Privacy Protection from GCN on a Per-Node Basis: Selective Learnability Lock (SLL)

Edward Wei, Lu Lin, Hongning Wang
University of Virginia
Charlottesville, Virginia, USA
nyw6dh@virginia.edu,ll5fy@virginia.edu,hw5x@virginia.edu

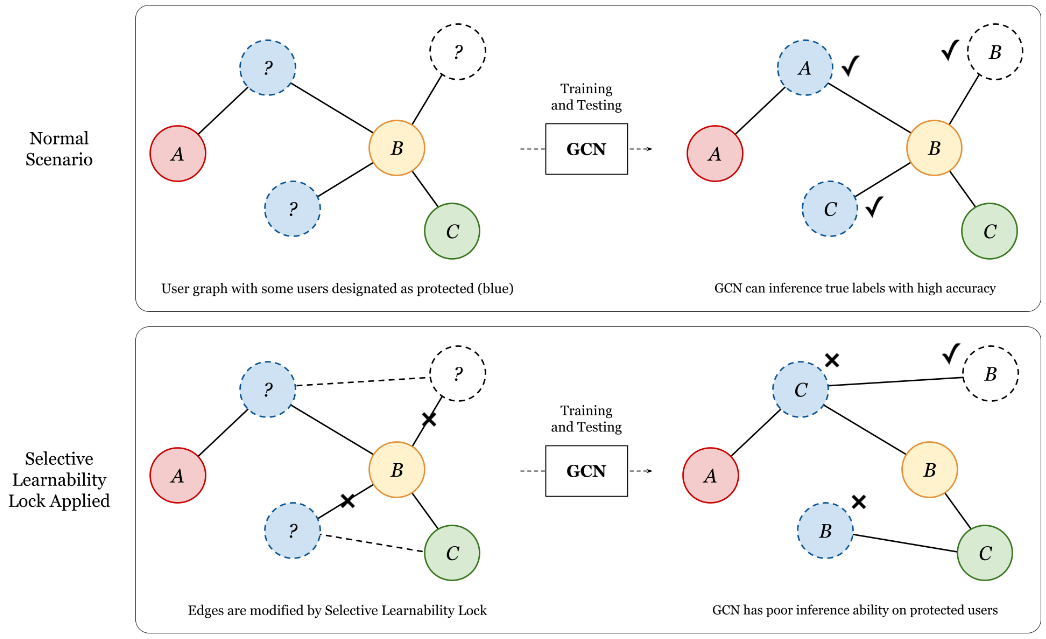
Normal Scenario
Training and Testing
GCN
A
B
C
?
User graph with some users designated as protected (blue)
GCN can inference true labels with high accuracy
Selective Learnability Lock Applied
Edges are modified by Selective Learnability Lock
GCN has poor inference ability on protected users

Figure 1: Strategic modifications of a graph selectively harm downstream classifier's ability to learn patterns in arbitrarily chosen set of "protected" users.

ABSTRACT

A novel problem setting where a defender seeks to stop a downstream user for training an effective GCN over an arbitrarily selected set of protected nodes of a graph is proposed and explored. A robust graph data protection method, Selective Learnability Lock (SLL), is developed to address the novel problem setting. SLL uses a surrogate GCN to approximate the downstream user and optimizes the topology of the graph to maximize loss on the protected nodes while minimizing loss on the other nodes. A gradient guidance optimization is implemented in SLL to allow it to be more robust and efficienct with regard to dataset size. SLL is experimentally effective, providing a better solution than existing possibilities such as simplistic solutions or random noise, and can be extended to apply to multi-task scenarios.

CCS CONCEPTS

• **Computing methodologies → Neural networks**; • **Security and privacy → Privacy protections**; • **Mathematics of computing** → *Graphs and surfaces.*

KEYWORDS

graph mining, privacy protection, graph convolutional network, topology attack

ACM Reference Format:
Edward Wei, Lu Lin, Hongning Wang. 2022. Privacy Protection from GCN on a Per-Node Basis: Selective Learnability Lock (SLL). In *Proceedings of (SIGKDD-UC '22).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph structured data (such as social networks, emails, and financial data [2, 4]) capture the organizational structure and characteristics of multiple objects (nodes) and their relationships (edges). Graph Convolution Networks (GCN) have shown remarkable performance on node classification tasks on graph data, with applications such as targeted marketing, disease spread prediction, etc. [6, 11].

