Quantifying Privacy Leakage in Graph Embedding

Vasisht Duddu
Univ Lyon, INSA Lyon, Inria, CITI
vduddu@tutamail.com

Antoine Boutet
Univ Lyon, INSA Lyon, Inria, CITI
antoine.boutet@insa-lyon.fr

Virat Shejwalkar
University of Massachusetts
Amherst
vshejwalkar@cs.umass.edu

ABSTRACT

Graph embeddings have been proposed to map graph data to low dimensional space for downstream processing (e.g., node classification or link prediction). With the increasing collection of personal data, graph embeddings can be trained on private and sensitive data. For the first time, we quantify the privacy leakage in graph embeddings through three inference attacks targeting Graph Neural Networks. We propose a membership inference attack to infer whether a graph node corresponding to individual user's data was member of the model's training or not. We consider a blackbox setting where the adversary exploits the output prediction scores, and a whitebox setting where the adversary has also access to the released node embeddings. This attack provides an accuracy up to 28% (blackbox) 36% (whitebox) beyond random guess by exploiting the distinguishable footprint between train and test data records left by the graph embedding. We propose a Graph Reconstruction attack where the adversary aims to reconstruct the target graph given the corresponding graph embeddings. Here, the adversary can reconstruct the graph with more than 80% of accuracy and link inference between two nodes around 30% more confidence than a random guess. We then propose an attribute inference attack where the adversary aims to infer a sensitive attribute. We show that graph embeddings are strongly correlated to node attributes letting the adversary inferring sensitive information (e.g., gender or location).

CCS CONCEPTS

• **Security and privacy** → **Privacy protections**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Privacy Leakage, Inference Attacks, Graph Neural Networks, Graph Embeddings.

MobiQuitous '20, November 7–9, 2020, N/A, Cyberspace

ACM ISBN . . . $
https://doi.org/

ACM Reference Format:
Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying Privacy Leakage in Graph Embedding. In *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '20), November 7–9, 2020, N/A, Cyberspace.* ACM, New York, NY, USA, 11 pages. https://doi.org/

1 INTRODUCTION

Large scale real-world systems are typically modelled in the form of graphs: mobility, network infrastructures, online social networks, world wide web, citation networks and biomedical datasets, which represent the entities as nodes and their relationship with edges [41]. Traditional Deep Neural Networks fail to capture the nuances of structured data but a specific class of algorithms, namely, Graph Neural Networks (GNNs) have shown state of the art performance on such complex graph data for node classification, link prediction etc. An important pre-processing step for using graph data with machine learning is embedding the high dimensional graph data to a low dimensional representation for easy processing by machine learning algorithms. In many applications, such embeddings are released for further processing to save storage cost without considering the privacy implications. Such large graph dataset raises the question of privacy specifically if the algorithms are trained with private and potentially sensitive data. Consider a graph capturing the outbreak of a disease where the nodes represent the individuals, medical symptoms as the node features and the edges indicating the disease transmission. Typically, in such datasets a GNN provides state of the art performance for predicting disease for an arbitrary user in the graph (node classification) and determining the future outbreak (link prediction). For such embedding models which do not account privacy, an adversary can however infer the health status of a particular user (node in graph) by identifying whether the user was part of the training data or not. Further, the adversary can potentially reconstruct the sensitive graph input from the low dimensional embeddings. Finally, graph embeddings capture important semantics from the input graph while maintaining the contextual information in the form of preferential connection which can be exploited to infer sensitive attributes about an individual. These three privacy attacks, namely, membership inference, graph reconstruction and attribute inference, are examples of a direct privacy violation of the individual which can further be used without user consent. Further, companies spend enormous resources to annotate the training dataset to achieve state of the art performance

and such attacks inferring training data also violates the Intellectual Property.

In the context of Machine Learning (ML), a privacy violation occurs when an adversary infers something about a *particular* user's data record in the training dataset which cannot be inferred from other models trained on similar data distribution [27, 31]. This information leakage is quantified using the success of inference attacks. In attribute inference attacks, the attacker infers sensitive features of an individual's data record used in model's training. A stronger case of attribute inference is where the attacker reconstructs a portion of the sensitive training data itself, i.e, data reconstruction attack. In case of membership inference, the adversary traces a particular individual's record to the training dataset, i.e., identify whether a given data record was a member of the training data. Prior literature in privacy attacks focus on models trained on non-graph data including text, images and speech to study the vulnerability to membership inference [30, 31], attribute inference [2, 12], property inference [11], model inversion [10] attacks as well as model parameter and hyperparameter stealing attacks [8, 9, 38]. While well studied in traditional ML, the privacy risk in graph-based ML models under adversarial setting has not been fully explored and quantified.

In this work, we propose the first comprehensive privacy analysis of Graph Embedding algorithms under different threat models and adversary assumptions. We mainly focus on exploiting publicly released graph embeddings trained with private data, used for different downstream tasks, under various attacks which violates the user's data privacy: membership inference, graph reconstruction and attribute inference. First, we evaluate the privacy leakage under membership inference attacks though a blackbox and whitebox settings. The blackbox setting considers the specific case of downstream node classification task for convolution kernel based graph embedding with neural network. In this setting, we propose two attacks for membership inference: with auxilary knowledge on the data distribution (shadow model attack) and without auxilary knowledge (confidence score attack). Here, we show that the proposed attacks have an inference accuracy of 78%, 63%, and 60% for confidence score attack and 62%, 60%, and 55% for shadow model attack, respectively for three different datasets, i.e., Cora, Citesser and Pubmed dataset. For the whitebox setting, we propose an unsupervised attack for the more generic case of using just the graph embeddings to differentiate whether a given node was part of the training graph or not. We show that an adversary in this setting can predict the training data with a high accuracy (70% on average on the three datasets).

Second, we propose a novel graph reconstruction attack where the adversary, given access to the node embeddings of a subgraph, trains an encoder-decoder model to reconstruct the target graph from its publicly released embeddings. This attack has serious privacy implications since the adversary reconstructs the input graph dataset which can be potentially sensitive. This attack has high precision: 0.722 for Cora, 0.778 for Citeseer and 0.95 for Pubmed dataset. Moreover, on increasing the adversary's prior knowledge, the attack performance increases significantly. An important privacy implication is link inference (i.e, predicting a link between any two nodes in the graph). Through this attack, an adversary infers a link between nodes with 80% of accuracy on average, compared to the 50% baseline random guess accuracy.

