Group Property Inference Attacks Against Graph Neural Networks

Xiuling Wang
Stevens Institute of Technology
Hoboken, NJ, USA
xwang193@stevens.edu

Wendy Hui Wang
Stevens Institute of Technology
Hoboken, NJ, USA
hwang4@stevens.edu

ABSTRACT

Recent research has shown that machine learning (ML) models are vulnerable to privacy attacks that leak information about the training data. In this work, we consider Graph Neural Networks (GNNs) as the target model, and focus on a particular type of privacy attack named *property inference attack* (PIA) which infers the sensitive properties of the training graph through the access to GNNs. While the existing work has investigated PIAs against graph-level properties (e.g., node degree and graph density), we are the first to perform a systematic study of the *group property inference attacks* (GPIAs) that infer the distribution of particular groups of nodes and links (e.g., there are more links between male nodes than those between female nodes) in the training graph. First, we consider a taxonomy of threat models with various types of adversary knowledge, and design six different attacks for these settings. Second, we demonstrate the effectiveness of these attacks through extensive experiments on three representative GNN models and three real-world graphs. Third, we analyze the underlying factors that contribute to GPIA's success, and show that the GNN model trained on the graphs with or without the target property represents some dissimilarity in model parameters and/or model outputs, which enables the adversary to infer the existence of the property. Further, we design a set of defense mechanisms against the GPIA attacks, and demonstrate empirically that these mechanisms can reduce attack accuracy effectively with small loss on GNN model accuracy.

CCS CONCEPTS

• **Security and privacy → Software and application security**.

KEYWORDS

Property inference attack; Graph neural networks; Privacy attacks and defense; Trustworthy machine learning.

ACM Reference Format:
Xiuling Wang and Wendy Hui Wang. 2022. Group Property Inference Attacks Against Graph Neural Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3560662

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560662

1 INTRODUCTION

Advances in machine learning (ML) in recent years have enabled a large array of applications such as data analytics, autonomous systems, and security diagnostics. Building good ML models, however, require computational resources and possibly significant financial investment. Small companies as well as developers and researchers with limited resources may not be able to afford such impractical cost. This motivates the creation of online ML model marketplaces where ML models are shared and traded [1, 3]. However, since these models may be trained on the data that contains sensitive information, it raises an important question: *how much does the model reveal about the training data?*

Recent studies have identified a number of attacks to infer the sensitive information in the training data. For example, membership inference attacks [31, 34] infer whether a particular data sample was used in the training of the ML models. Model inversion attacks reconstruct the training examples given the access to the target model [15, 16, 46]. These attacks focus on the privacy of *individual records* in the dataset. On the other hand, property inference attacks [8, 18, 29] infer the aggregate information (property) of the dataset.

In this paper, we consider Graph Neural Networks (GNNs) as the target model. We consider the *group properties* that are defined over distribution of nodes and links as the attack target. An example of the node-level group property is that a professional network graph contains more male users than female ones, while an example of the link-level group properties is that a social network graph has more links between White users than those between African American users. Obtaining these properties by the adversary may directly violate the intellectual property (IP) of the model owner [49].

In general, property inference exploits the idea that ML models trained on similar datasets will represent some similarity in the model parameters and/or the model outputs [8, 18]. Following this idea, various PIA models have been designed to attack the classification models [8], deep neural networks [18, 29], and generative adversarial networks [49]. All these works mainly focus on ML models trained on tabular and image data. Few works [36, 47, 48] have studied PIA against GNN models over graph data. And none of these works have investigated the leakage of nodes and links at group level. More detailed comparison between these works and ours can be found in Table 1 and Section 2.

Intuitively, the attacker can infer the data properties by inferring the values of the property features (e.g., Gender) from the model outputs through the attribute inference attacks (AIAs) [12, 35, 42]. However, the effectiveness of AIAs highly relies on several assumptions such as strong correlation between the property features and the label as well as the adversary knowledge of the property features of a subset of nodes in the original graph [12, 35]. These

	GNN model	Adversary knowledge		Target properties	
		White-box	Black-box	Graph-level	Group-level
[36]	GCN	Model loss	×	Avg. node degree	×
[47]	GCN	×	Model prediction	×	Node group distribution
[48]	GraghSAGE	Graph embedding	×	# of nodes & edges, graph density	×
Ours	GCN, GraghSAGE, GAT	Node embedding	Model prediction	×	Node & link group distribution

Table 1: Comparison between the existing works on property inference attacks against GNNs and ours.

assumptions may not hold for PIAs. Indeed, both prior work [47] and our analysis in this paper show that PIA can incur leakage even when the property features are weakly correlated with the task label.

In this paper, we perform the first systematic investigation of vulnerability of GNNs against *group property inference attacks* (GPIA). We consider both black-box and white-box settings. Under the former setting, the adversary only can access the output of the target model (e.g., posterior probability), while under the latter setting, he/she can access the architecture and parameters (e.g., node embeddings) of the target model. For both settings, we consider a comprehensive taxonomy of the threat model with various types of adversary knowledge, and design six attacks for these settings. All these attacks are designed as the classification models which are trained to distinguish the positive graphs (i.e., the graphs with the properties) from the negative ones (i.e., the graphs without the properties) by the behaviors and outputs of either the target model or a *shadow model* that mimics the target model on these graphs.

We evaluate the effectiveness of the proposed attacks on three representative GNN models, namely, GCN [24], GraphSAGE [19], and GAT [38], and three real-world graph datasets. The results demonstrate that our attacks are effective under various settings. For example, a black-box GPIA attack with only 20% of the training graph available in the adversary knowledge can achieve the accuracy in the range of [0.9, 1] and [0.72, 0.92] for node and link properties respectively. The attack accuracy remains to be effective when the adversary transfers the knowledge learned from a shadow graph to infer the properties in the target graph, even when the shadow and target graphs have different structures and domains. Furthermore, our attacks greatly outperform the baseline methods that utilize the attribute inference attacks (AIAs) [12, 35, 42] and the meta-classifier [8, 18, 47].

Next, we analyze the main factors that contribute to GPIA's success. We found that, due to the (indirect) correlation between the property feature and the label as well as the non-negligible disparity in the *influence* of different node/link groups on the target model, the model parameters (node embeddings) and model outputs by the target model trained on the data with the target property $\mathbb{P}$ is distinctly dissimilar to those obtained from the data without $\mathbb{P}$, which enables the adversary to infer the existence of $\mathbb{P}$.

Further, we design three defense mechanisms to mitigate the vulnerabilities of GPIA under both black-box and white-box settings. For the former setting, we add Laplace noise to perturb the posterior output. For the latter setting, we design two defense mechanisms, namely adding Laplace noise on the node embeddings and compressing node embeddings. We evaluate the performance of these defense mechanisms, and show that these defense mechanisms can reduce the attack accuracy significantly with either a small amounts of noise or a small compression ratio. Furthermore, both defense mechanisms address the trade-off between privacy and model accuracy; the model accuracy is still acceptable when the defense is sufficiently strong.

In summary, we make the following contributions in this paper:

- We design the first set of attacks against GNNs that can infer the properties of groups of nodes and links in the training graph.
- We perform extensive empirical studies and demonstrate the effectiveness of our proposed attacks.
- We analyze the main factors that contribute to the success of the attacks.
- We design three defense mechanisms and demonstrate their effectiveness against the proposed attacks.

2 RELATED WORK

Privacy attacks against GNNs. Many studies have explored the privacy vulnerability of GNNs. Based on which assets in GNN models are considered as sensitive and the adversary tries to obtain, these privacy attacks can be grouped into two categories: (1) *privacy attacks on GNN models* that aim to extract information about the model's structure and parameters; and (2) *privacy attacks on training data* that aim to infer the sensitive information in the training graph. There has been few studies on privacy attacks on GNN models: Wu et al. [44] designed the *model extraction attack* that aims to reconstruct a duplicated GNN model. In terms of privacy attacks on training data of GNNs, He et al. [20] designed the link stealing attacks to infer if some specific links exist in the training graph. Duddu et al. [12] designed three privacy attacks against GNNs - a *membership inference attack* that infers whether a graph node was in the training data, a *graph reconstruction attack* that reconstructs the target graph, and an *attribute inference attack* that infers the sensitive attributes. He et al. [21] proposed the node-level membership inference attacks against GNNs. Wu et al. [45] considered the data partition setting where each data holder has either node features or edge information, and proposed a link-level membership inference attack to infer the existence of links. Zhang et al. [48] designed three inference attacks against GNNs: (1) a *property inference attack* that infers the graph-level information such as graph density and number of nodes/edges of the training graph; (2) a *subgraph inference attack* that infers whether a given subgraph is contained in the training graph; and (3) a *graph reconstruction attack* that reconstructs the structure of the training graph.

Property inference attacks against ML models. Ateniese et al. [8] first proposed the concept of the property inference attack against ML models. They design a white-box PIA model and demonstrate its effectiveness against SVM and HMM models. In the following years, the design of PIA has been extended to fully connected neural networks [18], Convolutional Neural Networks (CNNs) [32], collaborative learning [29], federated learning [39, 41], and GANs

[49]. Mahloujifar et al. [10] designed a property inference poisoning attack by which the adversary can learn a particular property in the training data by injecting specially crafted poison data in the data. Unlike their work, we assume the adversary has no update access to the training data.

