MEMBERSHIP INFERENCE ATTACK ON GRAPH NEURAL NETWORKS

Iyiola E. Olatunji, Wolfgang Nejdl & Megha Khosla *
L3S Research Center,
Hannover, Germany.
{iyiola,nejdl,khosla}@l3s.de

ABSTRACT

We focus on how trained Graph Neural Network (GNN) models could leak information about the *member* nodes that they were trained on. We introduce two realistic inductive settings for carrying out a membership inference (MI) attack on GNNs. While choosing the simplest possible attack model that utilizes the posteriors of the trained model, we thoroughly analyze the properties of GNNs which dictate the differences in their robustness towards MI attack. The surprising and worrying fact is that the attack is successful even if the target model generalizes well. While in traditional machine learning models, overfitting is considered the main cause of such leakage, we show that in GNNs the additional structural information is the major contributing factor. We support our findings by extensive experiments on four representative GNN models. On a positive note, we identify properties of certain models which make them less vulnerable to MI attacks than others.

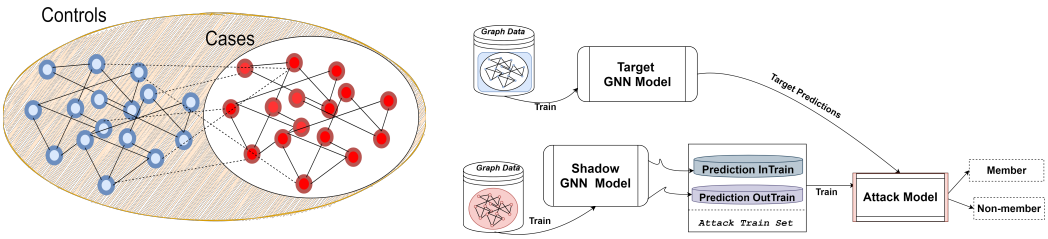
1 INTRODUCTION

Graph neural networks (GNNs) have gained substantial attention from academia and industry in the past few years. These models differ from the traditional machine learning models, in that they use additional relational information among the node instances to make predictions. In fact, the graph convolution-based model (Kipf & Welling, 2017) which is the most popular class of GNNs embeds graph structure into the model itself by computing representation of a node via recursive aggregation and transformation of feature representations of its neighbors. We focus on exposing the vulnerability of such models to *membership inference* (MI) attacks. In particular, we ask *whether the trained GNN model can be used to identify the input instances (nodes) that it was trained on.*

To motivate the importance of the problem for graphs, consider the use case shown in Figure 1a. Suppose a researcher has a list of patients infected with COVID19. The researcher is interested in understanding the various factor contributing to the infection. To account for the factors such as their social activity, she might want to utilize knowledge of friendship/social connection known among the patients. She then trains a GNN model on the graph induced on the nodes of interest and uses the trained node representations as additional input for her disease analysis models. A successful MI attack on the trained model would reveal the list of infected persons even though the model might have not used any disease-related sensitive information.

MI attack (Shokri et al., 2017; Salem et al., 2020) is usually modeled as a binary classification task, where the goal is to distinguish between the *target* model's behavior for the inputs it encountered during training from the ones which it did not. The inputs to the attack model are the class probabilities (posteriors) or the confidence values output by the target model for the corresponding data point. Thus, the adversary only requires a *black-box* access to the model where she can query the model on her desired data record and obtain the model predictions (output class probabilities). This is made possible due to low-cost APIs offering machine learning as a service (MLaaS). The attacker with access to such a service and her own data can then exploit the patterns in the posterior class probabilities to identify the training set. Intuitively, the model should be more confident of its predictions on training data as compared to other inputs.

*Full paper is available on arXiv https://arxiv.org/abs/2101.06570



(a) An example graph where the induced subgraph (red-colored nodes) was used for training the network. This induced graph corresponds to the social connections among the patients infected with COVID19. The graph structure among these nodes might be used to learn useful features of their interactions. The adversary then having access to the social network would be interested in discovering the infected patients (red nodes).

(b) Attack methodology for membership inference on GNN models. The training nodes and neighbor information (InTrain) used for training the shadow GNN model are labeled as *Member*. We also query the shadow model with nodes from a test graph (OutTrain) and labeled the predictions as *non-Member*. These predictions are used to train the attack model. The attacker then queries his trained attack model with posteriors obtained from the target model (target predictions) to infer membership.

Figure 1: (a) Motivation of MI attack for graphs and (b) Attack methodology for MI on GNNs

Consequently, much of the success of MI attacks in traditional ML models has been attributed to the tendency of the model to overfit or memorize the dataset (Zhang et al., 2016). We ask *if overfitting in GNNs is also the main contributing factor for successful membership inference.*

To summarize, our key contributions are as follows. **(1)** We introduce two realistic inductive settings for carrying out MI attack on GNNs. **(2)** We perform an extensive experimental study to expose the risks of data leakage in GNN models. We further attribute the differences between the model's robustness towards MI attack to the model architecture, in particular the employed aggregation mechanisms. **(3)** Contrary to the popular belief, we show that for GNNs, lack of overfitting does not guarantee robustness towards privacy attacks. Instead, the connectivity among the instances (unlike in tabular data) increases the vulnerability of GNN models towards privacy attacks.