Finally, we propose the attribute inference attack where the adversary tries to infer sensitive attributes for user node in the graph using the released graph embeddings. We consider two state of the art unsupervised random walk based embeddings, Node2Vec and DeepWalk, on two real-world social networking datasets: Facebook and LastFM, where the adversary aims to infer the user gender and location, respectively. Given access to the embeddings of a subgraph and corresponding sensitive attributes, we model attribute inference as a supervised learning problem. The adversary trains a supervised attack model to predict sensitive hidden attributes for target users given the released publicly available target embeddings. Here, the attack model's F1 score (capturing the balance between precision and recall) on LastFM was as high as 0.65 for DeepWalk and 0.83 for Node2Vec. For Facebook, the F1 score was 0.61 for Node2Vec and 0.59 for DeepWalk. The paper indicates the serious data privacy risks in graph data processing algorithms and calls for further research to design privacy-preserving embedding algorithms for graph data. The code for all the experiments is made publicly available for easy reproducibility[1].

The paper is organized as follows. Backgrounds on graph embedding algorithms and GNNs are introduced Section 2, followed by the considered threat models Section 3. Experimental setup is described Section 4. Evaluations of the proposed attacks are then given Section 5 before related work Section 6 Finally, concluding remarks are given Section 7.

2 BACKGROUND

A large number of real-world applications require processing graph data which contains rich relational information between different entities (e.g., online social media, disease outbreaks, recommendation engines, knowledge graphs and navigation systems) [41]. Deep Learning and more precisely Convolutional Neural Networks have shown tremendous performance over non-graph data such as images by capturing the spatial relation between pixels of image and extracting features over multiple layers. However, this machine learning scheme has shown its limits for graph data and the learning on such data is still challenging [41]. Indeed, the models have to capture the connections in the data while ensuring invariance of graph data representation, even without fixed ordering between the nodes (i.e., the adjacency matrix representing the connections between nodes varies but still results in the same graph). To overcome this limitation, the graph data is passed through embedding algorithms which map the large graphs to lower dimensions which are then used for downstream processing with GNNs. Graph embedding algorithms enable models operating on low dimensional euclidean

[1] https://github.com/vasishtduddu/GraphLeaks

datasets (i.e., such as images) to graph data by mapping them into a low dimensional embedding. We represent a graph as $G = (V, E)$ where V represents the vertex set consisting of nodes $\{v_1, ..., v_n\}$ where the connections between the edges E is represented as a symmetric, sparse adjacency matrix $A \in R_{nxn}$ where a_{ij} denotes the edge weight between nodes with $a_{ij} = 0$ for missing edges.

2.1 Graph Embedding Algorithms

In order to mitigate the space and computation overhead of downstream graph processing, graph embedding algorithms provide an efficient approach to represent the graph data in a low dimension embedding [6]. Specifically, an embedding algorithm $\Psi : V \rightarrow \mathcal{R}^d$ where nodes $V \in G$. The embedding $\Psi(v)$ of a single node v is hence an d-dimensional vector capturing the properties of the original graph such as distance from other nodes. Different graph embedding algorithms to embed both the entire graphs as well as the nodes have been well studied [4]. Random walk based node embedding algorithms traverse the graph to sample random walks which are then passed as sentences to SkipGram algorithm to obtain the corresponding node features [14, 29]. In deep learning based graph embeddings, both the features along with adjacency matrix can be used to generate low dimensional embeddings for each of the nodes. For generating these embeddings, the parameters of the embedding function are updated to improve the representation of the graph nodes while maintaining the original properties. These are typically modelled using GNNs and Graph Convolutional Networks. In this work, we mainly focus on node embeddings which we refer as graph embeddings throughout the paper. We consider random walk based embeddings for attribute inference and deep learning based embeddings for node inference and graph reconstruction attacks.

2.2 Graph Neural Networks

The initial layers of a GNN are used to generate embedding for the input graphs which can be extended for node classification and link prediction tasks by attacking a classifier network as GNNs. The pre-processed graph in the form of embeddings along with the node features are represented as matrices for computation. The training of GNNs relies on a message passing algorithm which is the weighted aggregation of features of neighbouring nodes $\mathcal{N}(v)$ to compute the feature of a particular node v. Given the features x of a single node v, the GNN produces an output label $f(x; W)$ which captures the probability of the input node with features x belonging to a particular class. The loss over the resultant classification for the node v is then backpropagated to update the model weights for aggregation. Consider a $N \times D_F$ feature matrix X where N is the number of nodes, D_F is the number of node features and an adjacency matrix A which captures the representation of graph structure in matrix form. The output of a layer with F features takes the feature matrix along with the adjacency matrix as input to produce a $N \times F$ matrix as an output. The computation is given by $H^{(l+1)} = f_{agg}(H^{(l)}, A)$ with $H(0) = X$ and $H(L) = Z$, L being the number of layers and H is the intermediate activation. Based on the different aggregation function $f_{agg}()$, we obtain different GNN algorithms such as Graph Convolutional Network (GCN) [21], GraphSAGE [15], Graph Attention Networks (GAT) [37] and Topology Adaptive GCN (TAGCN) [7].

3 THREAT MODELS

In this section, we describe the considered threat models as well as the methodology and adversary assumptions.

3.1 Membership Inference Attacks

Membership Inference attacks allow personal information leakage in GNNs. Specifically, the goal of the adversary is to identify whether a user node v is part of the graph G_{train} used for training the target model. This is a binary classification problem where the adversary learns the threshold to predict the membership of a user node. Depending on the adversary's knowledge about $f()$, we consider two settings: blackbox (with and without auxiliary knowledge) and whitebox. As shown Figure 1, to distinguish between members and non-members of training graph G_{train}, the blackbox attacks exploit the statistical difference in output predictions while the whitebox attack exploits the intermediate low dimensional embedding.