Property inference attacks against GNNs. Very few works [36, 47, 48] have studied property inference attacks against GNNs. Suri et al. [36] proposed a generic definition of PIA which defines the attack goal as distinguishing between two possible training distributions. They assume that the adversary has the access to (transformed) data distributions, while we assume the adversary only has access to either node embeddings or posterior probabilities. Zhang et al. [47] studied the leakage of properties of node group distribution in the centralized multi-party setting. They show that PIA can incur leakage even when the property attribute is not correlated with the label. Probably the property inference attack in [48] is the most relevant to our work. However, it differs from our attack fundamentally from the following perspectives. First, [48] considers the properties at graph level (e.g., number of nodes/edges and graph density), while we consider the properties of nodes and links at group level. Second, [48] considers the graph embedding (i.e., the vector representation of the whole graph) in the adversary knowledge, while we consider node embeddings. These major differences in the type of properties and adversary knowledge lead to fundamentally different design of PIA models. Furthermore, besides the empirical results to demonstrate the effectiveness of PIA and its defenses, we provide in-depth investigation of which factor(s) contribute to GPIA's success.

3 GRAPH NEURAL NETWORK

In general, GNNs take an input graph $G(V, E)$, along with a set of node features, to generate a representation vector z_i (node embedding) for each node $v_i \in V$. One of the defining features of GNN models is that is uses a form of *neural message passing* by which vector messages are exchanged between nodes in the graph and updated using neural networks.

In particular, during each message-passing iteration in a GNN, the embedding $z_i^{(\ell)}$ corresponding to each node $v_i \in V$ at layer ℓ is updated according to v_i's graph neighborhood $\mathcal{N}(v_i)$ (typically 1-hop neighborhood). This update process can be expressed as:

$$z_i^{\ell+1} = \mathsf{UPDATE}^{\ell}(z_i^{\ell}, \mathsf{AGGREGATE}^{\ell}(\{z_j^{\ell}, \forall v_j \in \mathcal{N}(v_i)\})), \quad (1)$$

where UPDATE and AGGREGATE are arbitrary differentiable functions (e.g., neural networks). The initial embeddings at $\ell = 0$ are set to the input features for all the nodes, i.e., $z_i^0 = x_i, \forall v_i \in V$.

After k iterations of message passing, a Readout function pools the node embeddings at the last layer and produces the prediction results. The Readout function varies by the learning tasks. In this paper, we consider node classification as the learning task. For this task, often the Readout function is a softmax function. The prediction output for each node v is a vector of probabilities, each corresponding to the predicted probability (posterior) that v is assigned to a class.

In this paper, we consider three representative GNN models, namely **Graph Convolutional Network (GCN)** [24], **GraphSAGE** [19], and **Graph Attention network (GAT)** [38]. These three models mainly differ on either AGGREGATE and UPDATE functions. More details of the two functions for the three GNN models can be found in our full version [40].

Symbol	Meaning
$v/e(v_i, v_j)$	node/link between two nodes v_i, v_j
A/X	Property/non-property feature
$\mathbb{P}$	Target property
G/G^S	Target/shadow graph
T/T^S	Target/shadow model
P	GPIA attack classifier
T^{train}, T^{test}	Training and testing datasets of target model T
P^{train}, P^{test}	Training and testing datasets of GPIA model P
Z^i	Node embedding generated at the ith-layer of T

Table 2: Notations

4 PROBLEM FORMULATION

Given a graph $G(V, E)$ and a GNN model T trained on G, the goal of GPIA is to infer whether G has a group property $\mathbb{P}$ from the access to T. Table 2 lists the common notations used in the paper.

4.1 Group Properties

In this paper, we consider two types of properties that the adversary aims to infer: *node group properties* (node properties) that specify the aggregate information of particular node groups; and *link group properties* (link properties) that specify the aggregate information of particular link groups. The property can be either binary or non-binary. An example of the binary property is whether the graph contains more female nodes than male ones. An example of the non-binary property is whether the graph has 75%, or 50%, or 25% female nodes. In this paper, we only consider binary properties. If a graph has the property $\mathbb{P}$, we say it is a *positive* graph. Otherwise, it is a *negative* graph.

Node/link groups. We assume the nodes are associated with a set of features P (called as *property features*) on which the grouping of nodes and links will be defined. For simplicity, we only consider one property feature in this paper. The rest of the node features are called as *non-property features*. Typical examples of the property features include the demographic features such as gender and race. The grouping of nodes and links is specified by adding value-based constraints (VBCs) on the property features. For example, `gender="Male"` defines the male group.

Node properties. The node properties are specified on the property features with aggregate functions and arithmetic comparison operators. In this paper, we consider COUNT() as the aggregate function, and five arithmetic comparison operators including $<$, $\leq$, $>$, $\geq$, $=$, and $\neq$. An example of the node property $\mathbb{P}$ is "`COUNT(Male) > COUNT(Female)`".

Link properties. The link properties are specified on property features of both end nodes in the links, with aggregate functions and arithmetic comparison operators. An example of the link property is "`COUNT(Male-Male) > COUNT(Female-Female)`", i.e., there are more links between male users than between female users.

Attack	Adversary knowledge		
	G^S	Access to T	G^A
A_1	×	White-box	✓
A_2	×	Black-box	✓
A_3	✓	White-box	×
A_4	✓	Black-box	×
A_5	✓	White-box	✓
A_6	✓	Black-box	✓

Table 3: Attack taxonomy (G^S: shadow graph; T: target model; G^A: partial graph).

4.2 Adversary Knowledge

The adversary may have additional background knowledge $\mathbb{K}$ which can be categorized along three dimensions:

- *Partial graph* G^A: the adversary has a subgraph $G^A \subset G$.
- *Shadow graph* G^S: the adversary has a shadow graph (or multiple graphs) G^S which contains its own structure and node attributes. G^S may have different domain and data distribution from G;
- *Target model* T: We consider two types of adversary knowledge of T: the *white-box* access to T, which reveals the model architecture, parameters, and the loss function, and the *black-box* access which allows the adversary to obtain the target model output (i.e., posteriors) only. We also assume that the adversary has the knowledge of the number of classes for the target model.

The assumption of the white-box setting is reasonable and quite common nowadays [8, 18]. For example, some online platforms [3, 5] share their models openly, including their parameters, thereby providing white-box access. On the other hand, ML-as-a-service services (e.g. [1, 2, 4]) that provide an API for users to query for predictions but keep their models inaccessible to users are typical examples of black-box settings.

5 METHODOLOGY

Given a target graph G and a GNN model T trained on G, the adversary aims to infer if G has the property $\mathbb{P}$ by either the white-box access to node embeddings or the black-box access to posterior probabilities output by T. An example for the former case is that the data owner uploads node embeddings to a third-party service provider such as Google's Embedding Projector service[1] to perform downstream analysis tasks, while an example for the latter case is that the data owner uploads the posterior probability (e.g., by a GNN-based recommender system [17]) to a third-party online optimization solver such as Gurobi[2] for optimization. Another possible attack scenario is the collaborative setting under which the attacker and other parties train a model jointly by sharing either the model predictions or node embeddings [47]. The attacker is curious to infer the properties of other parties' data from their shared embeddings/predictions.

Formally, the attack's goal is to design a binary classifier P that can be formulated as: $P : \mathbb{K}, \mathbb{P} \rightarrow L$, where $\mathbb{K}$ denotes the adversary knowledge, and L is the set of class labels for property prediction. In this paper, we only consider binary property (i.e., $L = \{0, 1\}$). We will discuss how to extend to non-binary properties in Section 8.

[1]https://projector.tensorflow.org/
[2]https://www.gurobi.com

Whether the adversary has each of G^A, T, and G^S in $\mathbb{K}$ is a binary choice. However, we assume at least one of G^A and G^S is available for training of GPIA model, as the adversary always can obtain some public graphs from external resources as the shadow graphs if the partial graph is not available. Therefore, we have a comprehensive taxonomy with six different threat models based on different combinations of G^A, T, and G^S in $\mathbb{K}$. We design six GPIA attack classifiers for these threat models, and summarize the taxonomy of our attacks in Table 3. Next, we describe the details of the black-box attacks (A_2, A_4, A_6) first, followed by the details of the white-box attacks (A_1, A_3, A_5). Enlightened by the existing PIA works [47, 48], our attacks also use shadow models. However, due to the assumption of different adversary knowledge (see Table 1), the design of our shadow models is fundamentally different from these works in the design of attack features.

5.1 Black-box Attacks

The black-box attack includes three phases: *shadow model training*, *attack model training*, and *property attack inference* (Figure 1). Next, we explain the details of these three phases.

Shadow model training phase. To collect the data P^{train} to train the GPIA classifier P, first, the adversary trains $k \geq 1$ shadow models $T_1^S, \ldots, T_k^S$. The training data for each shadow model T_i^S is a subgraph G_i^S that is randomly sampled from the partial graph G^A (Attack A_2), the shadow graph G^S (Attack A_4), or both (Attack A_6). Each shadow training graph G_i^S may or may not have the property $\mathbb{P}$. In this paper, we assume all shadow graphs have the same size. Let n_s be the number of nodes in the shadow graphs. Intuitively, to ensure the shadow models mimic the behaviors of the target model, they should be trained in the way that the output of each shadow model T_i^S on the shadow training dataset G_i^S is close to the output of the target model T on G_i^S. In this paper, we follow the prior works [8, 18] and consider the strongest attack scenario that the shadow models are identical to the target model, i.e., they have the same architecture and parameters.