GCNs can also be used by nefarious users to infer information about any node in a dataset, resulting in a possible data breach or privacy violation. Within recent years, increases in the frequency and size of data breaches and privacy violations that expose sensitive information have called for increased privacy protection methods [13]. In 2009, Netflix released anonymized consumer data, removing all identifying attributes (name, email, etc.) except for movie ratings and dates. Despite the removal of information, GCNs were still able to infer sensitive information of the users, violating their privacy [10]. An effective way to address this risk of privacy breach is to alter presence of edges (in small enough quantities that the downstream user is not aware of such modifications, yet enough that patterns in the graph are destroyed) so that a downstream user can not perform effective inference by training a GCN. Graph data poisoning attacks such as Topology Attack [18] or application of noise can accomplish this. However, these methods harm overall downstream GCN performance and reduce the usability of graph structured data for beneficial services mentioned earlier.

There are then two conflicting objectives: 1. Allow graph structured data to be utilized effectively for authorized services, and 2. Prevent malicious downstream users from using GCN to infer private information about users who wish to remain private. Imagine the following scenario: A marketing company collaborates with Facebook to study a product's appeal and uses survey responses to identify target customers. Some users opt-out while others participate and share their information. The company uses the structured graph of users shared by Facebook to train a model to optimize its marketing. The model can accurately classify users as good marketing targets based on their survey responses and relationships with other survey respondents. However, the model is also effective at classifying opted-out users as targets because of the structural dependency among nodes in the graph, violating the opted-out users' privacy expectations. To protect user's privacy, Facebook can take additional steps in the data sharing procedure to limit the abilities of the marketing company. To address the challenge, we propose to study the following research problem: *How can we achieve a graph data protection mechanism that hides patterns of private information on an arbitrarily chosen set of protected nodes, while maintaining data utility such that effective patterns can be learned on authorized nodes?*

There are two major challenges that make this task nontrivial:

- Because users are connected in graph data, anonymizing protected users' information is insufficient - the information of a few connected, authorized users can still be used to recover anonymized information [7, 10].
- A trade-off exists between privacy preserving on protected users and prediction utility on authorized users.

A simple solution to this vulnerability could be to exclude all or partial relationship data from data sharing. However, excluding graph relationship data damages the structure of user network that is necessary for training an accurate GCN, while making it obvious to the malicious downstream user which data was modified. Instead, we develop a graph data protection method, Selective Learnability Lock (SLL), to robustly address the problem.

Our main contributions can be summarized as follows:

- A novel problem setting: partitioning graph nodes into an authorized set and a protected set, where we (the "defender") aim to restrict the ability of a downstream user to train a GCN to infer information about the protected set, while allowing a GCN to learn effectively on the authorized set.
- A robust data protection method, Selective Learnability Lock (SLL), to address the problem. SLL optimizes graph topology to simultaneously minimize and maximize the losses on the two sets of nodes. The product of the method is a "lock": a relatively small collection of edge modifications that, when applied to the topology of the graph data, greatly reduces the ability of a downstream GCN to learn patterns on the protected nodes while having minimal impact on the authorized set compared to a GCN trained on the original graph.

2 RELATED WORKS

Many privacy-focused defense methods on graph data and GCNs have been studied in recent years [16, 19]. Dataset-based methods in recent literature focus primarily on producing significant negative impacts on the capabilities of GCNs trained on altered datasets. Other privacy protection methods, such as model training paradigms, can selectively protect user information from being disclosed by models trained by the data holder.