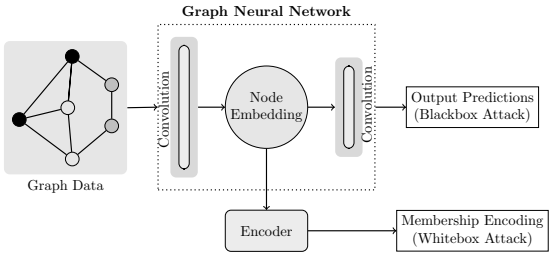
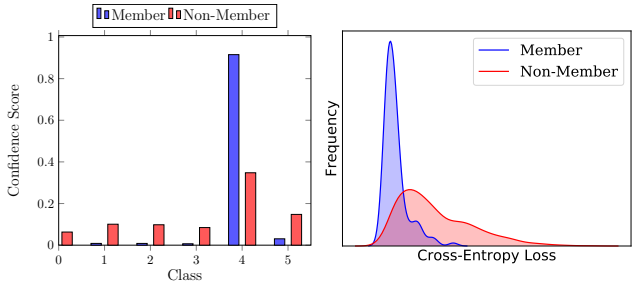


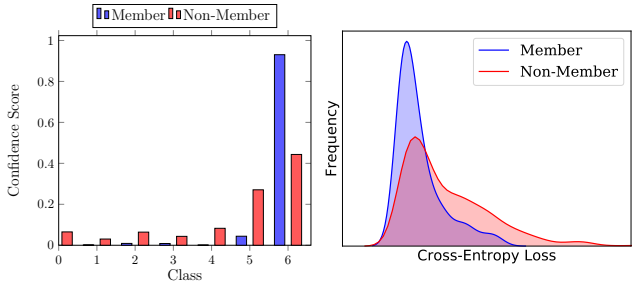
Figure 1: Blackbox and whitebox inference attacks to distinguish members and non-members of G_{train}.

3.1.1 Blackbox: Inference using Output Predictions. In this setting, we consider the target model as trained GNN for node classification task. The adversary aims to infer whether a user's node in the graph was used in training the target model $f()$. In a blackbox setting, adversary has only access to the model outputs $f(x; W)$ for a given input x. The parameters of the trained model W as well as the intermediate computation are inaccessible to the adversary. This is a practical setting, typically seen in the case of Machine Learning as a Service, where a trained model is deployed in the cloud and the adversary queries the model through an API and receives corresponding predictions.

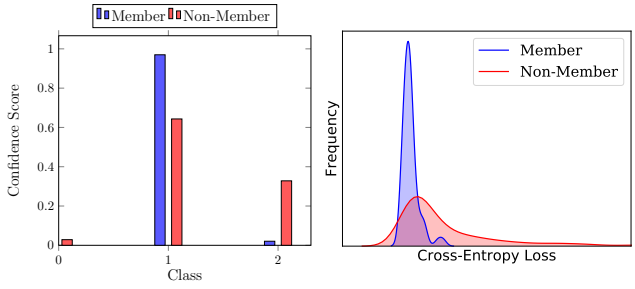
Blackbox adversary exploits the statistical difference between the confidence in prediction on training and testing data [31]. Figure 2 (left) illustrates this difference where the prediction confidence for one class is much higher for training data points. Predicting with higher confidence on seen G_{train} nodes compared to unseen test nodes is referred as overfitting. This difference in the output prediction confidence



(a) Citeseer



(b) Cora



(c) Pubmed

Figure 2: Model predictions are more confident for nodes in G_{train} compared to test graph (left). The extent of overfitting can be detected by a non-overlapping region between the output prediction distributions across all data points (right).

directly results from a distinguishable output distribution between train and test data indicated by non-overlapping region between distributions (Figure 2, right).

We consider two attacks within the blackbox setting: (a) shadow model attack and (b) confidence score attack.

Shadow Attack: The shadow attack relies on the adversary training a local substitute model with similar functionality as the target model to identify characteristics of members and non-members. The adversary has knowledge about the target GNN architecture and auxiliary graph dataset G_{aux} sampled from the same underlying data distribution as G_{train} which is consistent with prior attack settings [2, 12, 16, 31] but the attack is transferable across different models [30]. This is a practical assumption where social networks have publicly available API enabling the adversary to obtain subgraphs of the original social network graph. To conduct its attack, the adversary uses prior knowledge to map the target model's

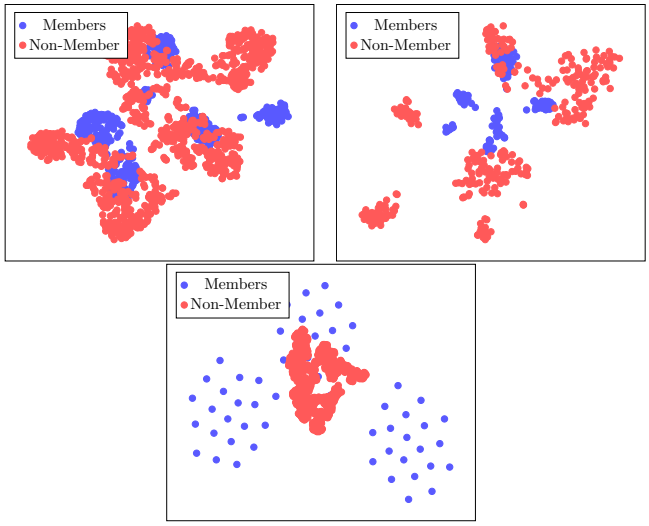


Figure 3: Whitebox membership inference attacks exploit the distinguishable intermediate embedding of train and test graph nodes for Citeseer (left), Cora (middle) and Pubmed dataset (right).

predictions to membership values and hence the attack is supervised. For a target model $f()$, the adversary trains a substitute model f_{local} on auxiliary graph data (G_{aux}) drawn from the same distribution as G_{train}. The datasets are assumed to be non-overlapping, i.e, $G_{train} \cap G_{aux} = \phi$, which makes the attack more practical. The goal is to train f_{local} to mimic the behaviour of $f()$, i.e, the output predictions should be similar to each other $f_{local}(v; W') \sim f(v; W)$ for the same input user node v but different parameters W' and W due to training on the different data. Given the substitute model, the adversary creates a synthetic dataset with binary classes for distinguishing members and non-members (encoded as class 1 and class 0) of f_{local}'s training data G_{aux} while using the output predictions as the input features. That is, the synthetic dataset has the input as f_{local}'s predictions for an user node v classified as "Member" if $v \in G_{aux}$ and "Non-Member" otherwise. Hence, f_{local} is used as a proxy for $f()$ to learn the mapping between the $f()$'s output predictions and the membership information. The adversary trains a binary attack classifier f_{attack} on the synthetic dataset used to predict whether a new user node was member of G_{train}.

