# Edge-level privacy in Graph Neural Networks

Rucha Bhalchandra Joshi[1,2][0000−0003−1214−7985] and Subhankar Mishra[1,2][0000−0002−9910−7291]

[1] National Institute of Science Education and Research, Bhubaneswar India
[2] Homi Bhabha National Institute, Mumbai, India
{rucha.joshi,smishra}@niser.ac.in

**Abstract.** The problem of privacy in graph neural networks (GNNs) is being studied recently. It is necessary in order to ensure the privacy of the system that is being modeled as graphs. Currently, the existing models primarily consider the node features and the node labels as privacy information corresponding to the graphs. We propose an edge-privacy preserving methodology called EP-GNN to incorporate the privacy of the structural information of the graphs as well in addition to the features and the label information of the graphs. In this preliminary work, we investigate the impact of the noisy neighborhood on the accuracy of the GNNs. This is in addition to the We experiment with the amount of neighborhood that we perturb and the privacy budget for the edge privacy. We propose two methods to consider the neighborhood, namely 1. $\lambda - selector$ from the neighborhood, and 2. complete neighborhood in order to ensure privacy in the edge data of the graph. We continue to use the node and label privacy as they are implemented in the previous methods for privacy in GNNs.

**Keywords:** Differential Privacy · Graph Neural Networks

## 1 Introduction

The real world systems such as social networks, citation networks, molecular networks are popularly modeled as graphs. A graph richly represents such systems as it considers all the entities in the system as well as relationships between the entities. Graph Neural Networks (GNN) are popularly used to tackle the tasks, such as node classification, graph classification, link prediction are addressed using GNNs. The primary goal of GNNs is the aggregation of structural as well as feature information in an efficient manner while it addresses the tasks related to the systems represented as graphs.

***Problem and Motivation.*** The problem that we address in this work is to ensure the data privacy of the users in critical systems that can be represented using graphs. The term *data privacy* here means privacy related to structure of the graphs and the features of the nodes in the graph. We look at the privacy in GNNs for node-level tasks in this work. Most of the previous work in the area of privacy in GNN has ensured the privacy of the features and labels of each of the nodes. It assumes that the server and the users know the connectivity

information, hence in the previous works, the term data privacy means the privacy of node features and of node labels. In this work we consider the privacy of structural information, typically given by the edges in the graphs, along with the feature and label information of the nodes.

***Challenges.*** In order to ensure the complete privacy that is, the privacy of nodes, edges and the labels in a graph, following challenges need to be addressed: 1. Edge privacy or the privacy of the graph structure is to be taken care of in order to avoid the information from being compromised. The existing methods [12] perturb only the node features and the node labels,however as the structure of the graph remains publicly available, we need consider that as private information as well. 2. To preserve the edge privacy, there has to be a mechanism to perturb the edges in the graph. Deciding the amount of noise to be added to the edges data and the mechanism to add noise is one of the challenges. 3. Determining the amount of edge perturbation, in addition to node and label perturbation to strike the right balance between privacy and utility in the graph data.

***Contributions.*** We propose completely locally private graph neural networks. The complete local privacy is obtained by privatizing the edge information in addition to node and label information. We propose three approaches to privatize the neighborhood of the nodes in the graphs.

1. completely private neighbourhood through edge perturbations.
2. $\lambda - selector$ from neighborhood
3. complete neighborhood, and

We provide experimental evidences of the performance of the model by varying the parameters that controls the noise addition to the edges for perturbations where we perturb the edges according to the neighborhood privatization approaches mentioned above.

***Paper Organization.*** This paper is organized as follows. Introduction, motivation, challenges and contribution of our work are described in section 1. In section 2, we discuss the works related to the topic. Section 3 gives the preliminaries with problem definition and background. Our proposed method is described in section 4. The experimental setup and the results are discussed in section 5 and section 6 concludes the paper.