Privacy protection in graph data structures. A defender attempts to stop an attacker (downstream user) from training effective GCNs on a dataset by making modifications on (attacking) the dataset such that there are misleading/no patterns for a downstream user to utilize. A budget of modifications to data (i.e., number of edges or modifications to features) is used to hide the defense from the downstream user, or attacker. The graph topology attack proposed by Xu et al. demonstrates that a surrogate GCN can be used to approximate a downstream user effectively, and that edge modifications which increase the surrogate's loss are also reflected in the training of the downstream GCN [18]. The mutual information minimization attack by Wang et al. shows other optimization problems that can cause downstream GCNs to have near random-guessing performance [17]. Peng et al. created graph data learnability lock, introducing a method to create a function that can be applied/removed to a graph dataset capable of achieving a similar effect as other dataset-based attacks [12]. However, all of these methods harm downstream GCN performance over the entire graph, not just a selected portion.

Privacy protection in model training. A defender attempts to stop an attacker (downstream user) from recovering sensitive information from GCNs trained by the defender given for public use. In this scenario, there is no concern for budget since the attacker has no information regarding the model nor dataset. The projection unlearning method created by Cong et al. demonstrates an "unlearning" procedure to make a GCN "forget" patterns learned on a specific set of nodes in a dataset [3], allowing a model to have selective predictive performance. The GCN proposed by Hu et al. utilizes a specialized loss function that causes the model to only learn an effective pattern on a specific portion of a graph [5]. Both methods are effective in allowing for selective performance, but accomplish selective learnability in models created by the defender.

In our scenario, the data is provided by the defender to the public, and the downstream user trains a GCN. Our protection method builds on existing methods in a novel problem setting, selectively targeting nodes such that a GCN trained on the data will have poor performance on protected nodes.

3 PROBLEM STATEMENT

In this section, preliminary definitions and notations are given; then the problem setting is clearly defined.

3.1 Graph and GCN notation

Notations. Let an undirected and unweighted graph be notated as $G = (V, E)$ with n nodes $v_i \in V$, and edges $(v_i, v_j) \in E$. We consider undirected graphs in this work and use a symmetric square adjacency matrix A to denote the existence of edges, where $A_{i,j} = A_{j,i} = 1$ if an edge is present between nodes v_i and v_j in G otherwise $A_{i,j} = A_{j,i} = 0$. A set of perturbations to the edges can be expressed as a binary matrix ΔA, where $A_{i,j}$ represents a "flip" or an inversion to A. A modified adjacency matrix can be created by taking the XOR of A and ΔA.

Graph Convolution Network (GCN). Let the Ith layer of feature representation of a neural network be notated as H^i and its weights W^i, σ an activation function such as ReLU. The self-connected adjacency matrix is notated as $\tilde{A}$ and $\tilde{D}$ the diagonal degree matrix. Then a Spectral GCN is a neural network comprised of L layers, where a forward pass on each feature representation layer is done via the following equation [6]:

$$H^{(I+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^I W^I). \tag{1}$$

Let a GCN used for node classification tasks be denoted as Θ. A loss function measures the inaccuracy of the GCN between its predicted node labels, based on a graph G, and their respective true values. The loss is represented as $\mathcal{L}(G, A|\Theta)$.

3.2 Problem Statement

In a graph G, an arbitrarily chosen set of nodes is designated as protected, notated as G_0. All remaining nodes (the set $G - G_0$) are authorized notes, notated as G_X. A set of perturbations, ΔA, is produced by the protection method that can be used to create G' under a budget of ϵ. A downstream GCN Θ' trained on G' will have classification accuracy that is poor on nodes in G_0 and relatively unchanged on nodes in G_X compared to the performance of a GCN Θ trained on the original graph G.

The effectiveness of the protection is determined by the setting–lowering the performance of the downstream GCN to near random-guessing on G_0 is ideal, but for most settings, a large reduction in downstream GCN accuracy can serve as effective deterrent for malicious users. The need for good/unchanged performance on G_X also strongly depends on the application of the dataset.

4 PROPOSED PROTECTION METHOD: SELECTIVE LEARNABILITY LOCK (SLL)

The basis for the attack is that maximizing the loss of a surrogate GCN (approximating the downstream user) over the protected nodes will harm a downstream GCN's performance on the protected nodes, and that minimizing the loss over the authorized nodes will counteract the negative effects on the authorized nodes.