Confidence Attack: In this particular case, we alleviate the assumption of adversary knowledge about data distribution and target model architecture as part of shadow model making the attack applicable to a wide range of practical scenarios. Since, the adversary does not have prior knowledge to map the output predictions of target model to classify the membership, the attack is performed in an unsupervised setting. To conduct its attack, the adversary leverages the fact that graph nodes with higher output confidence prediction are likely to be members of G_{train}. Here, the adversary finds the output prediction with highest confidence and compares whether this is above a certain threshold to decide

whether the corresponding graph node was in the model's training graph G_{train} or not. A large output confidence indicates membership of the data point in the training data. The adversary sweeps across different values to select the threshold value which best suits the application. Prior work in traditional ML has indicated that the confidence attack is much better compared to shadow model attack as the which is verified in our experiments [35]. The signal used to distinguish members from non-members from confidence score attacks is directly from the target model which the shadow model is subtle and uses a substitute model's output as a signal. The attack success in this case is reliant on the quality of auxiliary data for training local substitute model and its functional similarity with the target model.

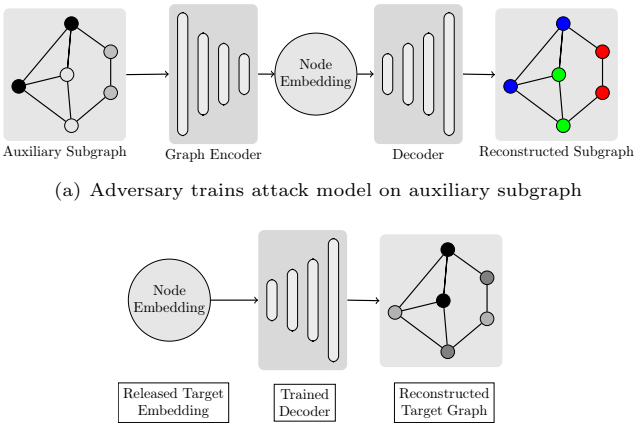
3.1.2 Whitebox: Inference using Graph Embedding. The adversary in a whitebox setting has access to the model output predictions $f(x;W)$ for an input x as well as the model parameters W. This allows the adversary to compute the intermediate computations after each layer which in our case corresponds to the embedding for each node as an output of graph convolutional layer. This is a strong adversary assumption but practical in the case of federated learning where intermediate computations can be observed [24, 27].

As explained Section 2, GNNs compute the low dimensional embedding for the input graph data. The parameters of GNNs are updated in each iteration of training and are tuned specifically for high performance on the train data resulting in a distinguishable footprint between embedding of train and test data points. Figure 3 illustrates this rationale by plotting embedding of train and test graph nodes after a dimension reduction using 2D-TSNE algorithm [36].

The attack is unsupervised as the adversary has no prior knowledge to map the intermediate embeddings to a membership value. The adversary trains an encoder-decoder network in unsupervised fashion to map the intermediate embedding to a single membership value. For an input graph node's embedding $\Psi(v)$, encoder $f_{enc}()$ generates a scalar membership value which is passed to decoder $f_{dec}(f_{enc}(\Psi(v)))$ to obtain $\Psi(v)$ by minimizing reconstruction loss: $||\Psi(v) - f_{dec}(f_{enc}(\Psi(v)))||_2^2$. Given the membership values for different training and testing data points, K-Means clustering is used to cluster nodes into two classes (members and non-members). For any new user node, the adversary then clusters as members or non-members of the training data.

3.2 Graph Reconstruction Attack

Given a sensitive target graph data (G_{target}) and the corresponding set of publicly released embeddings, $\Psi(v)\ \forall v \in G_{target}$, the goal of the adversary in this attack is to reconstruct G_{target} and the corresponding connections between the different nodes A_{target}. Specifically, the goal of the adversary is to reconstruct the adjacency matrix A_{target} of the graph which is binary with $A_{ij} = 1$ if there exists an edge between the node i and j and zero otherwise. While node membership inference is a subtle privacy violation of user's data, this is a stronger attack where the entire sensitive graph data is reconstructed by the adversary.



(a) Adversary trains attack model on auxiliary subgraph

(b) Attack model reconstructs target graph

Figure 4: Attack methodology for graph reconstruction from released embeddings.

Graph embeddings are specially computed to ensure that the underlying graph properties do not change. In other words, the embeddings capture the rich semantic, invariant and structural information about the graph, for instance, by preserving proximity to the neighbouring nodes. Hence, there exists a strong correlation between the released graph embeddings and the actual graph which can be exploited to reconstruct the graph data.

The adversary is assumed to have knowledge of the auxiliary subgraph G_{aux} which is sampled from the same distribution as the target graph G_{target}. Empirically, this is obtained by sampling two non-overlapping subgraphs from the full graph dataset. The adversary performs graph reconstruction in two phases (Figure 4). In Phase I, the adversary trains a graph encoder-decoder attack model on G_{aux}. The graph encoder f_{enc} maps the adjacency matrix of G_{aux} to corresponding node embeddings $\Psi(v) \rightarrow f_{enc}(v)\ \forall\ v \in V$ represented as adjacency matrix. The decoder f_{dec} reconstructs the adjacency matrix $A_{rec} = f_{dec}(\Psi(v))$ while both the models are trained using backpropagation to minimize reconstruction loss: $||A - A_{rec}||_2^2$. For the attack model, we consider an architecture with graph convolution as encoder and a decoder which computes the dot product between the embedding vector $\Psi(v)$ and its transpose $\Psi^T(v)$ [21]. The attack models are trained on G_{aux} and tested on the target embeddings corresponding to target graph G_{target} to reconstruct the graph data. Given the target released embeddings, the adversary then uses the trained decoder attack model to map the released embeddings to the target adjacency matrix $A^{rec}_{target} = f_{dec}(\Psi(v'))\ \forall\ v' \in G_{target}$.

Link inference: Link inference is a direct result from the graph reconstruction attack where the adversary can check for an edge between two users using the adjacency matrix of the reconstructed graph. This inference is a binary classification

problem where the adversary aims to infer whether there exists a link between two nodes in the graph. This inference represents the knowledge that two people know each other in online social networks for instance leading to identify the friendship circle of users which is a privacy violation of individuals. More formally, given two user nodes i and j, the adversary queries the reconstructed adjacency matrix A^{rec}_{target} to infer whether there exists a link between ij (if $A^{rec}_{target}[i][j] = 1$) or not (if $A^{rec}_{target}[i][j] = 0$). The success of link inference attack closely depends on the success of reconstructing the target adjacency matrix $A^{rec}_{target} \sim A_{target}$.