Attack model training phase. Before training the GPIA classifier, the adversary constructs the attack training data P^{train} by the following procedure. For each trained shadow model T_i^S and its training data G_i^S, the adversary aggregates the set of posterior probability values generated by T_i^S on G_i^S into a vector $\vec{v}_i$. The vector $\vec{v}_i$ is inserted into the GPIA training dataset P^{train} as the features, which is associated with a GPIA label "1" if G_i^S is positive, and "0" otherwise.

How to aggregate multiple posterior probability values into one vector as the GPIA features? We consider two different approaches:

- *Concatenation:* Given n_s nodes in the shadow graph, each associated with ℓ posterior probabilities, there are $N = n_s \times \ell$ posterior probability $p_1, \ldots, p_N$ in total. These N probability values are concatenated into a vector $\vec{v} =< p_1, \ldots, p_N >$ as GPIA features.
- *Element-wise difference* (EWD): For each node v, which is associated with ℓ posterior probability values $p_1, \ldots, p_\ell$, we calculate the *average element-wise difference* p^{diff} of v as follows:

$$p^{\text{diff}} = \frac{1}{\ell(\ell-1)} \sum_{1 \leq i,j \leq \ell, i \neq j} (|p_i - p_j|).$$

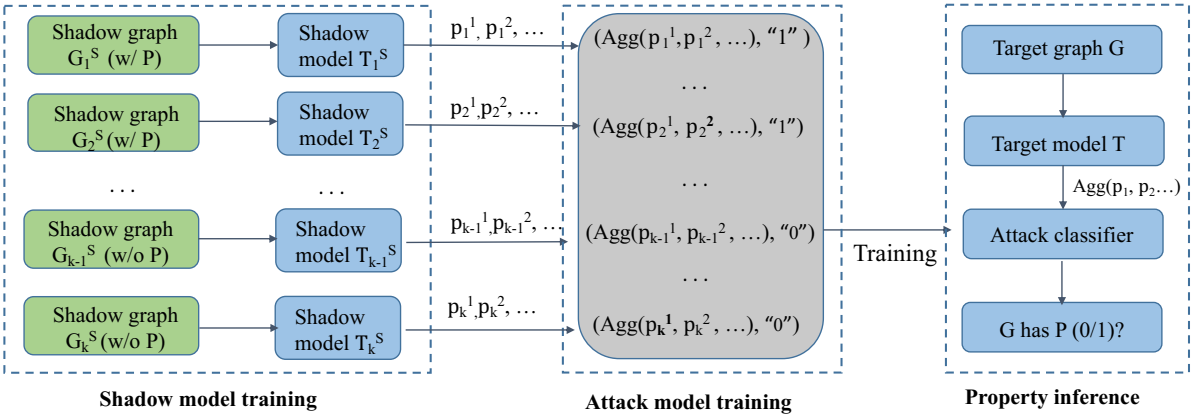


Figure 1: Overview of the black-box GPIA. Agg() is the aggregation function that generates GPIA features from posteriors.

Intuitively, the more (less, resp.) skewed posterior distribution, the higher (lower, resp.) p^{diff} will be. In a uniform distribution of posteriors, $p^{\text{diff}} = 0$. After the element-wise difference of each node is calculated, all values are concatenated into a vector $\vec{v} =< p_1^{\text{diff}}, \ldots, p_{n_s}^{\text{diff}} >$ as GPIA features.

The intuition behind the EWD method is that the distribution of posterior output of positive and negative graphs is significantly different, where such difference can be captured by the element-wise difference value.

After P^{train} is generated, the adversary trains the GPIA classifier P on P^{train}. In this paper, we consider three types of classifiers, namely Multi-layer Perceptron (MLP), Random Forest (RF), and Linear Regression (LR).

Property attack inference phase. At inference time, the adversary computes the aggregated posterior probability output by the target model T on the target graph G, using the same posterior aggregation function in the training phase. Then the adversary feeds the aggregated posterior as the input feature of the testing sample to P, and obtains the predicted GPIA label.

5.2 White-box Attacks

Unlike the black-box attacks that need shadow models, the white-box attacks do not need any shadow model due to its white-box access to the target model. Therefore, the white-box attacks only include two phases, namely, *attack model training* and *property attack inference*. Next, we discuss the details of these two phases.

Attack model training phase. since the adversary has the white-box access to the target model T, he will construct the attack training data P^{train} by using the model parameters of T as the features in P^{train}. The motivation behind this is that the parameters of the models trained on the positive graphs will be more similar than those trained on the negative graphs. Following this, we design the following method to construct P^{train}. For each shadow graph G_i^S, the adversary uses it to train T and obtains all the parameters of T, where the parameters are the node embeddings of G_i^S. The shadow graph G_i^S can be randomly sampled from the partial graph G^A (Attack A_1), the shadow graph G^S (Attack A_3), or both (Attack A_5). Then the adversary aggregates these node embeddings into a vector $\vec{v}_i$, and inserts $\vec{v}_i$ into P^{train} as the features. He further associates $\vec{v}_i$ with a GPIA label "1" if G_i^S is positive, and "0" otherwise.

In general, given a GNN model of k layers, the adversary can choose any $k' \leq k$ layers, and collect the node embeddings of these k' layers to generate GPIA features. We use $A_i^{j_1,\ldots,j_t}$ to indicate that the attack A_i uses the model parameters (i.e., node embeddings) at the j_1-th, …, j_t-th layers of GNN. For example, A_1^2 indicates the attack A_1 that utilizes the embedding at the 2nd layer of the target model T, and $A_1^{1,2}$ indicates the attack A_1 that utilizes the embeddings at both the 1st and 2nd layers of T. There are $2^k - 1$ possible choices of choosing these k' layers in total. Besides these embeddings, the posterior probabilities also can be included to generate features in the same way as in the black-box setting (Section 5.1). We will investigate the impact of choosing different amounts of node embeddings on GPIA performance later (Section 6).

Next, we discuss how to aggregate a set of node embeddings into one vector as the GPIA feature. We consider the following three aggregation methods in this paper. We adapt two pooling methods that have been widely used for Convolutional Neural Networks [9, 26, 43], namely *max-pooling* and *mean-pooling*. Both pooling methods take a set of network parameters in the format of vectors as the input, and summarize these vectors as a single vector of fixed length. Max-pooling preserves the most prominent features, while mean-pooling has a smoothing effect.

- *Concatenation*: Given n_s node embeddings $z_1, \ldots, z_{n_s}$ of the shadow graph, they are concatenated into a vector of dimension n_s, $\vec{v} =< z_1, \ldots, z_{n_s} >$, as GPIA features.
- *Max-pooling*: Given n_s node embeddings $z_1, \ldots, z_{n_s}$ from the shadow graph, we generate a vector $\vec{v} =< z_1^{max}, \ldots, z_{n_s}^{max} >$ as the GPIA features, where z_i^{max} is the maximum of all value in the embedding z_i.
- *Mean-pooling*: Given n_s node embeddings $z_1, \ldots, z_{n_s}$, we generate a vector $\vec{v} =< z_1^{mean}, \ldots, z_{n_s}^{mean} >$, where z_i^{mean} is the mean of all values in z_i.

Different aggregation methods generate different GPIA features, and thus lead to different attack performance. We will investigate the impact of different embedding aggregation methods on GPIA performance in Section 6.

Dataset	# nodes	# edges	# features	# classes
Pokec	45,036	170,964	5	2
Facebook	4,309	88,234	1,284	2
Pubmed	19,717	44,338	500	3

Table 4: Description of datasets

Property attack inference phase. At inference time, the adversary collects the model parameters of the target model T trained on the target graph G, and aggregates the parameters into a vector as the input feature of the testing sample, by using the same embedding aggregation function in the training phase. Therefore, the feature P^{test} is a vector whose size is the same as the number of nodes in G, whereas the feature of P^{train} is a vector whose size is the same as the number of nodes in G^S. Since G and G^S may have different number of nodes, the feature of G and G^S can be of different sizes. This raises the challenge of how to predict on P^{test} if its feature is not of the same size as that of P^{train}.

To address this challenge, we consider four different methods to align the features of P^{train} and P^{test} to be of same dimensions: (1) Sampling: the most straightforward approach is to ensure that G and G^S have the same number of nodes. This can be achieved as the adversary can obtain the knowledge of the number of nodes in G by counting the number of node embeddings via its white-box access to the target model. Then the adversary samples the same number of nodes from G^S. This method is applicable when the number of nodes in G^S is no less than that of G; (2) TSNE projection [37]: it projects high-dimensional data to either two or three-dimensional data. We apply TSNE on the features of P^{train} and P^{test} to project them into the same two-dimensional space, regardless of their original dimensions; (3) PCA dimension reduction [30]: We apply PCA, a widely-used dimension reduction method in the literature, on the feature vector of both P^{train} and P^{test} and project them into a space of the same dimension; (4) Autoencoder dimension compression: Autoencoder [22] compresses the dimensions in the way that the data in the high-dimensional space can be reconstructed from the representation of lower dimension with small error. We apply Autoencoder on the features of P^{train} and P^{test} to compress both into the same space of a lower dimension. To reduce the amounts of information loss by compression, we only compress the feature vector of the larger dimension into the space of the feature vector of smaller one.

Different alignments methods incur different amounts of information loss on the resulting embeddings, and thus lead to different GPIA performance. We will investigate the impact of different alignment methods on GPIA performance in Section 6.