First, a composite loss must be calculated to determine the ideal locations for edge perturbations ΔA. A surrogate GCN Θ, with sensible defaults to mimic a downstream user, is created to calculate the loss of different nodes on the graph. The composite loss is composed of the sum of the cross-entropy loss of nodes in G_0 and the negative of the cross-entropy loss of the nodes in G_X. The adjacency matrix is optimized to minimize the composite loss via gradient descent, while the surrogate GCN is trained further using the modified adjacency matrix ($A' = A \oplus \Delta A$). This bi-level optimization problem can be formulated as such: 0

$$\text{let} \quad A' = A \oplus \Delta A,$$

$$\tilde{\Theta} = \min_{\Theta} \mathcal{L}(G, A'|\Theta), \qquad \text{(inner loop)} \tag{2}$$

$$\min_{|\Delta A| \leq \epsilon} \mathcal{L}(G_X, A'|\tilde{\Theta}) - \mathcal{L}(G_0, A'|\tilde{\Theta}). \qquad \text{(outer loop)} \tag{3}$$

The gradient of the adjacency matrix A w.r.t. the composite loss is projected onto the ball of ϵ, the budget, to create ∇A. The perturbation matrix ΔA is then found by taking Bernoulli samples from ∇A, repeating until the number of modifications present in the perturbation matrix is less than the budget ϵ. The sampling process is formulated as follows

$$\text{let} \quad p = |\text{proj}_{\epsilon} \nabla A|,$$

$$\Delta A_{i,j} = \begin{cases} \text{sign}(\nabla A_{i,j}) * 1, & \text{with } p_{i,j}; \\ 0, & \text{with } 1 - p_{i,j}. \end{cases} \tag{4}$$

4.1 Multitask implementation of SLL

The protection of multiple tasks from a downstream GCN is accomplished by expanding the composite loss function for each task, where the surrogate GCN and loss for tasks $t \in T$ is notated by $\tilde{\Theta}_t$ and $\mathcal{L}_t(\cdot)$, respectively. Then, the composite loss function becomes

$$\min_{|\Delta A| \leq \epsilon} \sum_{t \in T} \mathcal{L}_t(G_X, A'|\tilde{\Theta}_t) - \mathcal{L}_t(G_0, A'|\tilde{\Theta}_t). \tag{5}$$

4.2 Optimizations to SLL

Gradient guidance. The vast majority of ∇A, the gradient of the adjacency matrix, is near-zero. However, many of these near-zero entries will be added to ΔA due to their overwhelming count. Preliminary experiments suggest that the gradient is denser in specific graph edge regions, notably between G_0 and G_X. To avoid useless edge modifications caused by the overwhelming number of near-zero entries, values of ∇A in more useful regions are amplified, while others are reduced. The importance of an edge region is determined according to the moving average of the density of gradients calculated. Three distinct regions in A are designated for this process: the region of possible edges between sets G_X and G_0, within the set G_0, and within the set G_X. Implementation of this "gradient guidance" during the gradient sampling process allows more of the budget to be utilized in more effective areas, improving the overall effectiveness of the protection.

Gradient sampling. The calculation of ∇A is very expensive, yet the majority of the gradient will have little useful information. To reduce the overall memory required, samples of the gradient can be calculated instead. Building on the effectiveness of the gradient guidance method, samples can be drawn at disproportionate rates from each of the graph edge regions mentioned above. The gradient of a much smaller sampled matrix can be calculated multiple times in place of the much larger adjacency matrix A, reducing the maximum memory required.