3.3 Attribute Inference Attack

While the previous attack focussed on sensitive graph data and inference attacks exploiting connections between different nodes, attribute inference attacks exploit user node's sensitive features. Particularly, given the embedding of the subgraph nodes and corresponding sensitive attribute $(\Psi(v), s_v)\ \forall v \in G_{aux}$, the adversary aims to infer the sensitive attributes s^* corresponding to the publicly released target embeddings $\Psi(v')\ \forall v' \in G_{target}$. This is a practical assumption as a small fraction of users indeed make their information publicly available on their profile while other users prefer to keep such information private such as gender and location.

Nodes in graphs for most practical real world applications follow preferential connections, i.e, nodes similar to each other are connected to each other. This is particularly true in case of social networks where users with similar likes and preferences, represented as features for nodes in the graph, are connected together [13, 20]. This feature similarity and preferential connections in graphs are captured by graph embeddings to preserve the graph properties. Hence, the embeddings are strongly correlated with the node features which can be exploited to infer sensitive attributes.

The adversary has access to the node embeddings and corresponding node's sensitive attributes $(\Psi(v), s_v)\ \forall v \in G_{aux}$ from the auxiliary subgraph known to the adversary. The adversary uses this prior knowledge to train a supervised attack classifier f_{attack} which maps the embedding to sensitive attributes, i.e, $f_{attack} : \Psi(v) \rightarrow s_v$. Using this trained attack model, the adversary infers the sensitive attribute s^* corresponding to the target embeddings $f_{attack}(\Psi(v'))$ where $v' \in G_{target}$.

4 EXPERIMENT SETUP

In this section, we present the considered datasets, embedding algorithms, evaluation metrics and the methodology.

4.1 Datasets

For the membership inference and graph reconstruction attack, we consider three standard benchmarking datasets: Pubmed, Citeseer and Cora. For the attribute inference attack, in turn, we consider two social networking datasets with anonymized sensitive attributes: Facebook[2] and LastFM[3].

[2]http://snap.stanford.edu/data/ego-Facebook.html
[3]http://snap.stanford.edu/data/feather-lastfm-social.html

Pubmed. The Pubmed Diabetes dataset consists of 19,717 scientific publications from PubMed database pertaining to diabetes classified into one of three classes. The citation network consists of 44,338 links. Each publication in the dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words. We use 60 train samples, 500 validation samples and 1,000 test samples.
Citeseer. The CiteSeer dataset consists of 3,312 scientific publications classified into one of six classes. The citation network consists of 4,732 links. Each publication is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3,703 unique words. The number of training samples is 120, 500 validation samples and 1,000 test samples.
Cora. The Cora dataset consists of 2,708 scientific publications classified into one of seven classes. The citation network consists of 5,429 links. Each publication is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1,433 unique words. For training 140 samples are used, 300 validation samples and 1,000 test samples.
Facebook. The dataset comprises of 4,039 nodes representing different user accounts on the social network connected with each other through 88,234 edges. Each user node has different features including the gender, education, hometown etc. The user information has been anonymized through pseudonymization and the interpretation of the features have been obscured (i.e, attributes 'Male' and 'Female' have been replace with 'Gender 1' and 'Gender 2', respectively).
LastFM. The dataset was collected in March 2,020 using the public API provided by the social network specifically created for users from Asian countries. The dataset has 7,624 nodes connected together with 27,806 edges based on mutual follower relationships. Each user has attributes such as the music and artists they likes, location etc.

4.2 Embedding Algorithms

For the purpose of our experiments, we consider two classes of embedding algorithms: GNNs and random walk based. We consider the following GNN based embedding techniques:
Graph Convolutional Network (GCN) [21]. GCN computes the target node features from neighbouring nodes using matrix factorization, by normalizing adjacency matrix A as $D^{-1}A$ where D is the diagonal node degree matrix and results in averaging of neighbouring node features. An additional trick is to use a symmetric normalization as $D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}$.
GraphSAGE [15]. GraphSAGE extends the operations in GCN to more generic functions for transformation and aggregating of node features. While the embedding of graph data in GCN relies on matrix factorization, GraphSAGE uses node feature aggregation using mean, LSTM and pooling to learn the embedding function.
Graph Attention Networks (GAT) [37]. Weights associated with features during aggregation are explicitly defined and learnt during training. GAT implicitly defines the weights using self-attention mechanism over the node features.

Topology Adaptive GCN (TAGCN) [7]. Instead of using the spectral convolutions for learning non-linear graph data, TAGCN proposes to use general K-localized filter convolution in the vertex domain. It replaces the fixed square filters in traditional spectral GCNs for the grid-structured input data volumes.

For membership inference, we consider the embeddings from all the above four architectures for the whitebox setting while for the blackbox setting we consider only GraphSAGE algorithm as inductive training graphs models is challenging and GraphSAGE architecture is specifically designed to work in such training settings [15]. In case of graph reconstruction attacks, we consider the generic GCN model as the encoder for the attack model. In case of attribute inference attacks, we consider two state of the art unsupervised graph embedding algorithms based on random walk, namely, Node2Vec [14] and DeepWalk [29].

DeepWalk [29]. The algorithm creates a transition matrix from the graph and samples random walks from the matrix. The nodes are viewed as words and the random walks are viewed as sentences and the resulting sequences are passed to Word2Vec and SkipGram [25] to obtain node embeddings.

Node2Vec [14]. This is an extension of DeepWalk which combines Breadth First and Depth First search explorations on the graph to create biased random walks. The embeddings are computed using Word2Vec algorithm as mentioned above.

4.3 Metrics

To estimate the attack success of both membership and link inference, we consider the inference accuracy.

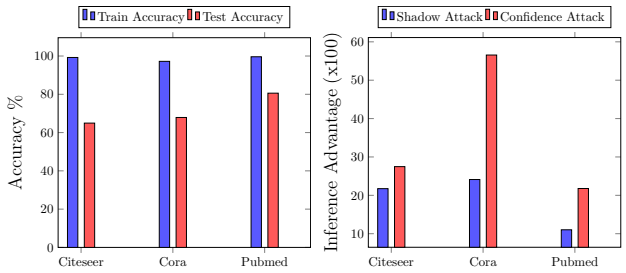
Inference Accuracy. Membership and link inference are binary classification problems: node is part of the training data or not (membership inference) and there exists a link between any two nodes or not (Link Inference). Hence, the accuracy of random guess is 50% and any higher accuracy indicates a privacy leakage about the model's sensitive training data. In order to compute the additional benefit the adversary gets in terms of performing the attack over random guess, we name 'adversary advantage' a metric computed as: $I_{adv} = 2 * (I_{acc} - 0.5)$. This metric estimates the information leakage from the model compared to a random guess.