2 Our Approach

2.1 Problem Description

Problem Statement. *Let a GNN model $\mathcal{M}$ be trained using the graph $G_t = (V_t, E_t)$. Given a node v and its L-hop neighborhood, determine if $v \in V_t$. Note that even if v was in the training set, the L-hop neighborhood known to the adversary might not be the same as the one used to train the model $\mathcal{M}$.*

Transductive vs Inductive Training. Graph models can either be trained transductively or inductively. In the transductive setting, the complete graph G is used during training where only a subset of nodes are labeled. In the inductive setting, only the edge information of the training nodes is present during training. Therefore, in this work we focus on the inductive setting. We present our arguments against using transductive setting in AppendixA.7.

We propose two realistic inductive settings: (i) in the first setting which we call the *TSTF (train on subgraph, test on full) setting*, in which the whole graph G is available to the adversary but she is not aware of the subgraph $\mathcal{G}_t$ used for training the target model, (ii) in our second setting *TSTS (train on subgraph test on subgraph) setting*, the target graph G_t is an isolated component of G. The adversary also does not have full access to G but only to the induced edges among the train/test nodes. From the perspective of model access, we only assume that the adversary has only black box access to the target model and access to the dataset drawn from a *similar* distribution as the target data. We show that the attacks can be successful even without the knowledge of the target model architecture and hyperparameter settings.

2.2 ATTACK METHODOLOGY

We model the problem of MI as a binary classification task where the goal is to determine if a given node $v \in V_t$. We denote our attack model which is a binary classifier by $\mathcal{A}$. To train $\mathcal{A}$, we first derive a labeled training dataset with ground truth information. In particular, we build a *shadow model* which simulates the behavior of the target model. Illustration of the attack methodology is shown in Figure 1b. For results shown in the main paper, the shadow model uses the same ML algorithm and has the same hyperparameters as the target model. We later relax these assumptions in Appendix A.5.

We organize the adversary's methodology into three phases, shadow model training, attack model training, and membership inference.

Shadow model training. To train the shadow model, we assume that the adversary has access to the graph with vertex set V_{shadow} that comes from the same underlying distribution as V_t. Then the adversary trains the shadow model using the shadow model's training split, $V_{shadow}^{Train} \subset V_{shadow}$. To replicate the behavior of the target model, we use the output class probabilities by the target model (when V_{shadow}^{Train} is used as input) as the ground truth for training the shadow model. This would result in querying the target model for each vertex in V_{shadow}^{Train}. We later relax the number of queries required to 0 by directly training the shadow model on ground truth labels. We observe there is no significant change in attack success rate (c.f. Section A.4). We also find that we do not need to know the exact target model. In fact we show that using GCN as the shadow model irrespective of the actual target model already results in good attack performance (c.f. Section A.6).

Attack model training. To construct the attack model we use the trained shadow model to perform prediction over all nodes in V_{shadow}^{Train} and $V_{shadow}^{Out} = V_{shadow} \setminus V_{shadow}^{Train}$ and obtain the corresponding output class posterior probabilities. For each node, we take the posteriors as input feature vector for the attack model and assigns a label 1 if the node is in V_{shadow}^{Train} and 0 if the node is from V_{shadow}^{Out}. These assigned labels serves as ground truth data for the attack model. All the generated feature vectors and labels are used in training the attack model.

Membership inference. To perform the inference attack on whether a given node $v \in V_t$, the adversary queries the target model with v and its known neighborhood to obtain the posteriors. Note that even if v was part of training data, the adversary would not always have access to the exact neighborhood structure that was used for training. Then she inputs the posteriors into the attack model $\mathcal{A}$ to obtain the membership prediction.

Our code is available on Github [1].

3 RESULTS AND DISCUSSION

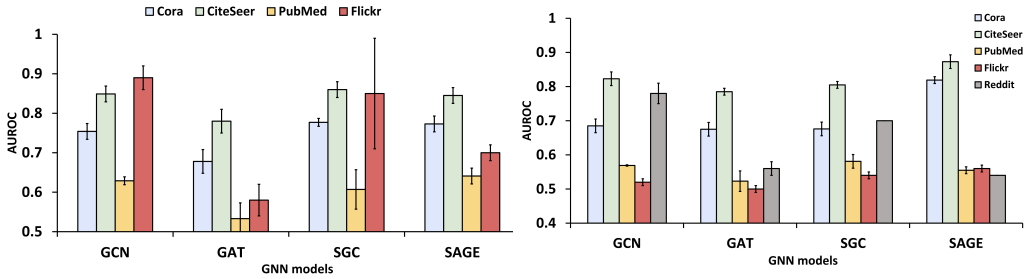
Evaluation and Metrics. We compare four popular GNN models: (i) graph convolution network (GCN)(Kipf & Welling, 2017), (ii) graph attention network (GAT) (Veličković et al., 2018) (iii) simplified graph convolution (SGC) (Wu et al., 2019) and (iv) GraphSage (SAGE) (Hamilton et al., 2017). We ran all experiments for 10 random data splits and report the average performance along with the standard deviation on five different dataset (CORA, CITESEER, FLICKR, PUBMED and REDDIT) commonly used as benchmark dataset for evaluating GNN performance. We report AUROC scores, Precision and Recall for the attack model. *Precision* measures the fraction of predicted member nodes that are indeed members of the training dataset while *recall* measures the fraction of the training dataset which are correctly inferred as members by the attacker. For the target GNN models, we report train and test accuracy.