Algorithm 1: Psuedocode for selective learnability lock protection process with optimizations implemented

```
$\Delta A \leftarrow 0$;
for epochs do
    Train $\tilde{\Theta}$ one iteration with $A \oplus \Delta A$ ; /* Inner loop */
    $loss \leftarrow \mathcal{L}(G_X, A'|\tilde{\Theta}) - \mathcal{L}(G_0, A'|\tilde{\Theta})$ ;  /* Outer loop */
    for s in samples do
        $R \leftarrow n$ random nodes from $G$ ;        /* Gradient sampling */
        $grad \leftarrow \nabla A_R$(w.r.t. $loss$);
    end
    for r in regions of G do
        $grad_r \leftarrow grad_r * bias_r$ ; /* Gradient guidance */
        $bias_r \leftarrow (bias_r + density(grad_r))/2$
    end
    $grad \leftarrow |\mathrm{proj}_\epsilon grad|$;
    while $sum(\Delta A) > \epsilon$ do
        $\Delta A \leftarrow$ Bernoulli sample($grad$);
    end
end
```

4.3 Attack Robustness

There lies a risk of privacy exposure if a downstream user is able to identify the protected set of nodes in a graph affected by SLL. Therefore, SLL must also be analyzed in its robustness to downstream attempts to recover information using the presence of the modifications. However, the identification of protected nodes is not straightforward, as the downstream user only has access to a minimal set of labels for training. Even if the downstream user has access to all labels, it may still not be clear which nodes are part of the protected set, since nodes of the authorized set do not have 100% accuracy, while nodes in the protected set do not have 0% accuracy. Possible approaches to detecting and identifying protected nodes in SLL-affected graphs could utilize strategies to identify network structures within the graph, since it is possible that the underlying patterns of the graph are distinct for the two sets. However, such methods are nontrivial and we leave their further exploration in future work.

5 EXPERIMENTS

5.1 Experimental Setup

For a graph dataset, the protected set G_0 is chosen by randomly selecting 10% of nodes. The rest of the graph, G_X, is designated as the authorized set. A budget ϵ is set regarding the perturbation rate, which is a proportion of the number of edges that are already present in a graph. The effectiveness of SLL is established in a comparison with other possible methods:

- Simplistic 1: Remove all edges between G_X and G_0.
- Simplistic 2: Remove all edges between G_X and G_0 and edges within G_0.
- Noise: Generate a small amount of noise to randomly perturb edges in G_0 within the budget specified by ϵ.
- SLL: Selective Learnability Lock constrained under budget specified by ϵ.
- SLL+G: Selective Learnability Lock with Gradient Guidance constrained under budget specified by ϵ.

SLL is tested on two citation networks (Citeseer [15] and Cora [8]) and three social networks (Polblogs [1], Blogcatalog [14], and flickr [9]). The evaluation of the effectiveness of each protection method is done using GCNs with sensible defaults. Two GCNs with the same neural architectures are trained, one using the original graph data and the other using the "locked" data. The change in predictive performance between the locked and original data is measured over the two partitioned sets of the graph. To quantify the effectiveness of each method, the effectiveness metric $E = -\Delta acc_{G_0} - |\Delta acc_{G_X}|$ is utilized. E is penalized by any change in downstream model accuracy on the set G_X, while reductions in accuracy over G_0 are rewarded. The objective of the methods is to maximize E.

5.2 Experimental Results and Discussion

The results in Figure 2 show that SLL and SLL+G are significantly more effective than other methods at accomplishing the objective. The Simplistic 1 method is mostly ineffective for all datasets, while Simplistic 2 is more effective for smaller datasets such as Polblogs (1,490 nodes) compared to flickr (7,575 nodes). The effectiveness of SLL+G appears more consistent than SLL, outperforming the Noise method for all datasets.

To help understand the effectiveness of SLL, we can examine how homophily has changed ΔH for various methods on a dataset, using Cora as an example. Homophily is a measure of how often similar nodes share an edge, so we expect that homophily would decrease in set G_0, presumably because that would hurt the surrogate GCN's performance on G_0 the most (minimizing the second term of Equation 3), while homophily would stay roughly equal or increase to preserve performance in set G_X (minimizing the first term of Equation 3). The change in homophily ΔH is used. The method SLL* is also tested. SLL* is the implementation of SLL, but the loss equation does not include minimization of loss over set G_X, and is instead as follows:

$$\min_{|\Delta A| \leq \epsilon} -\mathcal{L}(G_0, A'|\tilde{\Theta}). \qquad \text{(outer minimization)} \qquad (6)$$