For evaluating the performance of graph reconstruction attacks, we use two main metrics: precision and roc score.

Precision. The ratio of true positives is given by the precision and estimates the percentage of the predicted samples that are actually in the target graph.

ROC-AUC Score. The ROC curve plots the true positive rate on the y-axis and the false positive rate on the x-axis. The AUC score computes the area under the ROC curve to get how good the model distinguishes between different classes. For a binary classification problem of graph reconstruction to obtain the binary adjacency matrix, the random guess accuracy is 50% and any higher accuracy indicates the adversary's advantage in reconstructing the target graph.

In case of attribute inference attack, we evaluate using the F1 score to balance both the recall and precision.



(a) Generalization Error (b) Inference Advantage

Figure 5: Blackbox membership inference attack uses the output predictions to give adversary an inference advantage.

F1-Score. This metric computes the harmonic mean between the precision and recall which estimates the percentage of samples in the target graph which are predicted as such.

4.4 Methodology

In this work, we specifically focus on inductive training of GNN where the model does not see test nodes during training unlike transductive learning where the entire graph and features are available apriori. Given the full graph G_{full}, we sample a subgraph G_{train} which is used for training the models and evaluate the model performance on the held out graph $G_{full} - G_{train}$. Such an inductive setting enables the adversary to learn new information about the target model's training graph resulting in a privacy leakage.

5 EVALUATION

In this section, we evaluate the privacy leakage from attacks.

5.1 Membership Inference Attack from Output Predictions (Blackbox setting)

The overfitting for GraphSage architecture trained on the three datasets is given in Figure 5(a). We evaluate the two blackbox inference attacks exploiting the output predictions from the models: shadow inference which uses auxiliary knowledge on the data distribution and confidence inference which does not use auxiliary knowledge. Results depicted Figure 5(b) show that under confidence inference attacks, the inference accuracy is 78.28% (corresponding to an adversary's advantage of 27.48%), 63.75% (an adversary's advantage of 56.56%), and 60.89% (an advantage of 21.78%) for Cora, Citesser, and Pubmed datasets, respectively. In case of shadow model attacks, the inference accuracy for Cora is 62.06% (representing an adversary advantage of 21.74%), 60.87% for Citeseer (an adversary advantage of 24.12%), 55.51% for Pubmed dataset (an adversary advantage of 11.02%). Membership leakage is thus higher in confidence attack (i.e., without auxiliary knowledge) compared to shadow model attack (i.e., with auxiliary knowledge). While counterintuitive, this result is consistent with similar attack methodology for traditional machine learning models [35].

Impact of Increasing Number of Layers. We evaluate the performance of confidence attack on increasing the range of neighbourhood nodes used for aggregating the features (Figure 6). To do that, we extend the range of the message passing algorithm by increasing the number of layers in the GNNs [22, 23]. On increasing the number of layers from 2 to 6, the inference accuracy decreases by 8% for GCN. Interestingly, the generalization error increases (train accuracy remains the same but the test accuracy decreases) for Cora, Citeseer and Pubmed dataset, but the inference accuracy continues to decrease which indicates that the influence of preferential connections between different nodes in the graph plays a significant role in influencing the inference accuracy. For large number of layers (> 8 layers) in the GNN, for all datasets and architectures, the model completely loses its predictive power. In general, the inference accuracy as well as prediction accuracy decreases with increasing the range of the message passing algorithm by increasing the layers from 2 to 16. This implies that the membership privacy leakage is influenced by the structured graph data with preferential connections between different nodes. Specifically, aggregating features from larger number of nodes results in higher averaging which reduces the distinguishability (over-smoothening of features) as model converges to random walks limit distribution [22, 23] which is crucial for inference attacks [30, 31].

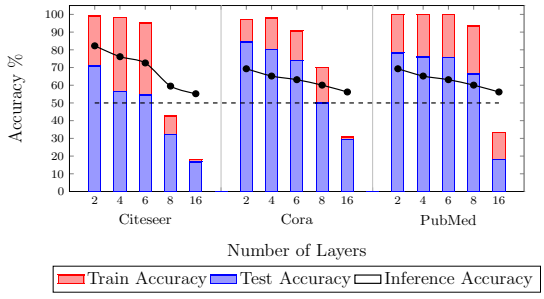


Figure 6: The inference accuracy and predictive power decreases on increasing the number of layers due to feature oversmoothening from nodes deeper in the graph.

5.2 Membership Inference Attack from Graph Embeddings (Whitebox setting)

We exploit the difference in intermediate feature representation of train and test data to perform membership inference attack in a whitebox setting. Results show that different models trained on PubMed dataset leak significantly more information between 20% and 36% over random guess accuracy. On the other hand, models trained on Citeseer dataset provide to the adversary an advantage between 7% and 17% over random guess while for Cora dataset it is between 4% and 7%. The embedding is significantly different for train and test data points for PubMed dataset as seen Figure 7 which result in a higher whitebox membership inference accuracy compared to Cora and Citeseer dataset. The higher

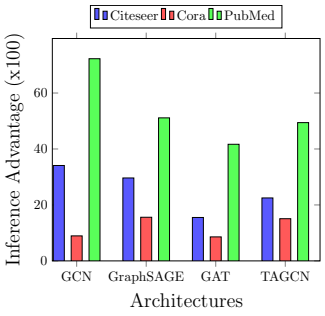


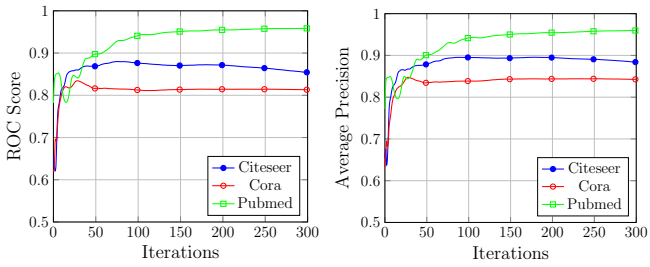
Figure 7: Adversary advantage for node membership inference from Graph Embeddings.

accuracy for Pubmed dataset can be attributed to significant distinguishability of features as seen by visually inspecting in Figure 3.