Due to space constraints, we present summarized results in the main paper while moving the detailed tables to the Appendix.

3.1 COMPARISON AMONG GNN MODELS

In this section, we show that all the four studied GNN models are prone to MI leakage and exhibit different levels of risks with respect to the attack.

[1] https://github.com/iyempissy/rebMIGraph



(a) TSTF Setting

(b) TSTS Setting

Figure 2: Mean AUROC scores of attack model against different GNN models.

Setting TSTF. The AUROC scores for the attack model on all datasets except Reddit are plotted in Figure 2a. For the models GCN and SGC, the attack model obtains similar scores. Note that the difference between SGC and GCN is that SGC does not use a non-linear transformation after the feature aggregation step. The feature aggregation scheme employed in both the models is exactly the same.

GAT is the most robust towards the attack. GAT also differs from the models in that it uses a weighted aggregation mechanism. SAGE employs a mean aggregation over the neighborhood's features. In addition, unlike the other models for the aggregation step it samples a fixed number of neighbors rather than using the complete neighborhood. Though it shows similar results for 3 citation networks, the attack is less successful for the larger graph FLICKR (when compared to GCN and SGC). We attribute the reason for such an observation to the induced noise in the neighborhood structures because of random sampling of neighbors (more details in Appendix A.3.1) . Obviously, the effect is more prominent in denser graphs like FLICKR as compared to CORA where the average degree is less than 2 (see Tables 1 and 2 in Appendix).

Setting TSTS. Unlike in the TSTF setting, the train and test sets in this setting are disconnected. This implies that any node $v \in V_t$ and its exact neighborhood used during training is available to the adversary. We also see a huge reduction in test set performance implying that the model is not generalising well to the test set. Intuitively it would be much easier to attack in this setting. The AUROC scores for the corresponding attack are shown in Figure 2b. Precision and recall of the attack model along with train-test set performance of the target model are shown in Tables 3,4 and 5. We observe that for CORA and CITESEER the attack has similar success rate as in TSTF setting, for FLICKR on the other hand, the attack performance degrades. For the larger dataset REDDIT, the attack is successful for GCN and SGC models with a mean precision of 0.81 and 0.74 (see Table 5) respectively.

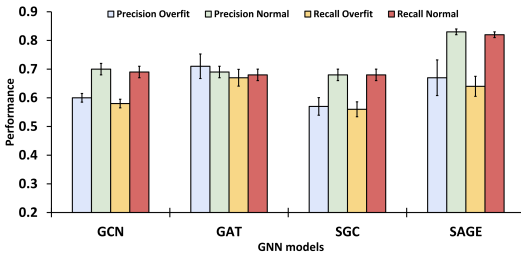
GAT and SAGE shows more robustness with AUROC scores close to 0.5 (implying that the attack model cannot distinguish between member and non-member nodes better than a random guess) for datasets: PUBMED, FLICKR, REDDIT even if the generalization error of the target models is high. In Section 3.2, we further investigate the specific case of overfitting, where we also find that due to overconfident posteriors for the test set (though for incorrect predictions), the attack model is unable to distinguish between the member and non-member nodes.

3.1.1 WHICH MODEL IS MORE ROBUST TOWARDS MI ATTACK AND WHY?

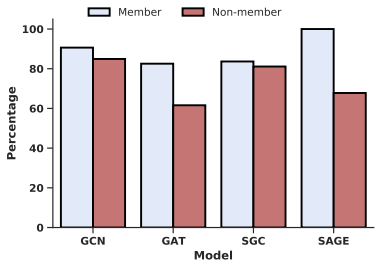
To summarize, we found GAT to be most robust towards MI attacks. The reason can be attributed to the learnable attention weights for different edges. The above fact implies that instead of the original graph model, a distorted one dictated by supervised signals of class labels is embedded in the model. This is in contrast with SGC and GCN where the actual graph is embedded with equal edge weights. Also in SAGE which uses neighborhood sampling before the aggregation operation does not use the complete information of the graph during training. The effect is more prominent in denser graphs in which only a small fraction of neighborhood is used during a training epoch.

3.2 EFFECT OF MODEL OVERFITTING

To investigate the effect of overfitting, we train the models such that they achieve zero training loss or high generalization error.



(a) Influence of overfitting on attack on Cora. "**Normal**" refers to the case when all models were trained for a fixed number of epochs.



(b) Percentage of nodes (members and non-members) for which the maximum posterior is greater than 0.8 when GNN models overfit on Cora. The statistics are shown for one random data split.

Figure 3: Effect of overfitting. In figure (a) we observe a surprising effect that attack is less successful for overfitted models. The reason behind such an effect is explained in (b) which illustrates that the model overfitted model also make extremely confident albeit incorrect predictions on the test (unseen) set.