Because SLL* is not influenced by loss of G_X, we expect that the homophily of G_X will significantly decrease and downstream

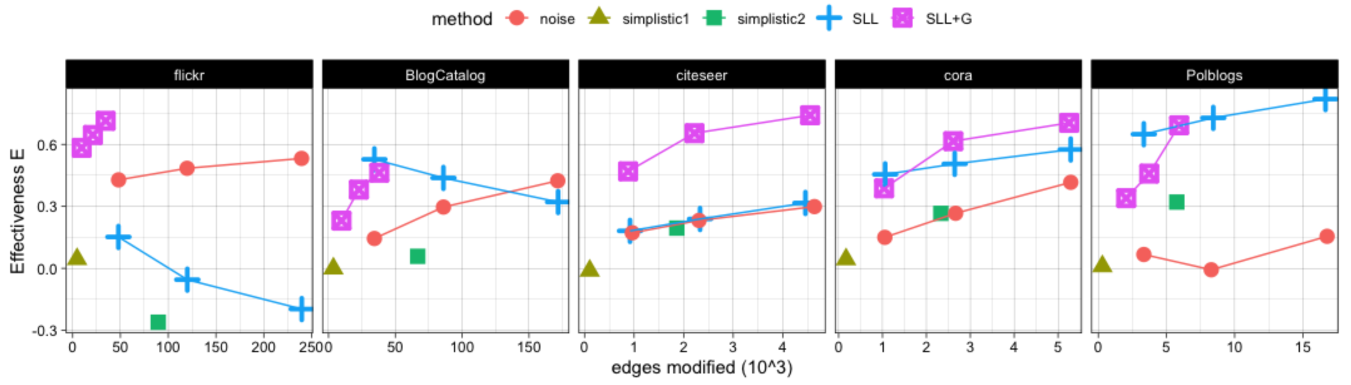

Figure 2: Mean effectiveness E of different methods on datasets by number of edges modified

accuracy over G_X will be poor. This is because the overall structure of the graph is uniform, so changes to maximize harm to G_0 should also ripple negative effects to the entire graph.

Table 1: Effectiveness of various protection methods on Cora dataset with 10% of nodes randomly selected as G_0

		ΔH by set			Δacc by set	
Method	ϵ'	G_0	G_X	Between	G_0	G_X
Simplistic 1	85	-53	0	0	-2.6%	0.0%
Simplistic 2	1165	-53	0	-1080	-26%	-2.1%
Noise in G_0	1330	-914	0	0	-34%	-5.0%
SLL*	1603	-76	-340	-919	-21%	-10%
SLL	1061	-33	-174	-351	-47%	-1.8%
SLL+G	**1346**	**-55**	**-11**	**-1006**	**-70%**	**1.0%**

The implementation of SLL without the loss term to minimize loss on G_X does not perform as well as the complete composite loss-based SLL. This implementation, SLL*, chose to modify more edges between G_0 and G_X to destroy overall graph patterns, and was not as efficient in harming G_0's downstream performance. The experimental results demonstrate that the most efficient edges to target to selectively damage downstream GCN performance are between the protected set and authorized set. SLL focused most of its budget on decreasing homophily between nodes in the protected set and the authorized set. When gradient guidance is implemented, even more budget is utilized in this edge area. Gradient guidance significantly reduced the budget spent on G_X, which is the largest graph edge region. Not much budget was spent on harming homophily within G_0. This is possibly because too much harm to homophily within G_0 would negatively impact the loss in G_X as patterns are destroyed. The changes to graph homophily produced by SLL do not necessarily guarantee worse performance for specific sets of nodes.