The success of the unsupervised whitebox attack is attributed to the message passing which updates the parameters (weights) to specifically ensure higher distinguishability between the data points of different classes for high accuracy on training dataset. Indeed, the parameters are specifically updated to fit the training dataset resulting in a high distinguishability between feature embedding of train and test data records. Moreover, the feature embedding for the initial layers are useful since for later layers the features are oversmoothened which also reduces accuracy (as seen in increasing the range of nodes of message passing algorithm).

5.3 Graph Reconstruction Attack

The success of graph reconstruction is evaluated on the unseen target graph while the model is trained on the train graph. The test AUC score for Cora dataset is 0.65 and the average precision is 0.722 while for Citeseer dataset the AUC score is 0.65 and 0.778 average precision. For Pubmed dataset, we get an average precision of 0.95 on the test set with an ROC of 0.94 in reconstructing the target test graph. The curve for variation of AUC score and the average precision for the three datasets on the validation sub graph is given in Figure 8.



(a) Validation ROC (b) Average Precision

Figure 8: Training curves for AUC score and Average Precision on the validation graph.

Impact of Adversary Knowledge. On increasing the adversary's knowledge to 50% of the target graph, we observe

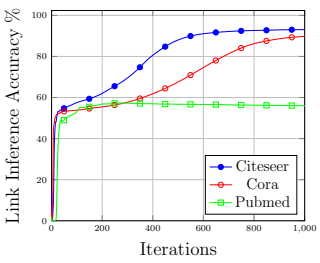
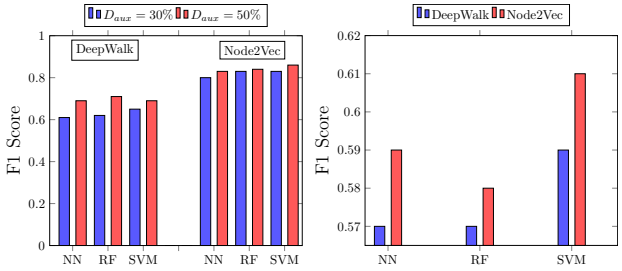


Figure 9: Link Inference Accuracy Curve over different epochs



(a) LastFM (b) Facebook

Figure 10: F1 score for different attack classifiers to infer sensitive attributes.

an increase the attack performance. Specifically, the AUC score for Cora increases to 0.76 from 0.65 while the average precision increases to 0.81 from 0.722. In Citeseer dataset, the AUC score increases to 0.779 from 0.65 while the average precision increases to 0.828 from 0.778. Finally, for Pubmed dataset showed an increase to 0.95 from 0.94 AUC score and 0.96 from 0.95 average precision.

5.4 Link Inference Attack

A direct implication of graph reconstruction attack is inferring whether there exists a link between two nodes in the network by querying the reconstructed adjacency matrix. This is a binary classification problem.

For Citeseer dataset, the accuracy of of inference is around 93.39% while for Cora dataset the inference accuracy is 90.73% and 57.28% for Pubmed dataset. This indicates an adversary advantage of 86.78% (Citeseer), 81.06% (Cora) and 14.56% (Pubmed). The same train-test-validation distribution is used for the three datasets with 30% train records and 60% test records and remaining 10% as validation records.

5.5 Attribute Inference Attack

In case of attribute inference attacks, we evaluate two state of the art embedding models: Node2Vec and DeepWalk, using three attack models: Neural Networks (NN), Random Forest (RF) and Support Vector Machine (SVM). We generate embeddings using the two algorithms on Facebook and LastFM dataset which contain gender and location as sensitive attributes, respectively. That is, the adversary infers user gender as a target sensitive attribute in Facebook dataset classified into one of three classes: Male, Female and Others. The location target attribute for LastFM dataset is categorized in 18 locations for the users in the network.

The inference attack performance is given by the F1 score (Figure 10). For Facebook, the graph embedding using DeepWalk resulted in an F1 score of 0.57 for NN, 0.58 for RF and 0.59 for SVM classifier. On the other hand, Facebook's Node2Vec embedding showed an F1 score of 0.59, 0.57 and 0.61 respectively for NN, Rf and SVM attack classifier. In case of LastFM, we found the attack F1 scores for Node2Vec for higher than DeepWalk embeddings. The F1 score for DeepWalk 0.61, 0.62 and 0.65 corresponding to NN, RF and SVM attack classifier while the F1 score using Node2Vec embeddings are 0.80, 0.83 and 0.83 for NN, RF and SVM.

Impact of Adversary's Knowledge. The performance of the attack model for Facebook dataset did not increase by much. On increasing the knowledge of the adversary's auxiliary dataset from 30% to 50%, the confidence of attack on LastFM dataset increases. For DeepWalk algorithm, the attack F1 score increases to 0.69 from 0.61 for NN, 0.71 from 0.62 for RF and 0.69 from 0.65 for SVM attack classifier. On the other hand, for Node2Vec, the attack model F1 score increased to 0.83 from 0.80 for NN, 0.84 from 0.83 for RF and 0.86 from 0.83 for SVM.

6 RELATED WORK

Inference attacks that violate data privacy have been explored in the context of traditional machine learning models. Membership Inference attacks can be deployed in both white-box [27] and blackbox [31] setting in traditional machine learning algorithms. These attacks are further extended to collaborative learning [24, 27] and generative models [16]. On the other hand, reconstruction attacks infer private attributes of the inputs passed to the models [2, 10–12]. Other privacy attacks aim to extract hyperparameters [38], reverse engineer the model architecture and parameters using side channels [9] or the output predictions [8]. Memorization of data by Neural Networks has been attributed as a major cause for privacy leakage [5, 33, 34]. Further, recent works have indicated privacy risks in Graph NNs where an adversary can infer the presence of a link between two nodes using a manual threshold between the distance of two node features [17]. This attack however, is subsumed within our more generic attack methodology where we extract the entire adjacency matrix which can be used to infer the presence of links. Text models have been shown to leak user data through attribute inference and inversion attacks [28, 32]. However, a direct application of these attacks is not possible in case of high dimensional graphs and requires additional consideration to the network structure making our problem challenging. Other than privacy attacks, adversarial attacks against GNNs [3, 43] have been explored as well as training algorithms to enhance the robustness against such attacks [42, 44].