Figure 3a shows the comparison between a "normal" model and overfitted model. The attack precision and recall of the overfitted model consistently decreases across all models except for GAT. To understand the reasons behind the above observations, we investigate the posterior distribution of member and non-member nodes. In Figure 3b, we show the distribution of *maximum* posterior (i.e., the posterior probability corresponding to the predicted class) of overfitted models on train (member) and test (non-member) set. We observe that in case of overfitting, the GNN model not only make highly confident predictions for the train set but also for the test set. Most of the nodes whether train or test set obtain the maximum class probability (or posterior) greater than 0.8 for models GCN and SGC. For GAT and SAGE, the attack models obtains higher precision given that relatively less number of non-member nodes obtain high maximum posterior.

4 CONCLUSION

We empirically compare the vulnerability of four GNN models to MI attacks. We show that the simplest binary classifier based attack model already suffices to launch an attack on GNN models even if they generalize well. We find that overfitting does not always lead to improvement in the attack success rate. We find that overfitted GNN models not only make very confident predictions for the member nodes but also for the non-member nodes (though such predictions might be incorrect). In such cases the attack model has less distinguishing power given only the black box access to posteriors.

On a positive note, among the studied models, GAT, is most robust towards MI attack. Also, SAGE shows robustness for larger datasets. The reasons can be attributed to the fact that they do not embed the actual training graph in the model. For example in GAT, a weighted aggregation mechanism is used where the weights are dictated by the supervised task signal. SAGE, instead of using all neighbors during aggregation, uses only a sample of the neighborhood. We could therefore observe its robustness towards the attack in larger denser graphs.

ACKNOWLEDGEMENT

This work is in part funded by the Lower Saxony Ministry of Science and Culture under grant number ZN3491 within the Lower Saxony "Vorab" of the Volkswagen Foundation and supported by the Center for Digital Innovations (ZDIN), and the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor (grant number 01DD20003).

REFERENCES

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pp. 1291–1308, 2020.

Reza Shokri, Stronati Marco, Song Congzheng, and Shmatikov Vitaly. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, 2017.

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

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A APPENDIX

A.1 EFFECT OF NUMBER OF CLASSES

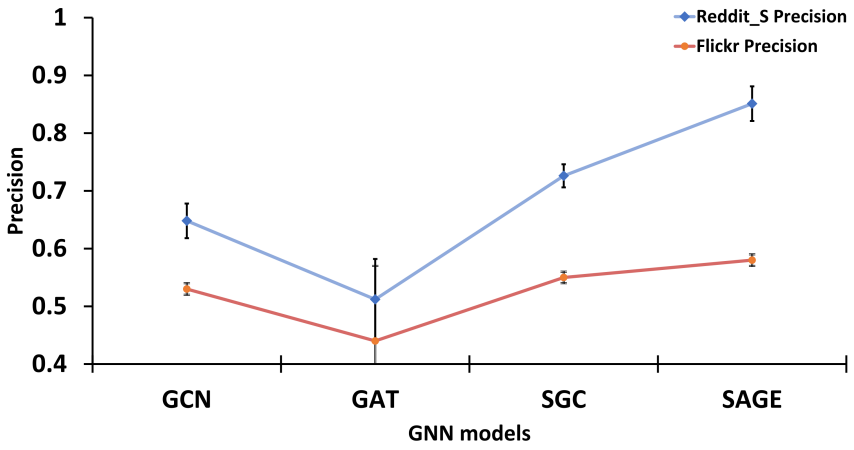


Figure 4: The effect of number of classes on attack performance. Reddit_S is subset of REDDIT dataset having 50 nodes per class. REDDIT has 41 classes while FLICKR has 7 classes.

We investigated the influence of the number of available node classes on attack prediction. For all dataset with 6 or more classes, the attack performs well. However, for PUBMED with 3 classes, the attack precision and recall is relatively low in both TSTF and TSTS setting across all models. Figure 4 shows the effect of number of classes on attack precision using 50 nodes in each class of the REDDIT dataset consisting of 41 classes and FLICKR dataset consisting of 7 classes. The result points out that GNN models are more vulnerable with increase in number of classes.

A.2 PERFORMANCE IN TSTF SETTING

Tables 1 and 2 report performance of the target and the attack models on CORA, CITESEER, PUBMED and FLICKR. All target models except SAGE encountered memory issues for the REDDIT in TSTF setting. Therefore, we only provide results for REDDIT in the TSTS setting.

Table 1: Performance of different GNN models on CORA and CITESEER dataset in the TSTF setting.

	Cora				CiteSeer			
	Target Model		Attack Model		Target Model		Attack Model	
	TRAIN	TEST	PRECISION	RECALL	TRAIN	TEST	PRECISION	RECALL
GCN	0.81 ± 0.01	0.84 ± 0.02	0.76 ± 0.02	0.75 ± 0.02	0.90 ± 0.01	0.75 ± 0.01	0.86 ± 0.02	0.85 ± 0.02
GAT	0.76 ± 0.01	0.83 ± 0.02	0.69 ± 0.03	0.68 ± 0.03	0.87 ± 0.01	0.73 ± 0.02	0.79 ± 0.02	0.78 ± 0.03
SGC	0.73 ± 0.02	0.84 ± 0.01	0.77 ± 0.02	0.78 ± 0.01	0.85 ± 0.01	0.74 ± 0.02	$\mathbf{0.87 \pm 0.02}$	0.86 ± 0.02
SAGE	0.99 ± 0.002	0.76 ± 0.02	$\mathbf{0.78 \pm 0.02}$	0.77 ± 0.02	0.99 ± 0.001	0.67 ± 0.01	0.86 ± 0.03	0.85 ± 0.02

Table 2: Performance of different GNN models on PUBMED and FLICKR dataset in the TSTF setting.

