. Finally, we list the hyperparameters used for our attacks and target model in Section F.

A RELATED WORKS

Private Graph Extraction Attacks. Given a black-box access to a GNN model that is trained on a target dataset and the adversary's background knowledge, He et al. [20] proposed link stealing attacks to infer whether there is a link between a given pair of nodes in the target dataset. Their attacks are specific to the adversary's background knowledge which range from simply exploiting the node feature similarities to a shadow model-based attack. Wu et al. [42] proposed an edge re-identification attack for vertically partitioned graph learning. Their attack setting is different from ours and is applicable in scenarios where the high-dimensional features and high-order adjacency information are usually heterogeneous and held by different data holders. GraphMI [48] aims at reconstructing the adjacency matrix of the target graph given white-box access to the trained model, node features, and labels.

Other Inference Attacks and Defenses on GNNs. Several other attacks, such as membership inference [9, 27] and model extraction attacks [41] have been proposed to quantify privacy leakage in GNNs. In a membership inference attack, the goal of the attacker is to infer whether a node was part of the data used in training the GNN model via black-box access to the trained model. In a model extraction attack, the attacker aims to steal the trained model's parameter and hyperparameters to duplicate or mimic the functionality of the target model via the predictions returned from querying the model[41]. Recently, several defenses against these attacks have been proposed, which are mainly based on differential privacy. Olatunji et al. [26] proposed a method for releasing GNN models by combining a knowledge-distillation framework with two noise mechanisms, random subsampling and noisy labeling. Their centralized setting approach trains a student model using a public graph and private labels obtained from a teacher model trained exclusively for each query node (personalized teacher models). Their method, by design, defends against membership inference attacks and model extraction attacks since only the student model (which has limited and perturbed information) is released. Sajadmanesh and Gatica-Perez [30] proposed a locally differentially private GNN model by considering a distributed setting where nodes and labels are private, and the graph structure is known to the central server. Their approach perturbs both the node features and labels to ensure a differential privacy guarantee. However, all attacks and defenses are not applicable to explanations.

Membership Inference Attack and Explanations. On euclidean data such as images, Shokri et al. [34] analyzed the privacy risks of feature-based model explanations using membership inference attacks which quantifies the extent to which model predictions and their explanations leak information about the presence of a datapoint in the training set of a model. We emphasize that the goal of Shokri et al. [34] differs from ours in that we focus on reconstructing the entire graph structure from feature-based explanations. Also, their investigations are limited to non-graph data and the corresponding target and explanation models.

Adversarial Attacks and GNNs. Another line of research focuses on the vulnerability of GNNs to adversarial attacks [5, 39, 43, 47, 49, 50]. The goal of the attacker is to fool the GNN model into making a wrong prediction by manipulating node features or the structural information of nodes. A recent work [11] used explanation method such as GNNExplainer as a method for detecting adversarial perturbation on graphs. Hence, acting as a tool for inspecting adversarial attacks on GNN models. They further proposed an adversarial attack framework (GEAttack) that exploits the vulnerabilities of explanation methods and the GNN model. This allows the attacker to simultaneously fool the GNN model and misguide the inspection from the explanation method. Our work differs significantly from this work in that first, we aim to reconstruct the graph from the explanations and, secondly, to quantify the privacy leakage of explanations on GNN models.

B DATA SAMPLING FOR EVALUATION

In this section, we elaborate on the data sampling method used for our evaluation as briefly explained in Section 4.1.3. To generate a test set, we randomly select 10% of all nodes. We use the set of all edges which are incident on at least one of the nodes in our selected set. We then sample an equal number of negative edges (pairs of nodes such that no edge exists between them; one of the nodes in this pair is from the selected set of 10% nodes). In total, we generate 10 random test sets.