6 EVALUATION

In this section, we aim to demonstrate the effectiveness of GPIA through answering the following three research questions:

- **RQ1** - How effective is GPIA on representative GNN models and real-world graph datasets?
- **RQ2** - Why GPIA work?
- **RQ3** - How various factors (e.g., attack classifier models, embedding/posterior aggregation methods, and complexity of GNN models) affect GPIA effectiveness?

6.1 Experimental Setup

All the experiments are executed on Google COlab with Tesla P100 (16G) and 200GB memory. All the algorithms are implemented in Python with PyTorch. Our code and datasets are available online[3].

Datasets. We consider three real-world datasets, namely *Pokec*, *Facebook*, and *Pubmed* datasets, that are popularly used for graph learning in the literature: (1) **Pokec** social network graph[4] is collected from the most popular on-line social network in Slovakia; (2) **Facebook** social network graph[5] consists of Facebook users as nodes and their friendship relationship as edges; and (3) **Pubmed** Diabetes dataset[6] consists of scientific publications from Pubmed database that are classified into three classes. Each publication node is associated with 500 unique keywords as the features. The links between publications indicate the citation relationship. Table 4 summarizes the information of the three datasets. The purpose of pick two graphs in one domain (social network graphs) and one graph from a different domain is for the validation of the effectiveness of transfer attacks (A_3 and A_4).

Target GNN models. We consider three state-of-the-art GNN models, namely GCN [24][7], GraphSAGE [19] and GAT [38][8], that are widely used by the ML community. For each hidden layer, the number of neurons is 64, which is the same as the dimension of node embedding. We set the number of epoches for training as 1,500, and use early stop with the tolerance as 50. We set the dimension of node embeddings to 64 for all the three datasets.

Properties and property groups. For each dataset, we design one node property and one link property to be attacked. The properties are summarized in Table 5. We pick the keywords "Insulin" (IS) and "streptozotocin" (ST) for Pubmed dataset as they are the keywords of the highest and lowest TF-IDF weight respectively. The successful attacks on these properties can reveal the gender distribution in Facebook and Pokec social network graphs, and the frequency distribution of particular keywords (which can be sensitive) in Pubmed graph.

Implementation of attack classifier. We use three types of attack classifiers for both attacks, namely Multi-layer Perceptron (MLP), Random Forest (RF), and Linear Regression (LR). We use the implementation of the three classifiers provided by sklearn package.[9] We set up the MLP classifier of three hidden layers, with the number of neurons for each layer as 64, 32, 16 respectively. We use ReLU as the activation function for the hidden layers and Sigmoid for the output layer. We train 1,000 epochs with a learning rate of 0.001. We use cross-entropy loss as the loss function and Adam optimizer. For RF classifier, we set the maximum depth as 150 and the minimum number of data points allowed in a leaf node as 1. For LR classifier, we use the L2 norm as the penalty term, and liblinear[10] as the optimization solver. We set the maximum number of iterations as 100 and the early-stop tolerance as 1e-4.

[3] https://anonymous.4open.science/r/PIA-CE14/
[4] https://snap.stanford.edu/data/soc-pokec.html
[5] https://snap.stanford.edu/data/ego-Facebook.html
[6] https://linqs-data.soe.ucsc.edu/public/Pubmed-Diabetes
[7] We use implementation of GCN at https://github.com/tkipf/pygcn
[8] We use the implementation of both GraphSAGE and GAT from DGL package available at https://github.com/dmlc/dgl
[9] https://scikit-learn.org/
[10] Liblinear libary: https://www.csie.ntu.edu.tw/ cjlin/liblinear/

Property	Type	Graph	Property feature	Description
P_1	Node	Pokec	Gender	COUNT(Male)> COUNT(Female)
P_2		Facebook	Gender	COUNT(Male)> COUNT(Female)
P_3		Pubmed	Keyword	COUNT(publications with "IS") > COUNT(publications without "IS")
P_4	Link	Pokec	Gender	COUNT(same-gender links) > COUNT(diff-gender links)
P_5		Facebook	Gender	COUNT(same-gender links) > COUNT(diff-gender links)
P_6		Pubmed	Keyword	COUNT(links btw. papers with "IS") > COUNT(links btw. papers with "ST")

Table 5: Properties to be attacked by GPIA. same-gender (diff-gender, resp.) links indicate those links between users of the same (different, resp.) gender. "IS" = "Insulin"; "ST" = "Streptozotocin".

Partial graphs. We randomly sample 1,000 subgraphs from each dataset as the partial graph. The size of each partial graph is 20%, 25%, and 30% of Pokec, Facebook, and Pubmed datasets respectively.

GPIA training and testing data. For A_1&A_2, we randomly sample 1,000 subgraphs from the same dataset to generate the training and testing data for GPIA. Each subgraph is of the same size as the partial graph. The training/testing split is 0.7/0.3, with the same number of positive and negative subgraphs in both training and testing data. There is no overlap of either links or subgraphs between training and testing data. However, it is challenging to enforce no node overlapping between training and testing data, especially for the datasets with a small number of nodes (e.g., Facebook dataset), as a large portion of sampled subgraphs in the training data will have highly similar structure. Therefore, we allow a small amounts of node overlap between training and testing data (3%, 5%, and 4% for Pokec, Facebook, and Pubmed dataset respectively). We will show the impact of node non-overlapping between training and testing data on attack accuracy in Section 6.4. For attacks A_3&A_4, we sample 700 subgraphs from the shadow graph as the GPIA training data, and 300 subgraphs from the target graph as the testing data. There is no node/link overlap between training and testing data for A_3 and A_4. For attacks A_5&A_6, we sample some subgraphs from the partial graph plus some subgraphs from the shadow graph (700 in total) as the training data, and 300 subgraphs from the target graph as the testing data. We consider various size ratios (1:10, 1:4, 1:2, 1:1, 2:1, 4:1, and 10:1) between partial and shadow graphs in the training data. Similar to A_1 & A_2, there is no link overlap but a small node overlap between training and testing data for A_5 and A_6, where the node overlap ratio does not exceed 5%.

Metrics. We measure classification accuracy as the GNN model performance. We measure *attack accuracy AC* as the effectiveness of the proposed attacks. In particular, $AC = \frac{N_c}{N}$, where N_c is the number of graphs that are correctly predicted by GPIA (either as positive or negative), and N is the total number of graphs in the testing data. Higher AC indicates that GPIA is more effective.

Baselines. We consider three approaches as baselines for comparison with our GPIA model: (1) **Attribute inference attack (AIA) (Baseline-1)**: we follow [35] and design an AIA that predicts the values of property features by the access to the embeddings/posteriors. Then we evaluate PIA accuracy based on the predicted values of property features. To ensure fair comparison between AIA and GPIA, we consider the same partial graphs in the adversary knowledge of GPIA for AIA. (2) **K-means clustering (Baseline-2)**: we apply k-means clustering ($k = 2$) on node embeddings and posteriors. Then we measure the average distance between the centroid of each cluster to the embedding/posteriors, and pick the cluster of the smaller distance; (3) **Meta-classifier (Baseline-3)**: We follow [8, 18, 47] and use a meta-classifier as the GPIA classifier. We also have two additional threshold-based baseline methods. More details of this method and its comparison with ours can be found in our full version [40].

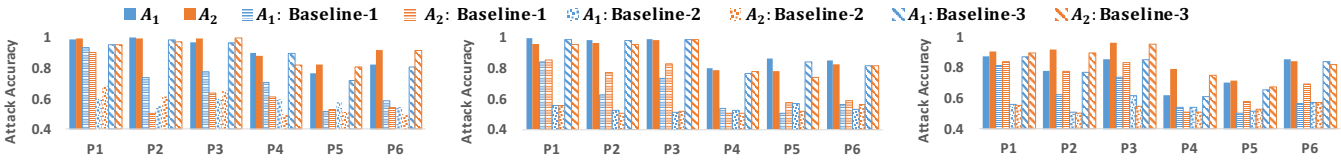
6.2 GPIA Performance (RQ1)

We launch the attacks A_1 - A_6 to attack GCN, GAT, and GraphSAGE, and measure their accuracy. To have a fair comparison of attack accuracy across different settings, we ensure that GPIA training data is of the same size for all the attacks. We use RF and MLP as the white-box and black-box attack classifiers, max-pooling as the embedding aggregation method, concatenation as the posterior aggregation method, and TSNE projection as the embedding alignment method, as these setups produce the best attack performance. More results of the attack performance under different setups can be found in Section 6.4. As the possible hidden layers that the attacker collects node embeddings from is exponential to the number of hidden layers, we only consider GNNs of two hidden layers in this part of experiments to ease explanation.

Performance of target model. To justify why the three GNN models are worthy to be attacked, we evaluate the performance of these models first, and include the results in the full version [40]. As the accuracy of all the three models is significantly higher than the random guess, these models are ready for the launch of GPIA. Furthermore, the three models do not have overfitting; the training-testing gap is in a small range of [0.02, 0.08].

Attacks A_1 and A_2. For A_1, we consider three variants whose GPIA features are generated from the embeddings at the first layer (A_1^1), the second layer (A_1^2), and both layers ($A_1^{1,2}$). We measured the performance of these variants and reported the best GPIA performance. More details of how different amounts of embeddings collected from different layers affect GPIA performance will be discussed in Section 6.4.