	PubMed				Flickr			
	Target Model		Attack Model		Target Model		Attack Model	
	TRAIN	TEST	PRECISION	RECALL	TRAIN	TEST	PRECISION	RECALL
GCN	0.74 ± 0.01	0.80 ± 0.01	0.64 ± 0.01	0.63 ± 0.01	0.37 ± 0.03	0.18 ± 0.02	$\mathbf{0.90 \pm 0.02}$	0.89 ± 0.03
GAT	0.70 ± 0.01	0.76 ± 0.02	0.40 ± 0.17	0.53 ± 0.04	0.21 ± 0.02	0.14 ± 0.03	0.61 ± 0.03	0.58 ± 0.03
SGC	0.68 ± 0.01	0.78 ± 0.01	0.64 ± 0.01	0.60 ± 0.05	0.45 ± 0.01	0.10 ± 0.02	0.85 ± 0.02	0.85 ± 0.20
SAGE	0.82 ± 0.003	0.78 ± 0.02	$\mathbf{0.66 \pm 0.02}$	0.64 ± 0.02	0.70 ± 0.04	0.20 ± 0.01	0.71 ± 0.01	0.70 ± 0.01

A.3 PERFORMANCE IN TSTS SETTING

Tables 3, 4 and 5 provide performance scores of all target and attack models in TSTS setting.

Table 3: Performance of different GNN models on CORA and CITESEER dataset in the TSTS setting.

	Cora				CiteSeer			
	Target Model		Attack Model		Target Model		Attack Model	
	TRAIN	TEST	PRECISION	RECALL	TRAIN	TEST	PRECISION	RECALL
GCN	0.81 ± 0.01	0.60 ± 0.02	0.70 ± 0.02	0.69 ± 0.02	0.90 ± 0.01	0.61 ± 0.01	0.83 ± 0.01	0.82 ± 0.02
GAT	0.76 ± 0.01	0.56 ± 0.02	0.69 ± 0.02	0.68 ± 0.02	0.87 ± 0.01	0.62 ± 0.02	0.80 ± 0.01	0.79 ± 0.01
SGC	0.72 ± 0.01	0.57 ± 0.02	0.68 ± 0.02	0.68 ± 0.02	0.85 ± 0.01	0.61 ± 0.02	0.82 ± 0.02	0.81 ± 0.02
SAGE	0.99 ± 0.002	0.70 ± 0.02	$\mathbf{0.83 \pm 0.01}$	0.82 ± 0.01	0.99 ± 0.003	0.70 ± 0.02	$\mathbf{0.88 \pm 0.01}$	0.87 ± 0.02

Table 4: Performance of different GNN models on PUBMED dataset in the TSTS setting.

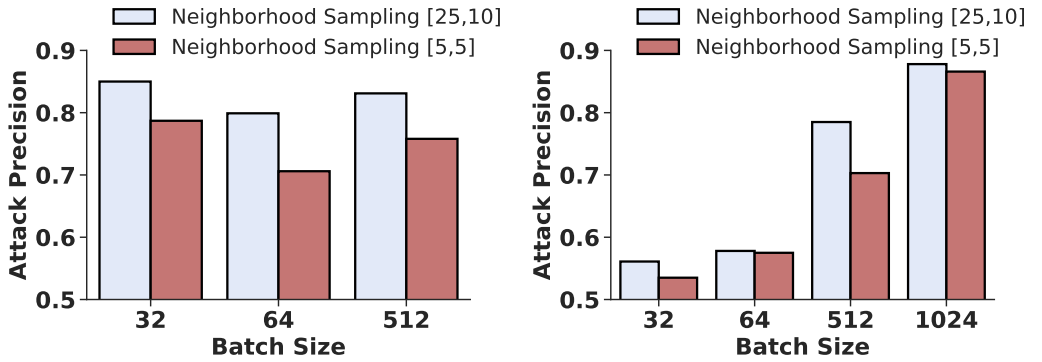
	PubMed			
	Target Model		Attack Model	
	TRAIN	TEST	PRECISION	RECALL
GCN	0.73 ± 0.01	0.71 ± 0.01	0.58 ± 0.004	0.57 ± 0.002
GAT	0.70 ± 0.01	0.67 ± 0.01	0.35 ± 0.18	0.52 ± 0.03
SGC	0.68 ± 0.01	0.66 ± 0.01	$\mathbf{0.59 \pm 0.02}$	0.58 ± 0.01
SAGE	0.82 ± 0.003	0.77 ± 0.01	0.57 ± 0.01	0.56 ± 0.01

Table 5: Performance of different GNN models on FLICKR and REDDIT dataset in the TSTS setting.