We employ 10 random instantiations of the attack models. Corresponding to each instantiation of the attack model, we test the model with one of our random test sets. In total, we have 10 observations of the final scores. We evaluated all methods on the same test sets. As we wanted to keep the number of observations the same for all compared methods, including the non-parameterized methods based on feature and explanation similarity (note that they do not use a neural model, and there is no initialization required in their cases), we tested each attack model (corresponding to each initialization) on one test set. We computed the average precision (AP) and AUROC result on the balanced pairs. For the partial explanation experiment, we randomly sample 30% of the nodes only once to generate a fixed subgraph. Evaluation is performed as in

the previous case. Here, the 10 test sets are generated from this subgraph.

We remark that using AUC and AP scores allows us to evaluate all methods across all ranges of cutoff decision thresholds. Our evaluation is domain and dataset-independent. Nevertheless, in real scenarios, an attacker might need to choose a single decision threshold for which domain-specific knowledge or an auxiliary validation dataset might be required.

C RESULTS FOR DEFENSE ON CORAML

The attack performance for different values of ϵ is plotted in Figure 10. For the lowest privacy budget, we observe that the attack is reduced to a random guess (with an AUC score close to 0.55). The variation in explanation utility and intersection with true explanation is shown in Table 8. Here also, the perturbed explanation is able to retain around 75% of the 1 bit of the true explanation. While there is a drop in fidelity and the explanation becomes denser, we note that the perturbed explanation still shows higher fidelity and sparsity than other explanation methods (c.f. Table 4).

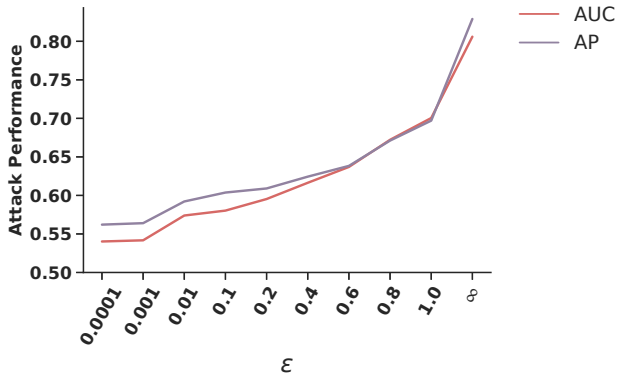


Figure 10: Privacy budget and corresponding attack performance of ExplainSim for Zorro explanation on the CoraML dataset. ∞ implies no privacy.

D DETAILED RESULTS

In Table 9, we present the mean and standard deviation of the results in Table 3. Since our results have a low standard deviation across the 10 runs of the experiment, our method is stable.

E ADDITIONAL EXPERIMENTS

We perform additional experiments on 3 additional datasets to verify our observations and the effectiveness of our attacks. We start by describing the additional datasets. The performance scores are provided in Table 12, and results are discussed in the main paper.

E.1 Additional Datasets

We used CiteSeer and PubMed, which are conventional datasets for node classification tasks and the Credit dataset. The details about the datasets are in Table 10. The fidelity and sparsity for the corresponding explanations are provided in Table 11.

Table 8: Fidelity, sparsity and percentage of 1 bits in the true explanation that is retained in the perturbed explanation (intersection) after defense for different ϵ on the CoraML dataset for Zorro explanation. ∞ implies no privacy.

ϵ	Fidelity	Sparsity	Intersection
0.0001	0.86	4.53	74.96
0.001	0.86	4.53	74.98
0.01	0.86	4.53	75.12
0.1	0.87	4.46	75.08
0.2	0.87	4.39	75.20
0.4	0.87	4.24	75.70
0.6	0.87	4.08	77.07
0.8	0.88	3.95	78.70
1	0.89	3.81	80.75
∞	0.96	3.33	100

PubMed and CiteSeer. Both PubMed and CiteSeer [33] are citation datasets, where each node is a research article, and there is an edge between two articles if one cites the other. In CiteSeer, features are binary values like Cora, while in PubMed, the features are represented by the TF/IDF weighted word vector of the unique words in the dictionary.