Figure 2 shows the attack accuracy of our proposed attacks and the baselines. First, we observe that the attack accuracy of our A_1 and A_2 attacks ranges in [0.62, 1], which is significantly higher than 0.5 (random guess). In some settings (e.g., Figure 2 (a)), the attack accuracy can be as high as close to 1, even under the black-box setting. This demonstrates the effectiveness of GPIA against these



(a) GCN (b) GraphSAGE (c) GAT

Figure 2: Attack accuracy of A_1 and A_2. A_1 and A_2 are indicated in different colors respectively, while our approaches, Baseline-1, Baseline-2, and Baseline-3 are indicated in solid fill, horizontal stripe fill, sphere fill, and diagonal shape fill respectively.

target models. Furthermore, both A_1 and A_2 outperform Baseline-1 in all the settings. Although the superiority of A_1 and A_2 to Baseline-1 is marginal on P_1 for GCN and GAT (Figure 2 (a) & (c)), it is significant for the rest of settings. We believe AIA is much less effective than GPIA for property inference is because there is no strong correlation between the property feature and the label in all the three graphs. More details of the correlation between the property feature and the label can be found in the full version [40]. Similarly, we observe that the accuracy of A_1 and A_2 is also much higher than Baseline-2. We also observe that the performance of A_1 and A_2 is similar to Baseline-3. Thus using either one classifier or stacking multiple classifiers into a meta-classifier does not impact the GPIA performance.

Second, although the attack performance varies across different types of properties, the attack accuracy of GPIA against the three node properties (P_1 - P_3) is noticeably higher than the link properties (P_4 - P_6) in general. The only exception is for the property P_6 and GAT as the target model (Figure 2 (c)), where the attack accuracy of P_6 is close to that of P_3, and higher than P_1 and P_2. One possible reason that GPIA is more successful against the node properties than the link properties is that it only needs to infer the node feature distribution over node features, but it has to infer both node feature distribution and graph structure for the link properties.

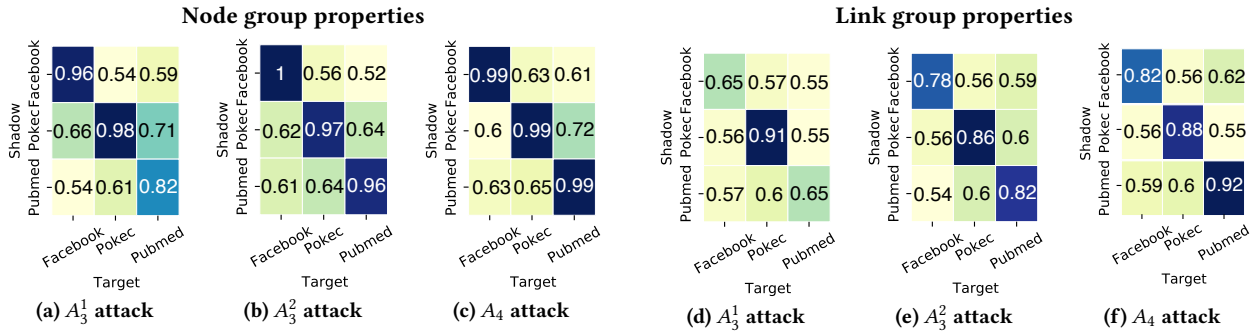
Third, we observe that the accuracy of the white-box and black-box attacks is very close. The difference between them is negligible in most of the cases. Interestingly, the white-box attack does not always outperform the black-box attack, even though its features may include those features used by the black-box attack, possibly due to overfitting of the GPIA classifier by including more features. This demonstrates the power of the property inference - the black-box access to the target model is sufficient to launch the attack.

Attacks A_3 and A_4. Figure 3 presents the results of A_3 and A_4. The results of GAT and GraphSAGE models are included in the full version [40]. In all the settings, A_3 and A_4 are effective as their accuracy is higher than 0.5 (random guess). The attack accuracy can be as high as 0.66 when the adversary uses the Facebook social network graph as the shadow graph to infer whether the Pokec social network graph has disproportionate distribution between male and female users. In other words, GPIA can transfer the knowledge learned from a graph to infer the properties of another graph. However, the accuracy of both A_3 and A_4 is worse than A_1 and A_2. The reason behind this that some amounts of information of the properties embedded in node embeddings/posteriors is lost due to the feature alignment methods used for A_3/A_4. We also observe that GPIA sometimes performs better under the settings that shadow and target datasets belong to different domains than the settings where they belong to the same domain. For example, GPIA accuracy can be as high as 0.72 when Pubmed and Pokec datasets are the target and shadow datasets respectively (Figure 3 (c)), but it is only 0.6 when the target dataset is changed to Facebook dataset while keeping Pokec dataset as the shadow dataset, although both Pokec and Facebook datasets are social network graphs. We analyze the reason behind this observation, and found that the distribution of the GPIA attack features for positive and negative graphs in the shadow graph can be similar to that of the target graph even though they are from different domains and/or of different structure. For example, the distribution of the attack features over positive and negative graphs of Pokec dataset is more similar to Pubmed dataset than Facebook dataset. Thus the attacker can transfer such knowledge learned from the shadow graph for property inference on the target graph successfully.

Attacks A_5 and A_6. We vary the portions of G^S and G^A in the adversary knowledge when we evaluate the effectiveness of A_5 and A_6. Figure 4 present the attack accuracy result of both properties P_1 (node property) and P_4 (link property) with the partial graph sampled from Pokec dataset and the shadow graph sampled from Facebook dataset. Note that both Pokec and Facebook are social network graphs. The performance of other settings can be found in the full version [40]. We have the following observations. First, the accuracy of A_5 and A_6 in all the settings is higher than 0.5, i.e., both attacks are effective. Furthermore, the attack accuracy increases as the size of the partial graph grows. In particular, when the partial graph size dominates the shadow graph size (e.g., when the ratio exceeds 4:1), the attack accuracy against the property P_1 can be close to 1 for both A_5 and A_6 for GCN model, and no less than 0.7 for GAT and GraphSAGE. Second, the attack accuracy of A_5 and A_6 on P_1 is lower than that of the attacks A_1 and A_2 (i.e., only the partial graph is available), similarly for P_4. This is because the features collected from the shadow graphs may not have consistent distribution with those collected from the partial graph, and thus becomes "noise" and degrades GPIA performance. On the other hand, the attack accuracy of A_5 and A_6 is higher than that of A_3 and A_4. This is unsurprising as, compared with A_3/A_4, A_5 and A_6 utilizes the additional knowledge learned from the partial graph to improve its accuracy.

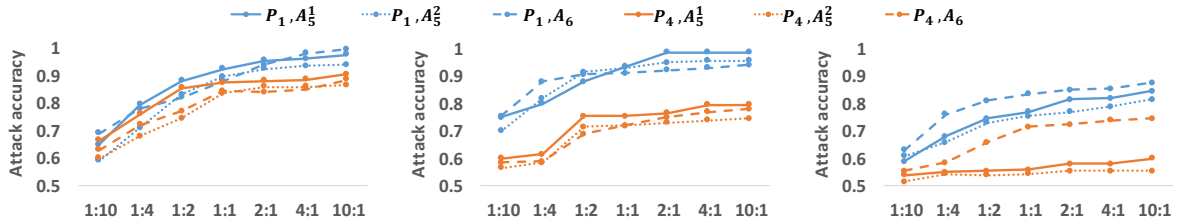
6.3 Why Does GPIA Work? (RQ2)

As the experimental results have demonstrated the effectiveness of GPIA, next, we analyze why GPIAs can infer the existence of property in the training graph successfully. Conducting the theoretical analysis is very challenging due to the complexity in both training



(a) A_3^1 attack (b) A_3^2 attack (c) A_4 attack (d) A_3^1 attack (e) A_3^2 attack (f) A_4 attack

Figure 3: Attack accuracy of A_3 and A_4 when the GCN model is the target model. A_3^1 and A_3^2 indicate the A_3 attack that uses the model parameters at Layer 1 and Layer 2 of GCN models respectively.



(a) GCN (b) GraphSAGE (c) GAT

Figure 4: Attack accuracy of A_5 and A_6 against the properties P_1 and P_4. The partial and shadow graphs are sampled from Pokec and Facebook datasets respectively. X-axis shows the size ratio between partial and shadow graphs.

data and GNN models. Thus we discuss why GPIAs work based on practical evaluations.

Correlation between property feature and label. Intuitively, the attacker can infer the properties from the model output possibly because the property feature is strongly correlated with the task (i.e., the class label). Following this intuition, we measure the Pearson correlation between the property feature and class label of the three graph datasets. These results can be found in the full version [40]. Essentially, the correlation between the property feature and the task label is weak for all the three datasets. Then why the properties can be leaked even when there is weak correlation between the property features and the label? This is possible due to the strong correlation between the property and non-property features in the data. For example, there is a strong Pearson correlation (0.81) between `gender` (property feature) and `height` (non-property feature) in Pokec dataset, between "Insulin" (property feature) and "dietaries" (non-property feature) in Pubmed dataset (Pearson correlation 0.41), and between `gender` (property feature) and `education year` (non-property feature) in Facebook dataset (Pearson correlation 0.92). As these non-property features are correlated with the task label, the information of the properties still can be leaked regardless of whether training data contained the property feature or not.