	Flickr				Reddit			
	Target Model		Attack Model		Target Model		Attack Model	
	TRAIN	TEST	PRECISION	RECALL	TRAIN	TEST	PRECISION	RECALL
GCN	0.39 ± 0.03	0.20 ± 0.01	0.53 ± 0.01	0.52 ± 0.01	0.42 ± 0.20	0.29 ± 0.08	$\mathbf{0.81 \pm 0.13}$	0.76 ± 0.06
GAT	0.21 ± 0.02	0.17 ± 0.01	0.44 ± 0.13	0.51 ± 0.01	0.50 ± 0.15	0.36 ± 0.10	0.60 ± 0.05	0.54 ± 0.05
SGC	0.44 ± 0.02	0.20 ± 0.01	0.55 ± 0.01	0.54 ± 0.01	0.40 ± 0.16	0.29 ± 0.05	0.74 ± 0.06	0.68 ± 0.11
SAGE	0.69 ± 0.05	0.23 ± 0.01	$\mathbf{0.58 \pm 0.01}$	0.56 ± 0.01	0.70 ± 0.04	0.52 ± 0.03	0.51 ± 0.13	0.52 ± 0.03

A.3.1 EFFECT OF NEIGHBORHOOD SAMPLING IN SAGE

To further understand the reason for SAGE model's behavior on smaller dataset, we varied the number of neighbors sampled at different layers of the network and the batch size. SAGE utilizes mini-batching technique that contains node on which representation is generated. We used [25,10] and [5,5] as sampled neighborhood size in layers 1 and 2. For instance, [5,5] implies that on the first and second layer of the SAGE model, only 5 neighbors of a node are sampled. As shown in Figure 5a, the attack precision decreases as the number of sampled nodes decreases. This is due to the fact that the model uses the noisy neighborhood information and is not able to fully encode the graph structure in the model, this in turn makes the posteriors of neighboring nodes less correlated. Similar results are obtained for a larger dataset, FLICKR (shown in Figure 5b).



(a) CORA (b) FLICKR

Figure 5: Effect of training batch size and sampled neighbors on attack precision for SAGE model on (a) CORA and (b) FLICKR dataset.

A.3.2 EFFECT OF INSTANCE CONNECTIVITY

Unlike tabular data, in graphs, the data points (which are the nodes) are linked or connected. Though our attack model only uses the prediction probabilities and has no knowledge of the graph structure, the attack pattern reveals the effect of correlated posteriors of neighboring nodes. We will illustrate the attack pattern with respect to node or instance connectivity using Cora dataset as an example.

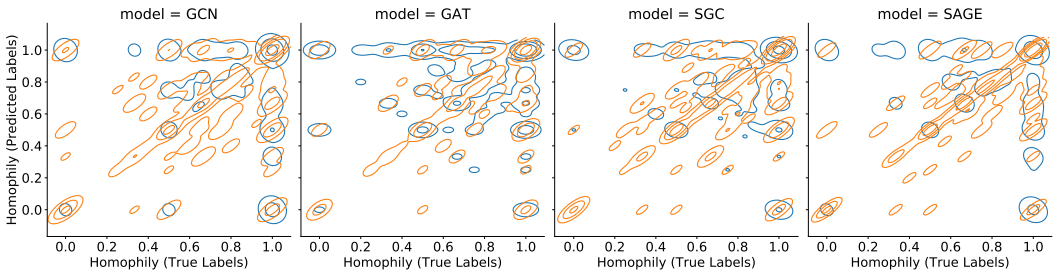


Figure 6: Joint density plots of true and predicted homophily. The orange contour lines correspond to correctly predicted nodes (by the attack model) and the blue lines correspond to incorrectly predicted nodes on CORA dataset in the TSTF setting.

Given the predicted posteriors as input, the attack model labels the node instance as member (label 1) or non-member (label 0) node. To understand the pattern of label assignments by the attack model we need the following definition.

Definition (Homophily)**.** *For any node u which is either a member or non-member, we define its homophily as the fraction of its one-hop neighbors which has the same label as u. The neighborhood of any node is computed using the graph available to the adversary. We call homophily with respect to*

*ground truth as the **true homophily** and with respect to the attack model predictions as the **predicted homophily**.*

Therefore, a true homophily of 1 means u and all its neighbors in the graph used by the adversary are members or non-members. Similarly, a predicted homophily of 1 implies that u and its neighbors were assigned the same label by the attack model. In Figure 6, we plot the joint distribution of true and predicted homophily of the correctly (orange contour lines) and incorrectly (blue contour lines) predicted nodes. Our following observations imply the attack model even without using explicitly the graph structure is greatly influenced by the correlation of prediction probabilities (posteriors) of the connected nodes and assigns connected nodes the same label.

1. Blue regions on the upper left corner correspond to nodes with 0 true homophily and high predicted homophily. Such a node for example if is a member node, then all of its neighbors are non-members. The attack model makes wrong predictions while matching their predicted labels to their neighbors. On the other hand, the attack model made the right predictions for its neighbors.
2. Blue regions on the upper right corner of the plot corresponds to nodes with high true and predicted homophily. This implies that the attack model assigned wrong label to the node and its neighbors.
3. Regions along the diagonal as well as above the diagonal correspond to nodes whose predicted homophily increased.
4. The lower precision of the attack model for GAT (as compared to other models) is evident in Figure 6 where a larger number of nodes (corresponding concentrated blue regions with high predicted homophily) and its neighbors are assigned the wrong label. Though the attack model is not very successful in this case, we still observe that the decisions of the attack model are correlated for connected nodes.
5. For GCN, SGC and SAGE the attack model obtains similar Precision and Recall. The corresponding plots in Figure 6 show similar patterns where for most of the nodes the predicted homophily either increases or stays the same. Note that high density of orange contour lines over the diagonal implies that the attack model has predicted the node as well its same labelled neighbors correctly.