Credit. The Credit defaulter graph [1] is a financial network where an edge exists between individuals (nodes) if there is a similarity in their spending and payment pattern. The task is to determine whether an individual will default on their credit card payment. We used a subset of this credit dataset which has 3000 nodes.

E.2 Ensuring the Stability of Results

To ensure the stability of the results, we perform an additional experiment by slightly modifying the evaluation procedure in Section B (hereafter referred to as *10-runs*). Instead of testing the attacks on one of our random test sets for each instantiation which resulted in 10 observations of the final scores, we modify the evaluation as follows. For each instantiation, we evaluate the attacks on *all* 10 held out balanced test sets. This results in 100 observations of the final score (recall that we have 10 instantiations). We report the mean and the standard deviation. This modification which we call *100-runs*, further mitigates any sampling biases from the test sets and any randomness. Similar to 10-runs, we evaluated all methods on the same test sets. The ExplainSim attack is excluded since there is no randomness involved, resulting in the same results as in 10-runs.

We observe that in most cases, the difference between all previous experiments, which were evaluated using the 10-runs procedure and the modified 100-runs is 0. This shows that our experiments are stable and the results are reliable. Although we observe some minor variations in some results, however, such variations are relatively small and do not invalidate our observations or experiments. For example, on the PubMed dataset, GSE attack on the Grad explanation for the 10-runs resulted in an AUC of 0.770 while that of 100-runs is 0.767. The results of the 100-runs are in Table 13.

Table 9: Results with standard deviation of Table 3. The best performing attack(s) on each explanation method is(are) highlighted in bold, and the second best attack(s) is(are) underlined.