To mitigate Membership attacks, Memguard [19] and AttriGuard [18] add carefully crafted noise to the final output prediction to misclassify the shadow model attacks. Adversarial regularization using minimax optimization regularizes the model to mitigate inference attacks [26]. Regularization through ensemble training, dropout and L2-regularization have been studied [30]. Differential Privacy mitigates such privacy attacks with theoretical guarantees by adding noise to gradients but faces an unbalanced privacy accuracy tradeoff [1]. Such Differential Privacy frameworks have also been explored in the context of graph and text embeddings [39, 40] but their efficacy on lowering privacy risks from the proposed attacks is yet to be explored.

7 DISCUSSIONS AND CONCLUSIONS

This work provides the first comprehensive privacy risk analysis related to graph embedding algorithms trained on sensitive graph data. Specifically, this paper quantifies privacy leakage of three major classes of privacy attacks under practical adversary assumptions and threat models, namely membership inference, graph reconstruction and attribute inference. Firstly, an adversary conducting a membership inference attack aims to infer whether a given user's node was used in the training graph dataset or not. Secondly, publicly released embeddings can be inverted to obtain the input graph data enabling an adversary to perform graph reconstruction attack on the sensitive graph data. This further enables the adversary to perform link inference attack to infer whether a link exists between two nodes in the network. Finally, we show that an adversary can infer sensitive hidden attributes of users such as gender and location from the graph embeddings. Our results underlines many privacy risks in graph embeddings and calls for further research to mitigate these privacy threats.

Potential mitigation strategies to lower the privacy risks can be considered. For instance, lowering the precision of the embedding vector for each node by rounding can help to reduce the attack model from learning rich features about the inputs [28, 31]. In the proposed attacks, the attacker model is a machine learning algorithm vulnerable to adversarial examples, i.e, imperceptible noise added to the output prediction to force the target model to misclassify. The embeddings can be released with an additional adversarial noise to misclassify the target model while additionally ensuring utility [18, 19]. Further, the inference attacks can be modelled within the training process as a minimax adversarial training with joint optimization to minimize the model loss using the graph embeddings (e.g., GNNs) while maximising the adversary's loss on inferring the sensitive inputs [26, 32]. Finally, Differential Privacy can provide a theoretical bound on the total privacy leakage from the downstream processing from embeddings on an individual's data point [39, 40]. However, the efficacy of these potential mitigations are left for future work.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *CCS*. 308–318.

[2] Giuseppe Ateniese, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. 2015. Hacking Smart Machines with Smarter Ones: How to Extract Meaningful Data from Machine Learning Classifiers. *Int. J. Secur. Netw.* 10, 3 (2015), 137–150.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *ICML*.

[4] H. Cai, V. W. Zheng, and K. C. Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[5] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security*. 267–284.

[6] Haochen Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2018. A Tutorial on Network Embeddings. arXiv:1808.02590

[7] Jian Du, Shanghang Zhang, Guanhang Wu, Jos M. F. Moura, and Soummya Kar. 2018. Topology Adaptive Graph Convolutional Networks.

[8] Vasisht Duddu and D Vijay Rao. 2019. Quantifying (Hyper) Parameter Leakage in Machine Learning. *arXiv:1910.14409* (2019).

[9] Vasisht Duddu, Debasis Samanta, D Vijay Rao, and Valentina E. Balas. 2018. Stealing Neural Networks via Timing Side Channels.

[10] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *CCS*. 1322–1333.

[11] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. 2018. Property Inference Attacks on Fully Connected Neural Networks Using Permutation Invariant Representations. In *CCS*. 619–633.

[12] N.Z. Gong and B. Liu. 2016. You Are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. In *USENIX Security*. 979–995.

[13] Neil Zhenqiang Gong and Bin Liu. 2016. You Are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. In *USENIX Security*. 979–995.

[14] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). 1024–1034.

[16] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership Inference Attacks Against Generative Models. *PETS* 1 (2019), 133 – 152.

[17] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2020. Stealing Links from Graph Neural Networks. arXiv:arXiv 2005.02131

[18] Jinyuan Jia and Neil Zhenqiang Gong. 2018. Attriguard: A Practical Defense against Attribute Inference Attacks via Adversarial Machine Learning. In *USENIX Security*. 513–529.

[19] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. 2019. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *CCS*. 259–274.

[20] Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. 2017. AttriInfer: Inferring User Attributes in Online Social Networks Using Markov Random Fields. In *WWW*. 1561–1569.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Gnnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *ICLR*.

[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*.

[24] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting Unintended Feature Leakage in Collaborative Learning. In *SP*. 691–706.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*. 3111–3119.

[26] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2018. Machine Learning with Membership Privacy Using Adversarial Regularization. In *CCS*. 634–646.

[27] M. Nasr, R. Shokri, and A. Houmansadr. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *SP*. 739–753.
[28] X. Pan, M. Zhang, S. Ji, and M. Yang. 2020. Privacy Risks of General-Purpose Language Models. In *SP*.
[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *KDD*. 701–710.
[30] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *NDSS*.
[31] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *SP*. 3–18.
[32] Congzheng Song and Ananth Raghunathan. 2020. Information Leakage in Embedding Models. arXiv:2004.00053
[33] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine Learning Models That Remember Too Much. In *CCS*. 587–601.
[34] Congzheng Song and Vitaly Shmatikov. 2020. Overlearning Reveals Sensitive Attributes. In *ICLR*.
[35] Liwei Song and Prateek Mittal. 2020. Systematic Evaluation of Privacy Risks of Machine Learning Models. arXiv:2003.10595
[36] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* (2008), 2579–2605.
[37] Petar Velikovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[38] B. Wang and N. Z. Gong. 2018. Stealing Hyperparameters in Machine Learning. In *SP*. 36–52.
[39] Depeng Xu, Shuhan Yuan, Xintao Wu, and HaiNhat Phan. 2018. DPNE: Differentially Private Network Embedding. 235–246.
[40] Lili Jiang Xuan-Son Vu, Son N. Tran. 2019. dpUGC: Learn Differentially Private Representation for User Generated Contents. In *CICLing*.
[41] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).
[42] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *KDD*. 1399–1407.
[43] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *KDD*. 2847–2856.
[44] Daniel Zügner and Stephan Günnemann. 2019. Certifiable Robustness and Robust Training for Graph Convolutional Networks. In *KDD*. 246–256.