To summarize, we observe that the attack model's predictions on a node are highly correlated with its predictions on its neighbors. As the attack model is agnostic to the graph structure, we conclude that posterior of neighboring nodes are also correlated which the attack model is able to exploit.

To further support our conclusions from CORA, we additionally investigated a larger dataset, FLICKR. Figure 7 shows the corresponding homophily plot for FLICKR dataset (for TSTF Setting) in where we observe clear distinctions among the attack model's behavior in different models. The attack is most successful in GCN and least in GAT. We observe that for both the models the predicted homophily increases for most of the nodes (with dense regions on or above the diagonal) with the difference that for GAT, the attack is more confused and assigns the wrong label not only to the node but also to most of its connected neighbors.

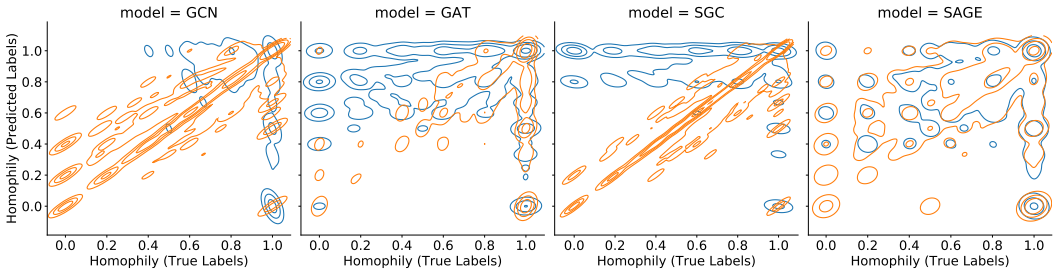


Figure 7: Joint density plot of true and predicted homophily on FLICKR dataset.

Besides, we investigate the sensitivity of attacks to *data used to train the shadow model* and knowledge of *target model architecture* and *hyperparameter settings*. **First**, in Section A.4, we relax the number

of queries required for imitating the behavior of target model to 0 with the assumption that the training data for shadow model is drawn from a similar distribution as that of the target model. **Second**, we find that as long as we use a graph convolution (message passing) based model architecture for constructing the shadow model, the assumption about knowledge of exact target model architecture can be relaxed (c.f. Section A.6). **Finally**, we experimented using different sizes of hidden layer and conclude that knowledge of exactly same hyperparameter settings is not necessary (c.f. Section A.5). We observe that the better the performance of shadow model on the shadow dataset, the better is the attack success rate.

A.4 RELAXING NUMBER OF QUERIES

We relax the number of queries required to imitate a target model to 0 by assuming that the adversary has access to dataset from a similar distribution as the dataset used for the target model. To construct such datasets we randomly sampled disjoint sets of nodes from the full graph for the target as well as the shadow model. We then construct the corresponding induced graphs on the node sets to train the shadow and target models. Note that the shadow model data in this case will not be exactly from the same distribution as the target graph since our construction would not exactly preserve the structural characteristics of these graphs e.g. degree distribution. The data used in training the shadow model is in fact similar but not from the exact same distribution as target model. We found that training the shadow model using ground truth labels perform similarly to querying the target model in the order of $\pm$ 0.02 standard deviation.

A.5 RELAXING HYPERPARAMETERS ASSUMPTION

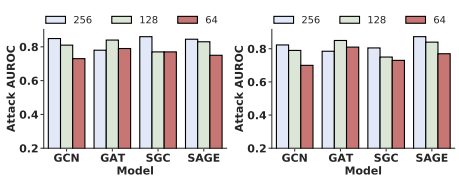
In this section, we relax the assumption that the hyperparameters of the target model are known by varying the number of hidden neurons of the shadow model. The only hyperparameter which can be varied is the number of neurons in hidden layer. We experiment with three values $\{256, 128, 64\}$. With respect to number of layers, it is widely accepted fact that increasing the number of layers in GNNs increases oversmoothing, therefore we fix that to the most common setting, i.e., 2.

The corresponding AUROC scores are provided in Figure 8. A general trend is that the larger the hidden layer size, the better the attack performance. This is expected as increase in size of the hidden layer increases the model parameters/capacity to store more specific details about the training set. We also observe that the better the performance of the shadow model on the train set, the better the attack success rate. Therefore, though we observe some reduction in attack performance for PUBMED when using 128 or 64 as hidden layer size, an attacker can just choose the hyperparameter which gives the best train set performance on its shadow dataset.

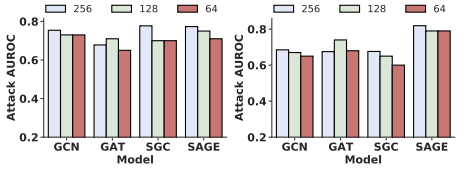
A.6 RELAXING THE KNOWLEDGE OF TARGET MODEL ASSUMPTION

We further relax the assumption that the attacker knows the architecture of the target model. Specifically, we used SGC as the shadow model and other GNN models as the target model. Motivated by the difference between SGC model and GCN model by removing the non-linear activation function from GCN, we aim to quantify how this difference affects the attack performance. Therefore, we also used GCN as a shadow model. The mean precision scores corresponding to attacks for different datasets are presented in Figure 9.