Exp	Attack	Cora		CoraML		Bitcoin	
		AUC	AP	AUC	AP	AUC	AP
Baseline	FeatureSim	0.799±0.04	0.827±0.04	0.706±0.08	0.753±0.07	0.535±0.03	0.478±0.02
	Lsa [20]	0.795±0.03	0.810±0.02	0.725±0.04	0.760±0.01	0.532±0.05	0.500±0.06
	GraphMI [48]	0.856±0.01	0.830±0.01	0.808±0.02	0.814±0.03	0.585±0.05	0.518±0.06
	Slaps [13]	0.736±0.05	0.776±0.05	0.649±0.06	0.702±0.07	0.597±0.09	0.577±0.07
Grad	GSEF-concat	0.734±0.05	0.773±0.04	0.640±0.05	0.705±0.04	0.527±0.04	0.515±0.03
	GSEF-mult	0.678±0.04	0.737±0.03	0.666±0.06	0.730±0.05	0.264±0.07	0.383±0.04
	GSEF	0.948 ± 0.02	0.953 ± 0.01	**0.902 ± 0.08**	0.833 ± 0.07	**0.700 ± 0.05**	**0.715 ± 0.04**
	GSE	0.924±0.03	0.939±0.02	0.699±0.07	0.768±0.05	0.229±0.03	0.365±0.02
	ExplainSim	**0.984 ± 0.01**	**0.978 ± 0.01**	0.890 ± 0.04	**0.891 ± 0.04**	0.681 ± 0.03	0.644 ± 0.03
Grad-I	GSEF-concat	0.734±0.06	0.775±0.04	0.674±0.05	0.724±0.04	0.525±0.09	0.527±0.05
	GSEF-mult	0.691±0.02	0.742±0.02	0.717±0.05	0.756±0.06	0.252±0.03	0.380±0.02
	GSEF	0.949 ± 0.02	0.950 ± 0.02	0.787 ± 0.08	0.832 ± 0.07	**0.709 ± 0.04**	**0.723 ± 0.03**
	GSE	0.903±0.04	0.923±0.04	0.717±0.08	0.781±0.06	0.256±0.03	0.380±0.02
	ExplainSim	**0.984 ± 0.01**	**0.979 ± 0.01**	**0.903 ± 0.04**	**0.899 ± 0.04**	0.681 ± 0.03	0.644 ± 0.03
Zorro	GSEF-concat	0.823±0.04	0.860±0.05	0.735±0.02	0.786±0.01	0.575 ± 0.03	0.529±0.05
	GSEF-mult	0.723±0.03	0.756±0.03	0.681±0.02	0.697±0.04	0.399±0.07	0.449±0.05
	GSEF	**0.884 ± 0.03**	**0.880 ± 0.04**	0.776 ± 0.03	0.820 ± 0.02	0.537±0.05	0.527 ± 0.04
	GSE	0.779±0.04	0.810±0.01	0.722±0.02	0.777±0.02	**0.596 ± 0.03**	**0.561 ± 0.03**
	ExplainSim	0.871 ± 0.02	0.873 ± 0.02	**0.806 ± 0.02**	**0.829 ± 0.03**	0.427±0.06	0.485±0.05
Zorro-S	GSEF-concat	0.907±0.03	0.922±0.02	0.747 ± 0.06	0.791 ± 0.05	**0.601 ± 0.06**	**0.590 ± 0.05**
	GSEF-mult	0.794±0.06	0.815±0.06	0.712±0.06	0.740±0.06	0.490±0.08	0.491±0.05
	GSEF	**0.918 ± 0.02**	0.923 ± 0.02	**0.776 ± 0.06**	0.819±0.05	0.598 ± 0.03	0.565 ± 0.03
	GSE	0.893±0.04	0.915±0.02	0.742±0.06	**0.784 ± 0.05**	0.571±0.03	0.564±0.04
	ExplainSim	0.908 ± 0.03	**0.934 ± 0.02**	0.732±0.05	0.787±0.03	0.484±0.04	0.496±0.03
GLime	GSEF-concat	0.643 ± 0.05	0.710 ± 0.04	0.610 ± 0.05	0.652 ± 0.04	0.473 ± 0.07	0.492 ± 0.05
	GSEF-mult	0.516±0.06	0.522±0.04	0.517±0.05	0.528±0.04	0.264±0.03	0.371±0.01
	GSEF	**0.730 ± 0.05**	**0.774 ± 0.03**	**0.681 ± 0.05**	**0.740 ± 0.05**	**0.542 ± 0.05**	**0.525 ± 0.03**
	GSE	0.558±0.06	0.571±0.05	0.540±0.06	0.555±0.05	0.236±0.04	0.361±0.02
	ExplainSim	0.505±0.04	0.524±0.04	0.520±0.04	0.523±0.04	0.504±0.05	0.512±0.03
GNNExp	GSEF-concat	0.614±0.04	0.650±0.04	0.653±0.05	0.705±0.04	0.467±0.11	0.489±0.06
	GSEF-mult	0.724 ± 0.05	0.760 ± 0.05	0.637 ± 0.05	0.692 ± 0.05	0.390±0.10	0.454±0.05
	GSEF	**0.762 ± 0.04**	**0.796 ± 0.03**	**0.700 ± 0.07**	**0.796 ± 0.06**	**0.590 ± 0.04**	**0.563 ± 0.03**
	GSE	0.517±0.04	0.552±0.03	0.490±0.05	0.508±0.04	0.386±0.11	0.451±0.07
	ExplainSim	0.537±0.05	0.541±0.04	0.484±0.05	0.508±0.04	0.551 ± 0.04	0.545 ± 0.03

F HYPERPARAMETER SETTINGS

In Table 14, we provide the hyperparameters used in training the different modules of our attack for reproducibility. We used hyperparameters from [13], which are already fine-tuned for these datasets.

G LIMITATIONS AND FUTURE DIRECTION

This section highlights some limitations and future directions for our work.

Partial Explanations. In the current work, the 30% nodes we sampled to generate the fixed subgraph for the partial explanation experiment were only sampled once. Such a sampling approach might be subject to sampling bias. Hence, sampling the subgraph multiple times may be desirable to avoid such bias.