Table 2: Effectiveness of gradient guidance on SLL on datasets of varying size and perturbation rate

# nodes	Dataset	Perturbation rate	Ef_{SLL+G}	Ef_{SLL}
7575	flickr	0.1	**61.8**	3.2
		0.25	**30.5**	-0.5
		0.5	**20.7**	-0.8
5196	BlogCatalog	0.1	**23.8**	15.4
		0.25	**16.8**	5.1
		0.5	**12.2**	1.9
3312	citeseer	0.1	**534.2**	197.7
		0.25	**295.0**	102.6
		0.5	**162.8**	71.0
2708	cora	0.1	373.9	**428.8**
		0.25	**235.7**	191.3
		0.5	**134.1**	108.8
1490	Polblogs	0.1	166.1	**194.7**
		0.25	**122.8**	86.4
		0.5	**116.9**	49.2

5.3 Gradient Guidance Experimental Performance

SLL was tested on five datasets with and without gradient guidance implemented. For conciseness, a measure of effectiveness, E is calculated for each experiment as follows: $E = -\Delta acc_{G_0} - |\Delta acc_{G_X}|$. E_{guided} is the effectiveness measure of SLL with gradient guidance, while E_{SLL} is the baseline effectiveness measure of SLL without gradient guidance.

To examine the improvements of gradient guidance, the efficiency of SLL and SLL+G is compared. The efficiency metric $Ef = 10^6 * E/\text{edges modified}$ is used to compare across different numbers of edges modified.

Figure 2 shows that SLL+G was able to achieve higher effectiveness is most datasets. Evidenced by Table 2, gradient guidance provided significant improvements in efficiency at most perturbation rates, especially on datasets of larger size. Smaller datasets (cora,

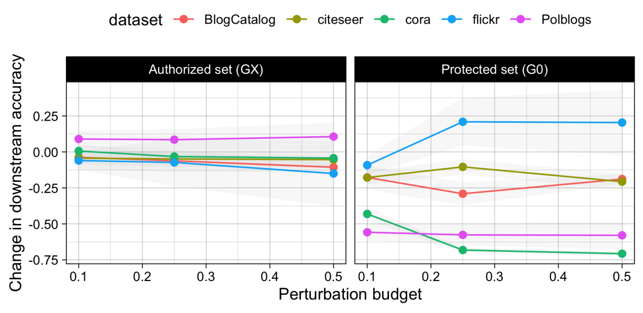

Figure 3: Change in downstream accuracy by location and budget

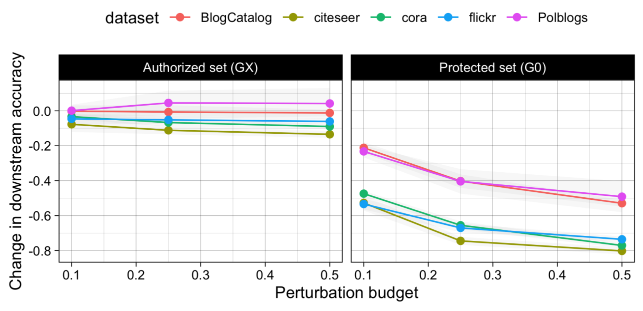

Figure 4: Change in downstream accuracy by location and budget with gradient guidance optimization applied

Polblogs) were impacted less by gradient guidance, while larger datasets (BlogCatalog, citeseer, flickr) saw significant increases in the efficiency of SLL. This is likely due to the gradient sampling process being overwhelmed by the larger adjacency matrix and targeting less important edges to be modified. By implementing the gradient guidance optimization, SLL becomes much more robust, with the change in predictive accuracy becoming more consistent across all datasets regardless of size, as shown in figure 3 and figure 4.

5.4 Multitask Experiment

Two additional classification tasks *FeatA* and *FeatB* are created by discretizing the top 2 highest-entropy features and are incorporated into the SLL loss equation. The features are hidden from the downstream model and surrogate models used in SLL. The classification framework remains the same. Evaluation is performed by separate GCNs, each with sensible defaults, trained for each task.

As shown in 3, SLL can be effective for multiple tasks simultaneously. The effectiveness varies significantly by task, with the label being affected the most. Further exploration into features with different distributions within graphs could further explain why SLL is more effective on some features.