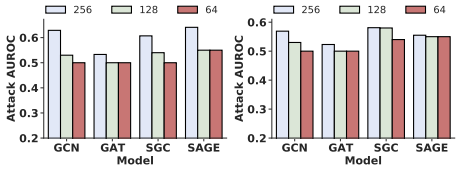
In both TSTF and TSTS, on the CITESEER and CORA dataset, the performance of using different shadow model is equivalent to using the same model as the target model except for SAGE where significant drop in performance is observed. However, GCN performs significantly better than SGC when used as the shadow model by the attacker. On the PUBMED dataset, an interesting observation, particularly for GAT is that when SGC is used for shadow model, the attack precision and recall increases more than when GAT (target model) is used as the shadow model. On the FLICKR and REDDIT dataset, using GCN as the shadow model performs comparable to an adversary knowing the architecture of the target model in both TSTS and TSTF setting. However, using SGC as shadow model significantly led to reduced attack precision in the TSTF setting. In fact, better attack precision is achieved when GCN is used as the shadow model and SAGE is used as the target model on large networks like REDDIT. Therefore, we conclude that using GCN as the shadow models is sufficient to



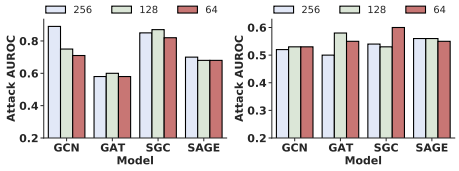
(a) CiteSeer TSTF (b) CiteSeer TSTS



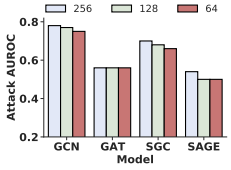
(c) Cora TSTF (d) Cora TSTS



(e) PubMed TSTF (f) PubMed TSTS

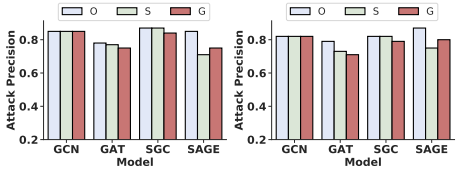


(g) Flickr TSTF (h) Flickr TSTS

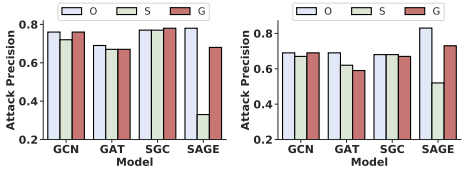


(i) Reddit TSTS

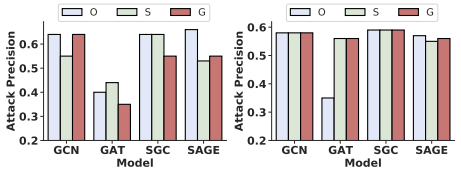
Figure 8: Relaxing hyperparameter assumption. We varied the number of hidden neurons. The original shadow model was trained with 256 hidden neurons.



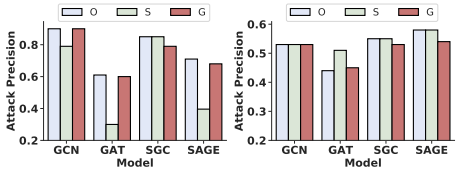
(a) CiteSeer TSTF (b) CiteSeer TSTS



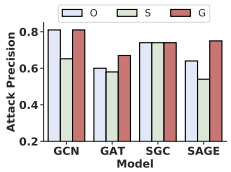
(c) Cora TSTF (d) Cora TSTS



(e) PubMed TSTF (f) PubMed TSTS



(g) Flickr TSTF (h) Flickr TSTS



(i) Reddit TSTS

Figure 9: Relaxing the knowledge of target model. O = Original performance when target and shadow model have the same architecture. S = using SGC as shadow model, G = using GCN as shadow model.

launch a successful attack and that the removed non-linear activation function of SGC makes it less attractive option to use an a "universal" shadow model.

A.7 TRANSDUCTIVE VS INDUCTIVE TRAINING

There are two popular train-test settings for graph based models: *transductive* and *inductive*. In the transductive setting, the complete graph G is used for training where only a subset of nodes are labelled. Though the GNN model is trained using the supervised loss on labelled nodes, the unlabelled nodes are also part of the training process. For a given input node instance, say u, a k-layer GNN model uses as input the k-hop neighborhood of u. This neighborhood might also include a number of unlabelled nodes whose feature representations were used to learn the representation of node u. This implies that each node whether labelled or unlabelled contributes to training of the model. Hence, from privacy perspective, these nodes are also sensitive as they influence the behavior of the model. Or if one assumes that labels are the sensitive information, in that case a successful

attack should reveal the labels of unlabelled nodes. That can be easily done just by querying the target model and does not constitute a MI attack. Hence, we focus on the inductive setting where only the edge information of the training nodes is present during training. Moreover, models trained in the inductive setting can be used to classify nodes that were not seen during training.