More Work on Attacks and Defenses. A promising direction will be to launch more attacks and defenses on other explanation outputs, such as nodes or subgraph-level explanations. These are natural derivatives of graph-based explanation methods, and it is worth investigating to quantify the risk involved in releasing explanations. More defenses that optimize the usefulness of the explanations (fidelity and sparsity) are desirable.

Fidelity-sparsity Analysis. In this work, we analyzed the effect of the usefulness of explanations as measured by their fidelity and

Table 10: Dataset statistics for the additional datasets. $|V|$ and $|E|$ denotes the number of nodes and edges respectively, C, $\mathbf{X}_d$, and deg denotes the number of classes, size of feature dimension and the average degree of the corresponding graph dataset.

	CiteSeer	PubMed	Credit
$\|V\|$	3327	19717	3000
$\|E\|$	9228	88651	28854
$\mathbf{X}_d$	3703	500	13
C	6	3	2
deg	2.77	4.50	9.62

Table 11: RDT-Fidelity and sparsity (entropy) of different explanation methods on the additional datasets. For fidelity, the higher the better. For sparsity, the lower the better.

Exp	CiteSeer		PubMed		Credit	
	Fidelity	Sparsity	Fidelity	Sparsity	Fidelity	Sparsity
Grad	0.24	4.15	0.44	4.38	0.55	0.94
Grad-I	0.19	4.16	0.37	4.39	0.59	0.95
Zorro-S	0.90	2.62	0.96	2.17	0.94	1.23
GLime	0.18	0.65	0.38	1.59	0.60	0.49
GNNExp	0.65	8.21	0.79	6.21	0.65	2.56

sparsity. More work can be done in this direction. In general, our fidelity and sparsity metrics can be complemented with other domain and dataset-specific metrics of explanation utility.

Table 12: Attack performance and baselines for additional datasets (CiteSeer, PubMed, and Credit). The best performing attack(s) on each explanation method is(are) highlighted in bold, and the second best attack(s) is(are) underlined.