6 CONCLUSION

In this paper, we start by exploring the rising occurrence of data privacy violations relate to GCNs, and how this is in conflict with the ability to provide datasets that are useful for services. With these two conflicting objectives, a new problem setting is proposed - protect an arbirarily chosen subset of nodes in a graph from downstream GCN inference, while allowing the rest of the graph to have good or unchanged inference ability. Previous literature was explored, and techniques that damage graph structure to harm overall downstream GCN performance are evaluated. Existing data protection methods are deemed unable to address the novel problem setting because they affect entire graph performance or are unable to affect a downstream user, so a novel method is proposed: Selective Learnability Lock (SLL). SLL produces a "lock", a collection of edge modifications that can be applied/removed easily from the graph data. SLL uses a surrogate GCN to approximate the downstream users, and optimizes graph topology to minimize loss on the authorized set of nodes of the graph and maximize loss on the protected set. The changes to the loss are reflected in downstream GCNs trained on the modified data. SLL is further improved by "gradient guidance" in the perturbation process, helping to avoid oversampling less important edge regions in larger adjacency matrices. SLL is experimentally effective compared to other possible approaches such as simplistic methods (removing all edges of a certain criteria) and applying random noise, and can be extended to affect multiple tasks (with features serving a tasks in the experiment). SLL primarily reduces homophily between the protected and authorized sets, and slightly reduces graph homophily overall. Investigating the impacts of SLL on similar/orthogonal tasks to label classification that are not directly included in the loss equation could be beneficial to scoping the impact of SLL on graph's learnability.

Table 3: Effectiveness of SLL on multiple tasks of Citeseer and Cora dataset with 10% of nodes randomly selected as G_0

Dataset	Ptb Rate	Edges Modified	Task	E
Citeseer	0.25	2243	Label	0.647
			Feat A	0.071
			Feat B	0.109
Citeseer	0.5	4508	Label	0.735
			Feat A	0.121
			Feat B	0.140
Cora	0.25	2646	Label	0.573
			Feat A	0.134
			Feat B	0.059
Cora	0.5	5279	Label	0.676
			Feat A	0.202
			Feat B	0.052

REFERENCES

[1] Lada A Adamic and Natalie Glance. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*. ACM, 36–43.

[2] Anshika Chaudhary, Himangi Mittal, and Anuja Arora. 2019. Anomaly detection using graph neural networks. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE, 346–350.

[3] Weilin Cong and Mehrdad Mahdavi. 2022. Privacy Matters! Efficient Graph Representation Unlearning with Data Removal Guarantee. (2022).

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

[5] Hui Hu, Lu Cheng, Jayden Parker Vap, and Mike Borowczak. 2022. Learning Privacy-Preserving Graph Convolutional Network with Partially Observed Sensitive Attributes. In *Proceedings of the ACM Web Conference 2022*. 3552–3561.

[6] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[7] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2021. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.

[8] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[9] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 393–401.

[10] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 111–125.

[11] OECD. 2019. *Enhancing Access to and Sharing of Data: Reconciling Risks and Benefits for Data Re-use across Societies*. OECD. https://doi.org/10.1787/276aaca8-en

[12] Weiqi Peng and Jinghui Chen. 2022. Learnability Lock: Authorized Learnability Control Through Adversarial Invertible Transformations. *arXiv preprint arXiv:2202.03576* (2022).

[13] Christian Reimsbach-Kounatze. 2021. Enhancing access to and sharing of data: Striking the balance between openness and control over data. In *Data Access, Consumer Interests and Public Welfare*. Nomos Verlagsgesellschaft mbH & Co. KG, 25–68.

[14] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

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

[16] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. 2018. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528* (2018).

[17] Binghui Wang, Jiayi Guo, Ang Li, Yiran Chen, and Hai Li. 2021. Privacy-Preserving Representation Learning on Graphs: A Mutual Information Perspective. *arXiv:2107.01475 [cs]* (2021). arXiv:2107.01475 http://arxiv.org/abs/2107.01475

[18] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214* (2019).

[19] Carl Yang, Haonan Wang, Ke Zhang, Liang Chen, and Lichao Sun. 2021. Secure deep graph generation with link differential privacy. In *IJCAI*.