Non-negligible disparate influence across different groups. As observed by the recent studies, ML models are "biased" in the sense that they behave differently across different groups in the training data [11, 33]. Following this, we measure the disparity in GNN model accuracy across different node/link groups in Facebook and Pokec datasets. The results are included in the full version [40]. We observe the existence of accuracy disparity to some extent for all the three GNN models. In particular, the disparity is significant on Facebook dataset, where the difference in node classification accuracy across male and female groups can be as large as 0.13. The disparity demonstrates that the GNN models behave differently for different node/link groups.

To have a deeper understanding of GNN models' behaviors towards different node/link groups, we measure the *influence score* of individual node/link to quantify the impact of a node/link on the GNN model performance. An intuitive idea measuring the influence of a given training node/link on a GNN model is to ask the counterfactual [14, 25]: *what would happen to the model behaviors if the model did not see the node/link*? Answering this counterfactual enables to connect the model's behaviors with the training data.

To quantify the effect of the counterfactual, we measure the difference in the model behaviors when it is trained with and without a particular node/link. We use gradients to capture the model behaviors. Formally, assuming $\mathbf{g}_v \triangleq \nabla\mathcal{L}(G\backslash\{v\})$, that is, $\mathbf{g}_v$ is the set of gradients induced from the training of the target model T given the graph G excluding the node v, where $\mathcal{L}$ is the loss function of T. Then the *influence score* $I(v)$ *of a node* v on T is measured as

$$I(v) \triangleq distance(\mathbf{g}_v, g),$$

where g is the set of gradients induced by T on G. Intuitively, higher influence score indicates the node v impacts more on T. Similarly, assuming $g_e \triangleq \nabla\mathcal{L}(G\backslash\{e\})$ for a given edge $e \in G$, the *influence score* $I(e)$ *of an edge* e on T is measured as follows:

$$I(e) \triangleq distance(g_e, g). \quad (2)$$

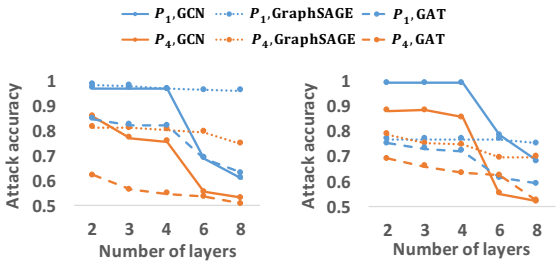
Various distance functions (e.g., Euclidean distance and cosine similarity) can be used. We use cosine similarity as the distance function. After we compute the influence score of individual nodes and links, we compute the average influence score of nodes/links in a particular group as the influence score of the group.

Since influence measurement is time consuming as it needs model retraining, we take a set of samples of Facebook dataset, with each sample containing ~ 700 nodes, and compute the average influence score for male and female groups (for property P_2), as well as the same-gender and diff-gender links (for property P_4) in these samples. From the results reported in Table 6, we observe that *all GNN models have noticeable disparity in the influence scores across different groups. Moreover, which group has higher influence is not solely determined by its group size. It is also dependent on the target model.* For example, as shown in Table 6, the Male group in Facebook dataset has higher influence score on GCN and GraphSAGE but lower score on GAT than the Female group.

We also measure the impact of disparate group influence on their model performance, and report the average group loss in Table 6. The results demonstrate the non-negligible disparity in model loss across different groups. Furthermore, the node groups of higher influence score also have higher loss. However, this does not hold for the link groups, as their influence is measured at link level, while their loss is calculated at node level.

Negligible loss gap between positive and negative graphs. Can the disparate influence and model loss of different node/link groups lead to different target model performance over positive and negative graphs, and thus enables GPIA? To answer this question, we generate a set of positive and negative graphs from the Facebook samples we used in Table 6, and measure the gap between the average loss of the target model trained on positive graphs and that on negative graphs. Our results ("Loss gap" column in Table 6) show that the loss gap between positive and negative graphs is indeed negligible - the loss gap does not exceed 0.01 for all target models (loss values in the range of [0.12, 0.44]). This is not surprising, as making positive graphs to negative graphs (and vice versa) essentially changes the size of a group (e.g., from a minority group to a majority group). However, changing group size does not necessarily lead to higher or lower model loss averaged over the whole graph, given that there is no relationship between group size and its loss. Furthermore, our empirical analysis shows that there is no linear relationship between GPIA accuracy and the loss gap between positive and negative graphs. In particular, GPIA accuracy neither increases or decreases consistently with the growth of the loss gap. Therefore, the loss gap between positive and negative graphs does not contribute to GPIA's success.

Dissimilar distribution of embeddings/posteriors of positive and negative graphs. As different node/link groups have different influence on the target model, how these groups are distributed in the training graph affects the model parameters (embeddings) and posterior outputs obtained from positive and negative graphs. To justify this, we visualize the distribution of GPIA features aggregated from embeddings/posteriors output by the target model on positive and negative graphs (can be found in the full version [40]). We observe that the distribution of GPIA features aggregated from embeddings and posteriors of positive and negative graphs are distinctly dissimilar and well distinguishable. Such dissimilarity is thus utilized by the GPIA classifier to infer the existence of the property in the training data.



(a) Attack A_1 **(b) Attack A_2**

Figure 5: Impact of GNN model complexity on GPIA.

6.4 Impact Factors of GPIA Performance (RQ3)

In this section, we investigate how various factors impact GPIA performance. We consider the following factors: type of attack classifier model, type of embedding /posterior aggregation methods, type of dimension alignment methods, complexity of GNN models, group size ratio, and node overlapping between GPIA training and testing data. We also consider the impact of different amounts of embeddings on GPIA, and observe that GPIA performance stays stable for all the settings. We thus omit this part of discussions and put the results in the full version [40].

Type of attack classifier models. We measure GPIA accuracy when MLP, RF, and LR are used as the attack classifiers, and include the results in the full version [40] due to the limited space. The main observation is that, while the three attack classifiers deliver similar performance in most of the settings, LR never outperforms MLP and RF. Furthermore, RF outperforms MLP slightly in most of the white-box attacks, while MLP has slightly better performance than RF for the black-box attacks. Therefore, we recommend RF and MLP as the white-box and black-box attack classifier respectively.

Embedding/posterior aggregation methods. We measure the impacts of the three embedding aggregation methods (i.e., concatenation, max-pooling and mean-pooling) on GPIA performance, and include the results in the full version [40]. The main observation is that max-pooling outperforms the other two methods in most of the settings. Therefore, we recommend max-pooling as the embedding aggregation method.

We also measured the impact of the posterior aggregation methods (concatenation and element-wise difference) on GPIA performance. Our observation is both methods have similar GPIA performance. The results can be found in the full version [40].

Dimension alignment methods. We measure GPIA performance for the four alignment methods (sampling, TSNE projection, PCA dimension reduction, and Autoencoder compression), and include the results in the full version [40] due to the limited space. The results suggest using TSNE as the dimension alignment method, as it delivers the best attack accuracy. On the other hand, the Autoencoder method always delivers the worst GPIA accuracy among all four alignment methods.

Complexity of GNN models. We define the network complexity by both the number of hidden layers and the total number of neurons in the network, and measure the GPIA performance against the target model of various complexity. We vary the number of

Model	Node group					Link group				
	Influence score		Model loss		Loss	Influence score		Model loss		Loss
	Male	Female*	Male	Female*	gap	Same-gender*	Diff-gender	Same-gender*	Diff-gender	gap
GCN	0.038	0.017	0.647	0.298	0.01	0.022	0.012	0.346	0.604	0.004
GraphSAGE	0.022	0.015	0.558	0.255	0.003	0.0039	0.011	0.353	0.586	0.0007
GAT	0.057	0.087	0.153	0.234	0.004	0.034	0.03	0.21	0.313	0.003

Table 6: Influence scores and GNN model loss per group, and loss gap between positive and negative graphs (Facebook dataset). The group of the larger size is marked with *. Between two groups, the group of higher influence score and higher loss is marked green and orange respectively.

hidden layers from 2 to 8, with 64 neurons at each layer, and use the embedding at the final hidden layer to launch the attack A_1. We only consider A_1 and A_2 in this set of experiments.

Figure 5 shows the results on Pokec dataset. We observe that both A_1 and A_2 are less effective on complex GNNs than the simple ones. For example, when the number of hidden layers increases to 8, the accuracy of both A_1 and A_2 against GAT becomes close to 0.5. Although this is against our initial hypothesis that more complex models would intrinsically learn more information from the training dataset and hence be more sensitive to GPIA, our observation is indeed consistent with the prior PIA studies when CNNs are the target model [32] that it is not necessary that more complex models are more vulnerable to PIA.

Group size ratio. To study if the size ratio between property groups can impact GPIA performance, we consider property P_2 on Facebook dataset and vary the size ratio between Male and Female groups. The results of GPIA performance for these settings can be found in the full version [40]. The main observation is that GPIA performance is affected by group prevalence - the attack accuracy is low (≤ 0.6) when the group size ratio is 1:1, and it grows with the increase of the group size ratio. The attack accuracy can be as high as 1 when the group size ratio increases to 1:3.

Node non-overlapping between GPIA training and testing data. As our results of attacks A_1&A_2 were evaluated over GPIA training and testing data that have small amounts of node overlap, we generate GPIA training and testing data with no node overlapping, and evaluate the accuracy of A_1&A_2. The results can be found in the full version [40]. The main observation is that the attack accuracy for the non-overlapping setting is very close to that for the node-overlapping setting (Figure 2). Thus a small amount of node overlapping between GPIA training and testing data does not affect GPIA accuracy significantly.