Exp	Attack	CiteSeer		PubMed		Credit	
		AUC	AP	AUC	AP	AUC	AP
Baseline	FeatureSim	0.890 ± 0.04	0.909 ± 0.03	0.888 ± 0.01	0.889 ± 0.01	0.720 ± 0.01	0.753 ± 0.01
	Lsa [20]	0.859 ± 0.03	0.893 ± 0.02	0.783 ± 0.05	0.801 ± 0.04	0.655 ± 0.03	0.671 ± 0.04
	GraphMI [48]	0.789 ± 0.02	0.746 ± 0.02	0.795 ± 0.01	0.758 ± 0.02	0.510 ± 0.04	0.569 ± 0.05
	Slaps [13]	0.834 ± 0.04	0.875 ± 0.03	0.579 ± 0.02	0.626 ± 0.02	0.723 ± 0.02	0.767 ± 0.02
Grad	GSEF-concat	0.811 ± 0.03	0.860 ± 0.03	0.672 ± 0.02	0.732 ± 0.01	0.825 ± 0.02	0.868 ± 0.02
	GSEF-mult	0.748 ± 0.05	0.808 ± 0.04	0.579 ± 0.03	0.627 ± 0.03	0.820 ± 0.02	0.861 ± 0.01
	GSEF	0.969 ± 0.02	0.971 ± 0.02	0.788 ± 0.02	0.856 ± 0.02	**0.881 ± 0.02**	**0.918 ± 0.02**
	GSE	0.945 ± 0.03	0.955 ± 0.02	0.770 ± 0.03	0.846 ± 0.02	0.861 ± 0.02	0.899± 0.01
	ExplainSim	**0.991 ± 0.01**	**0.985 ± 0.01**	**0.987 ± 0.01**	**0.981 ± 0.01**	0.861 ± 0.01	0.867 ± 0.01
Grad-I	GSEF-concat	0.810 ± 0.04	0.857 ± 0.04	0.583 ± 0.02	0.632 ± 0.02	0.826 ± 0.02	0.867 ± 0.02
	GSEF-mult	0.745 ± 0.05	0.804 ± 0.04	0.574 ± 0.01	0.625 ± 0.01	0.813 ± 0.02	0.863 ± 0.01
	GSEF	0.970 ± 0.02	0.974 ± 0.02	0.782 ± 0.02	0.853 ± 0.02	**0.881 ± 0.02**	**0.918 ± 0.01**
	GSE	0.947 ± 0.02	0.960 ± 0.02	0.740 ± 0.02	0.836 ± 0.01	0.861 ± 0.02	0.893 ± 0.02
	ExplainSim	**0.992 ± 0.001**	**0.986 ± 0.001**	**0.988 ± 0.001**	**0.981 ± 0.001**	0.864 ± 0.001	0.868 ± 0.001
Zorro-S	GSEF-concat	0.932 ± 0.02	0.944 ± 0.02	0.663 ± 0.02	0.729 ± 0.01	0.825 ± 0.03	0.868 ± 0.02
	GSEF-mult	0.828 ± 0.06	0.855 ± 0.06	0.554 ± 0.03	0.601 ± 0.02	0.828 ± 0.03	0.868 ± 0.02
	GSEF	**0.953 ± 0.02**	**0.960 ± 0.02**	**0.901 ± 0.02**	**0.914 ± 0.02**	**0.822 ± 0.02**	**0.869 ± 0.02**
	GSE	0.932 ± 0.03	0.947 ± 0.03	0.753 ± 0.02	0.772 ± 0.02	0.785 ± 0.01	0.840 ± 0.01
	ExplainSim	0.950 ± 0.01	0.957 ± 0.01	0.874 ± 0.02	0.904 ± 0.01	0.734 ± 0.02	0.768 ± 0.02
Glime	GSEF-concat	0.757 ± 0.05	0.801 ± 0.05	0.614 ± 0.03	0.641 ± 0.02	0.825 ± 0.03	0.867 ± 0.02
	GSEF-mult	0.567 ± 0.03	0.637 ± 0.04	0.507 ± 0.10	0.509 ± 0.08	0.791 ± 0.01	0.855 ± 0.01
	GSEF	**0.862 ± 0.05**	**0.894 ± 0.04**	**0.697 ± 0.02**	**0.712 ± 0.02**	**0.845 ± 0.02**	**0.890 ± 0.01**
	GSE	0.571 ± 0.04	0.618 ± 0.04	0.602 ± 0.03	0.591 ± 0.03	0.798 ± 0.03	0.843 ± 0.03
	ExplainSim	0.703 ± 0.06	0.705 ± 0.05	0.661 ± 0.03	0.724 ± 0.02	0.765 ± 0.03	0.810 ± 0.02
GNNExp	GSEF-concat	0.639 ± 0.04	0.713 ± 0.04	0.552 ± 0.03	0.590 ± 0.03	**0.826 ± 0.03**	**0.869 ± 0.02**
	GSEF-mult	0.812 ± 0.05	0.862 ± 0.04	0.570 ± 0.01	0.618 ± 0.01	0.821 ± 0.02	0.865 ± 0.02
	GSEF	**0.847 ± 0.05**	**0.881 ± 0.05**	**0.612 ± 0.02**	**0.648 ± 0.02**	0.817 ± 0.02	0.869 ± 0.02
	GSE	0.594 ± 0.07	0.640 ± 0.06	0.507 ± 0.02	0.512 ± 0.02	0.760 ± 0.04	0.804 ± 0.05
	ExplainSim	0.618 ± 0.05	0.654 ± 0.05	0.495 ± 0.01	0.499 ± 0.01	0.787 ± 0.02	0.845 ± 0.02

Table 13: 100-runs results with standard deviation. The best performing attack(s) on each explanation method is(are) highlighted in bold, and the second best attack(s) is(are) underlined.

Exp	Attack	Cora		CoraML		Bitcoin		CiteSeer		PubMed		Credit	
		AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Grad	GSEF-concat	0.740 ± 0.05	0.780 ± 0.04	0.639 ± 0.07	0.703 ± 0.05	0.474 ± 0.10	0.503 ± 0.07	0.814 ± 0.04	0.862 ± 0.03	0.666 ± 0.03	0.728 ± 0.02	0.825 ± 0.03	0.868 ± 0.02
	GSEF-mult	0.682 ± 0.05	0.742 ± 0.04	0.667 ± 0.07	0.727 ± 0.06	0.283 ± 0.03	0.392 ± 0.02	0.750 ± 0.05	0.806 ± 0.04	0.576 ± 0.02	0.627 ± 0.02	0.820 ± 0.02	0.861 ± 0.01
	GSEF	**0.949 ± 0.02**	**0.955 ± 0.01**	**0.801 ± 0.07**	**0.836 ± 0.06**	**0.563 ± 0.15**	**0.552 ± 0.14**	**0.969 ± 0.02**	**0.972 ± 0.02**	**0.787 ± 0.02**	**0.855 ± 0.02**	**0.878 ± 0.02**	**0.916 ± 0.01**
	GSE	0.919 ± 0.04	0.935 ± 0.02	0.697 ± 0.07	0.767 ± 0.05	0.265 ± 0.04	0.380 ± 0.02	0.946 ± 0.03	0.955 ± 0.02	0.767 ± 0.03	0.845 ± 0.02	0.868 ± 0.02	0.902 ± 0.01
Grad-I	GSEF-concat	0.727 ± 0.05	0.767 ± 0.04	0.666 ± 0.06	0.719 ± 0.05	0.523 ± 0.08	0.528 ± 0.05	0.813 ± 0.04	0.860 ± 0.04	0.578 ± 0.02	0.628 ± 0.02	0.824 ± 0.03	0.867 ± 0.02
	GSEF-mult	0.689 ± 0.02	0.741 ± 0.02	0.699 ± 0.06	0.747 ± 0.06	0.289 ± 0.06	0.403 ± 0.04	0.738 ± 0.04	0.797 ± 0.04	0.573 ± 0.02	0.625 ± 0.02	0.810 ± 0.02	0.860 ± 0.01
	GSEF	**0.949 ± 0.02**	**0.953 ± 0.02**	**0.792 ± 0.07**	**0.830 ± 0.06**	**0.557 ± 0.05**	**0.547 ± 0.03**	**0.969 ± 0.02**	**0.973 ± 0.02**	**0.784 ± 0.02**	**0.854 ± 0.02**	**0.879 ± 0.02**	**0.918 ± 0.01**
	GSE	0.900 ± 0.04	0.921 ± 0.03	0.706 ± 0.06	0.772 ± 0.05	0.242 ± 0.05	0.370 ± 0.02	0.945 ± 0.03	0.958 ± 0.02	0.734 ± 0.03	0.834 ± 0.02	0.858 ± 0.02	0.892 ± 0.02
Zorro-S	GSEF-concat	**0.909 ± 0.03**	**0.924 ± 0.02**	0.750 ± 0.06	0.790 ± 0.05	0.573 ± 0.04	**0.569 ± 0.03**	0.935 ± 0.02	0.947 ± 0.02	0.663 ± 0.02	0.730 ± 0.02	0.826 ± 0.03	0.868 ± 0.02
	GSEF-mult	0.796 ± 0.05	0.821 ± 0.05	0.717 ± 0.06	0.744 ± 0.06	0.512 ± 0.09	0.512 ± 0.06	0.826 ± 0.06	0.853 ± 0.05	0.559 ± 0.03	0.606 ± 0.02	**0.829 ± 0.03**	**0.869 ± 0.02**