7 DEFENSE MECHANISMS

In this section, we present our defense mechanisms against GPIA.

7.1 Details of Defense Mechanisms

Defense against black-box attacks. As GPIA features are generated from the posteriors of the target model, we perturb these posteriors to defend against GPIA. In particular, for each node $v \in G$ and its associated posterior probabilities, we add noise on each probability [48] where the noise follows the Laplace distribution whose density function is given by $\frac{1}{2b}e^{-\frac{x-\mu}{b}}$ (b: noise scale, μ: the location parameter of the Laplace distribution).

Besides the noisy posterior mechanism, we evaluated two alternative methods: (1) *top-k posterior output* method: For each node $v \in G$ and its associated posterior probability values, we keep the top-k largest posteriors as the output. GPIA will be launched on the top-k posterior output; (2) *label-only output* method that the target model outputs the classification label instead of the posteriors. Our results show that both methods fail to either decrease GPIA accuracy significantly or provide acceptable target model accuracy. Thus we will not discuss these two defense mechanisms.

Defense against white-box attacks. We design two types of defense mechanisms that mitigate GPIA effectiveness by modifying the node embeddings: (1) *noisy embedding*: For each node $v \in G$, let z be its node embedding. We add Laplace noise on z, where the noise follows the Laplace distribution whose density function is the same as noisy posterior method; and (2) *embedding truncation*: An embedding of dimension d is converted to another embedding of lower dimension $d' = d \times (1 - r)$, where $r \in (0, 1)$ is the *truncation ratio*. Higher r indicates more dimensions to be truncated and less information is kept in the embedding. We randomly pick $d' < d$ dimensions from the original embedding. Different node embeddings may have different dimensions to be truncated even under the same truncation ratio.

For the embedding truncation method, we implemented and evaluated three embedding dimension reduction methods including PCA [30], TSNE projection [37], and Autoencoder [22]. However, all of them fail to provide strong defense against GPIA, as they still preserve large amounts of information in the embedding which can be utilized by GPIA. Thus we will not present the details of these alternative truncation methods.

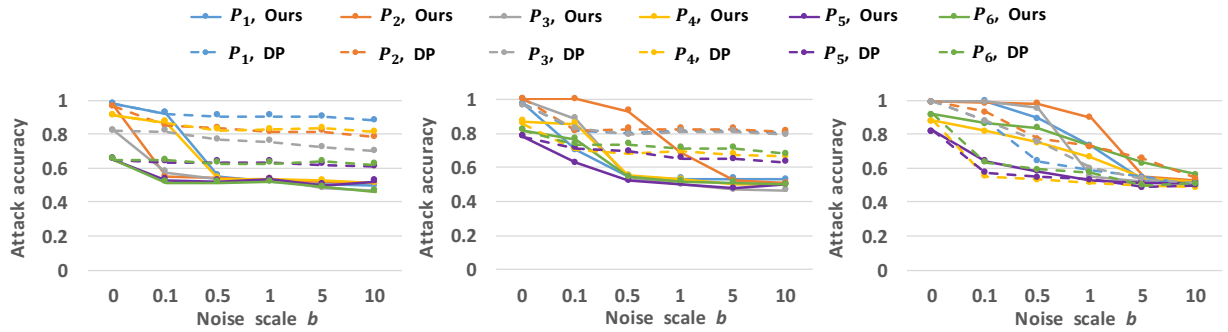
7.2 Evaluation of Defense Mechanisms

We evaluate both effectiveness of the proposed defense methods and their impact on target model accuracy. We only consider attacks A_1 and A_2 as the defense effectiveness against these two attacks are expected to be applied to $A_3 - A_6$ due to their similarities.

Setup of defense mechanisms. For both noisy embedding and noisy posterior defense mechanisms, we set the noise scale $b = \{0.1, 0.5, 1, 5, 10\}$. For the embedding embedding truncation defense, we consider the compression ratio $r = \{0.01, 0.05, 0.1, 0.2, 0.3\}$. The setup of the target model is the same as in Section 6.

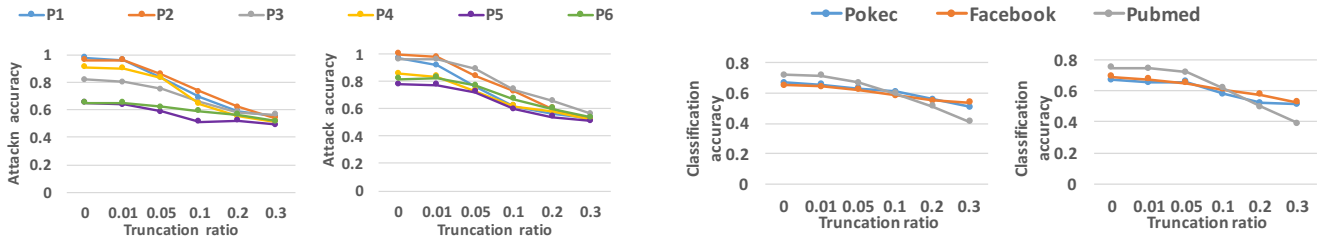
Metrics. We measure *defense effectiveness* as the accuracy of GPIA against the GNN with defense. We measure *target model accuracy* as the accuracy of node classification by the target model.

Baseline. Differential privacy (DP) [13] has been shown as effective against inference attacks on ML models [23, 34]. Therefore, we use differentially private deep learning method [6] that adds Laplace noise to the gradients as the baseline. We set the noise scale $\epsilon = \frac{1}{b}$ (i.e., $\epsilon = \{10,5,1,0.5,0.1\}$), where b is the noise scale value for



(a) Attack A_1^1 (b) Attack A_1^2 (c) Attack A_2

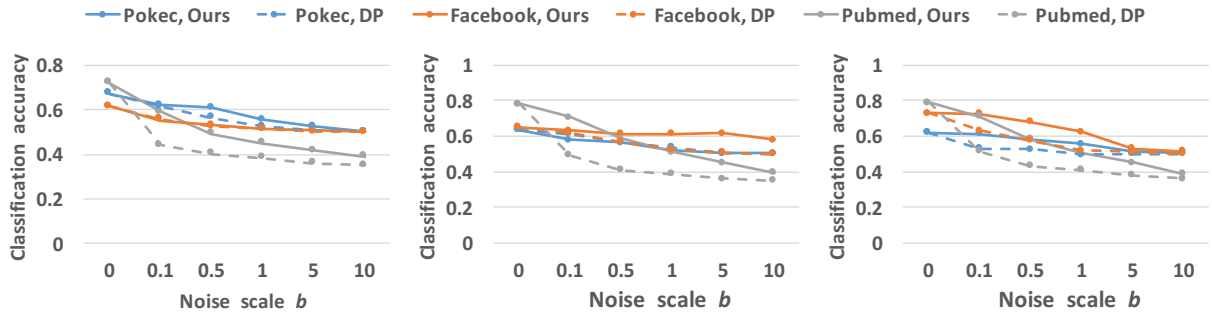
Figure 6: Defense effectiveness of the noisy posterior/embedding defense method (GCN as the target model). The noisy embedding defense is used against A_1^1 and A_1^2, while the noisy posterior defense is used against A_2.



(a) Attack A_1^1 (b) Attack A_1^2 (c) Truncate embedding Z^1 (d) Truncate embedding Z^2

Attack accuracy **Target model accuracy**

Figure 7: Performance of the embedding truncation defense (GCN as the target model).



(a) Noise added to Z^1 (b) Noise added to Z^2 (c) Noise added to posteriors

Figure 8: Target model accuracy under the noisy posterior/embedding defense (GCN as the target model).

the noisy embedding/posterior scheme. Lower ϵ indicates stronger noise scale and thus strong privacy protection.

Effectiveness of defense. We add the noise to the embeddings Z^1 and Z^2 to defend against A_1^1 and A_1^2. We do not consider adding noise to both layers as more noise will lead to higher loss of target model accuracy. Figure 6 shows the attack accuracy results under the noisy embedding/posterior defense when GCN is the target model. The defense performance on GraphSAGE and GAT are shown in the full version [40]. First, we observe that the noisy embedding/posterior defense can reduce the attack accuracy effectively by achieving the accuracy around 0.5 in all the settings. Second, different attacks require different amounts of noise to achieve the same degree of protection. For example, the defense against A_1^1 and A_1^2 requires the noise $b = 0.5$ and $b = 5$ to reduce the attack accuracy to be close to 0.5 respectively. Different properties also require different amounts of noise to be added to achieve the same degree of protection. For example, the defense against A_1^2 on P_2 and P_5 requires noise of scale $b = 5$ and $b = 0.5$ to reach the attack accuracy around 0.5.

We also observe that the defense power of the DP baseline is weaker than our method in the defense against the white-box attack A_1 (Figure 6 (a) & (b)), even with the noise scale as large as 10 (i.e., DP noise scale $\epsilon = 0.1$). Indeed, similar observations have been made that DP is ineffective against GPIA for other target models such as HMMs and SVMs [8]. Indeed, although DP provides theoretical guarantee of privacy protection against *individual data*

points, it is unclear if it can provide sufficient protection over GPIA inference of aggregate information of a group of samples. Further, the data independence assumption of DP [27] is indeed violated in the context of GNNs, as the edges in graph are dependent and correlated. However, we also observe that DP is effective against the black-box attack A_2 (Figure 6 (c)), and outperforms our method when the noise scale $b < 5$ in most of the settings.

Figure 7 (a) and (b) demonstrate the effectiveness of the embedding truncation defense method with GCN as the target model. The defense performance on GraphSAGE and GAT are shown in the full version [40]. The embedding truncation defense can reduce the attack accuracy of A_1^1 to be close to 0.5 when the truncation ratio is as small as 0.1 (i.e., remove 10% of embeddings). However, it requires more noise (truncation ratio as large as 0.3) to reduce the attack accuracy of A_1^2 to be close to 0.5. We believe this is because A_1^2 is stronger than A_1^1 as it encodes more information in the embedding that can be utilized by the attack.

Target model accuracy under defense. For the noisy posterior/embedding defense mechanism, we show the result of GCN accuracy in Figure 8. The results of GraphSAGE and GAT are included in the full version [40]. We observe that GCN accuracy downgrades when more noise is added to the embeddings/posteriors. The accuracy loss varies for different datasets. For example, the accuracy loss never exceeds 10% for Facebook when the noise is added to Z^2, but becomes as large as 50.7% for Pubmed dataset (Figure 8 (b)). Indeed, Pubmed dataset is the most sensitive to the noise among the three datasets, as it witnesses the largest amounts of accuracy loss. Nevertheless, the target model accuracy is acceptable when the defense is sufficient (i.e., the attack accuracy is close to 0.5). For example, consider Pokec dataset and noise scale $b = 1$, the accuracy of attack A_1^1 is mitigated to around 0.5 (Figure 6 (a)), while the target model accuracy is still 0.6 (Figure 8 (a)), which is higher than random guess for a binary classification task. We also observe that the target model accuracy by our defense always outperforms that of DP baseline. This demonstrates that adding noise on embeddings and posteriors better address the trade-off between defense and target model accuracy than adding noise on gradients.

For the embedding truncation defense, we show the result of GCN in Figure 7 (c) & (d). The results of GraphSAGE and GAT are shown in the full version [40]. We observe that the target model accuracy downgrades when the truncation ratio increases, and the accuracy loss varies for different datasets and different embeddings that are truncated. For example, the target model accuracy loss is 13.9% and 36.1% for Facebook and Pubmed datasets respectively when Z^1 is truncated. Pubmed dataset witnesses the highest accuracy loss among the three datasets. Second, in terms of the trade-off between privacy and accuracy, the embedding truncation method loses to the noisy embedding method, as its target model accuracy is lower than that by the noisy embedding method under similar attack accuracy. For example, by the embedding truncation method, the target model accuracy is 0.58 for Facebook dataset (Figure 7 (d)) when the attack accuracy of A_1^2 against all properties becomes around 0.5 (Figure 7 (b)). This is slightly lower than that by the noisy embedding method, where the target model accuracy is 0.61 for Facebook dataset (noise scale $b = 5$ in Figure 8 (b)) when the attack accuracy of A_1^2 against all properties becomes around 0.5 (noise scale $b = 5$ in Figure 6 (b)).

Why are defense mechanisms effective? Intuitively, our defense mechanisms add perturbations on individual embeddings and posteriors. Then why can they defend against the property inference at group level? To answer this question, we recall that one of root causes of GPIA is the disparate model loss across different groups (Section 6). We measure the loss of all groups after the defense mechanisms are applied, and observe that the loss disparity across different groups is mitigated to some extent by the perturbation added to embeddings/posteriors. There is more mitigation of loss disparity when more perturbation is added (i.e., stronger defense). More details of how loss disparity across different groups is mitigated by defense can be found in the full version [40].

8 CONCLUSION

In this paper, we propose the first systematic study of GPIA against GNNs. We design six GPIA attacks for both white-box and black-box settings, and demonstrate the attack effectiveness through extensive experiments. We analyze the main factors that contribute to the success of GPIA. We also present various defense mechanisms against the proposed attacks, and demonstrate the effectiveness of these mechanisms.

Limitations and future work. Next, we discuss the limitations of our work and several research directions for the future work.

Properties at subgraph level. So far, we only consider the group properties at node and link levels. In general, the properties at subgraph level, e.g., imbalanced data distribution across different communities, are sensitive. Thus an interesting direction for the future research is to extend our GPIA model to deal with subgraph-level properties. The design strategy of the subgraph-based PIAs can be similar to GPIA: we generate the training data from positive and negative shadow/partial graphs, and train the classifier on the generated data.

Non-binary properties. Our attacks only deal with binary properties. A more powerful attack can be predicting from multiple classes, for example, inferring the population ratio of particular racial groups in the given graph. One straightforward solution to non-binary properties is simply replacing the binary GPIA classifiers with multi-class ones using the same GPIA features. An alternative solution is to use meta-classifiers [8, 18] for inference.

Sparse graphs. The success of GPIA relies on its training data that consists of sufficient number of positive graphs sampled from shadow/partial graphs. This may not be achievable on shadow/target graphs that are sparse, as their samples are likely to contain few links and thus fail to meet the properties, especially the link-level ones. In this case, the adversary may need to apply link prediction algorithms [7, 28] to add links to the shadow/partial graphs in the GPIA training data.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their feedback. This project was supported by the National Science Foundation (#CNS-2029038; #CNS-2135988). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agency.

REFERENCES

[1] Amazon aws. https://aws.amazon.com/marketplace/solutions/machine-learning.
[2] Bigml inc. https://bigml.com/.
[3] Caffe model zoo. https://caffe.berkeleyvision.org/model_zoo.html.
[4] Google cloud. https://www.googleadservices.com/.
[5] Modzy: Ai model marketplace. https://www.modzy.com/marketplace/.
[6] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, 2016.
[7] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805, 2006.
[8] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3):137–150, 2015.
[9] Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun. Ask the locals: multi-way local pooling for image recognition. In *Proceedings of the International Conference on Computer Vision*, pages 2651–2658, 2011.
[10] Melissa Chase, Esha Ghosh, and Saeed Mahloujifar. Property inference from poisoning. *arXiv preprint arXiv:2101.11073*, 2021.
[11] Alexandra Chouldechova and Aaron Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.
[12] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *Proceedings of 17th EAI International Conference on Mobile and Ubiquitous Systems*, pages 76–85, 2020.
[13] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.
[14] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.
[15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.
[16] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *Proceedings of 23rd USENIX Security Symposium*, pages 17–32, 2014.
[17] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 69–78, 2020.
[18] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 619–633, 2018.
[19] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the Conference on Neural Information Processing Systems*, 2018.
[20] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *Proceedings of the 30th USENIX Security Symposium*, pages 2669–2686, 2021.
[21] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.
[22] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
[23] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *Proceedings of the 28th USENIX Security Symposium*, 2019.
[24] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.
[25] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of International Conference on Machine Learning*, pages 1885–1894, 2017.
[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[27] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnberable: Differential privacy under dependent tuples. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, volume 16, pages 21–24, 2016.
[28] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.
[29] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 691–706, 2019.
[30] Thomas Minka. Automatic choice of dimensionality for pca. *Advances in neural information processing systems*, 13, 2000.
[31] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 634–646, 2018.
[32] Mathias P. M. Parisot, Balazs Pejo, and Dayana Spagnuelo. Property inference attacks on convolutional neural networks: Influence and implications of target model's complexity. In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT)*, pages 2687–2704, 2021.
[33] Dana Pessach and Erez Shmueli. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3):1–44, 2022.
[34] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, pages 3–18, 2017.
[35] Congzheng Song and Ananth Raghunathan. Information leakage in embedding models. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 377–390, 2020.
[36] Anshuman Suri and David Evans. Formalizing and estimating distribution inference risks. *ICML Workshop on Theory and Practice of Differential Privacy*, 2021.
[37] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
[38] Petar Velivcković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of International Conference on Learning Representations*, 2018.
[39] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. Eavesdrop the composition proportion of training labels in federated learning. *arXiv preprint arXiv:1910.06044*, 2019.
[40] Xiuling Wang and Wendy Hui Wang. Full paper: Group property inference attacks against graph neural networks. https://arxiv.org/abs/2209.01100, 2022.
[41] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, pages 2512–2520, 2019.
[42] Zhihao Wen, Yuan Fang, and Zemin Liu. Meta-inductive node classification across graphs. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1219–1228, 2021.
[43] Juyang Weng, Narendra Ahuja, and Thomas S Huang. Cresceptron: a self-organizing neural network which grows adaptively. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, volume 1, pages 576–581, 1992.
[44] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Model extraction attacks on graph neural networks: Taxonomy and realization. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (ASIACCS)*, 2021.
[45] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, 2022.
[46] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *Proceedings of the 29th IEEE Computer Security Foundations Symposium (CSF)*, pages 355–370, 2016.
[47] Wanrong Zhang, Shruti Tople, and Olga Ohrimenko. Leakage of dataset properties in multi-party machine learning. In *Proceedings of the 30th USENIX Security Symposium*, pages 2687–2704, 2021.
[48] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *Proceedings of the 31th USENIX Security Symposium*, pages 1–18, 2022.
[49] Junhao Zhou, Yufei Chen, Chao Shen, and Yang Zhang. Property inference attacks against gans. In *Proceedings of the 29th Network and Distributed System Security Symposium (NDSS)*, 2022.