	GSEF	0.902 ± 0.02	0.922 ± 0.02	**0.772 ± 0.06**	**0.811 ± 0.05**	**0.602 ± 0.03**	0.568 ± 0.03	**0.953 ± 0.03**	**0.961 ± 0.03**	**0.686 ± 0.02**	**0.754 ± 0.02**	0.818 ± 0.03	0.867 ± 0.02
	GSE	0.889 ± 0.04	0.915 ± 0.02	0.745 ± 0.07	0.787 ± 0.06	0.547 ± 0.04	0.546 ± 0.04	0.933 ± 0.03	0.948 ± 0.03	0.664 ± 0.02	0.730 ± 0.01	0.785 ± 0.02	0.841 ± 0.01
GLime	GSEF-concat	0.655 ± 0.06	0.716 ± 0.05	0.614 ± 0.06	0.655 ± 0.05	**0.537 ± 0.07**	**0.520 ± 0.05**	0.762 ± 0.05	0.802 ± 0.05	0.614 ± 0.04	0.643 ± 0.03	0.827 ± 0.03	0.869 ± 0.02
	GSEF-mult	0.518 ± 0.05	0.540 ± 0.05	0.559 ± 0.06	0.564 ± 0.05	0.232 ± 0.03	0.358 ± 0.01	0.566 ± 0.04	0.635 ± 0.05	0.529 ± 0.07	0.513 ± 0.06	0.791 ± 0.01	0.855 ± 0.01
	GSEF	**0.743 ± 0.04**	**0.784 ± 0.03**	**0.649 ± 0.06**	**0.697 ± 0.05**	0.516 ± 0.05	0.512 ± 0.03	**0.853 ± 0.05**	**0.887 ± 0.05**	**0.680 ± 0.03**	**0.698 ± 0.03**	**0.844 ± 0.03**	**0.889 ± 0.02**
	GSE	0.534 ± 0.05	0.565 ± 0.05	0.523 ± 0.05	0.542 ± 0.05	0.274 ± 0.05	0.373 ± 0.02	0.581 ± 0.05	0.620 ± 0.04	0.578 ± 0.05	0.569 ± 0.05	0.790 ± 0.03	0.830 ± 0.04
GNNExp	GSEF-concat	0.628 ± 0.04	0.648 ± 0.05	0.642 ± 0.06	0.695 ± 0.05	0.476 ± 0.09	0.497 ± 0.05	0.628 ± 0.05	0.704 ± 0.04	0.555 ± 0.02	0.596 ± 0.02	**0.825 ± 0.03**	0.868 ± 0.02
	GSEF-mult	0.730 ± 0.05	0.768 ± 0.05	**0.645 ± 0.06**	**0.699 ± 0.05**	0.393 ± 0.11	0.453 ± 0.05	0.805 ± 0.05	0.854 ± 0.04	0.564 ± 0.02	0.613 ± 0.02	0.819 ± 0.02	0.863 ± 0.02
	GSEF	**0.762 ± 0.04**	**0.794 ± 0.03**	0.644 ± 0.07	0.684 ± 0.06	**0.601 ± 0.04**	**0.574 ± 0.03**	**0.856 ± 0.05**	**0.887 ± 0.04**	**0.616 ± 0.02**	**0.644 ± 0.02**	0.818 ± 0.02	**0.868 ± 0.02**
	GSE	0.515 ± 0.05	0.542 ± 0.04	0.500 ± 0.06	0.518 ± 0.05	0.311 ± 0.08	0.407 ± 0.04	0.593 ± 0.05	0.647 ± 0.05	0.499 ± 0.02	0.507 ± 0.02	0.766 ± 0.03	0.816 ± 0.03

Table 14: Hyperparameters of the different modules of the attack and target model

Param/Module	Self-supervision	Classification	Target Model
num_layers	2	2	2
learning rate	0.01	0.001	0.01
epochs	2000	200	200
hidden_size	512	32	32
dropout	0.5	0.5	0.5
noisy_mask_ratio	20	–	–
weight_decay	–	–	5e-4