## 2 Related Works

With different possible attacks on Graph Neural Networks, the privacy of the data involved can be compromised. [6] introduced seven different link stealing attacks on Graphs. The adversary has the black-box access to Graph Neural Networks, and it can infer whether or not any two nodes used in the training of the GNN have a link between them. [17] introduced similar attack called LinkTeller attack that concerns with the privacy of the edges in the graph. In a setting where the node features and the adjacency information are with different parties, the party with the adjacency matrix trains the GNN upon receiving the

node features from the other party and provides back the inference API. The party holding the node features provides the test node features and also can query API for predictions related to test nodes. The LinkTeller attack tries to infer the links present between nodes, based on the queries. The other works [9], [21], [3] discuss the attacks possible on GNNs, such as membership inference attack, graph reconstruction attack and attribute inference attack.

A federated framework for privacy preserving GNNs for recommendation systems is presented in [16]. The GNNs are trained locally at users end and they upload the local gradients to a server for aggregation. To enhance privacy of users by protecting user-item interaction, local differential privacy techniques are applied to the locally computed gradients. [20] is another work involving federated graph neural networks for privacy preserving classification tasks, the features and the edges are split among the users, while all the users have the access to same set of nodes.

To get the differentially private node embeddings, DPNE [18] applies objective perturbation on the objective function of matrix factorization. [12] proposes a privacy preserving GNN learning algorithm for privacy of nodes where the node features and their labels are assumed to be private while the structure of the graph is not private.

## 3 Preliminaries

### 3.1 Problem Definition

**Definition 1 (Graph).** *A graph $G$ is defined as $G = (V, E, X, Y)$ where, $V = V_L \cup V_U$ is a set union of labeled nodes $V_L$ and unlabeled nodes $V_U$ and $|V| = n$. $E$ is a set of edges and $|E| = m$, $X$ is the feature matrix corresponding to the graph where $X \in \mathbb{R}^{n \times d}$ and $d$ is the size of feature vector corresponding to each of the nodes in the graph. $Y \in \{0, 1\}^{n \times c}$, where $c$ is the total number of classes nodes belong to, is the matrix of one-hot encodings of the labels corresponding to labeled nodes and it is a vector of all zeros for the unlabeled nodes in the graph.*

We consider the problem of node classification in a private setting.

Consider a client-server system where the clients are the users represented by nodes and the server is where the Graph Neural Network run and predict the labels of unlabeled nodes. The nodes are aware of their feature vectors and their neighbors. The server is aware only about the vertex set $V$. The GNN needs to be trained for the given task on server side. The problem is how to do train the GNN on server side while preserving neighborhood privacy in addition to preserving privacy of node features and labels?

### 3.2 Background

**Graph Neural Network** Graph neural networks are typically used to answer the questions based on some graph data. The tasks such as node classification, link prediction, graph classification can be addressed using GNNs. A graph $G =$

$(V, E)$ with a matrix feature vectors $X \in \mathbb{R}^{n \times d}$ where $n = |V|$ and $d$ is the size of the feature vector $x_v$ of node $v \in V$. We represent the intermediate embedding of node $v$ at $l$th layer as $h_v^l$. The intermediate representation are then passed to downstream deep network so as to make the desirable predictions related to the task. To get the representation a GNN has two primary steps at each layer for every node: 1. Aggregation of node's neighborhood and 2. Updating the node's embedding. These steps are applied as many times as the number of layers in the GNN.

$$h'_{\mathcal{N}_v} = \textsc{Aggregate}\left(\{h_u^{l-1} : u \in \mathcal{N}(v)\}\right) \tag{1}$$

$$h_v^l = \textsc{Update}\left(\left\{h_v^{l-1}, h'_{\mathcal{N}_v}\right\}\right) \tag{2}$$

In equation 1 above, Aggregate is a differentiable, permutation invariant function that aggregates the feature vectors from layer $l-1$ of the nodes in negihborhood $\mathcal{N}(v)$ of the node $v$. The neighborhood is defined as $\mathcal{N}(v) = \{u : (u, v) \in E\}$. In equation 2, Update is a differentiable function.

**Differential Privacy** Local differential privacy (LDP) enables users to share noise added private data with the untrusted aggregator instead of their true private data, while maintaining considerable accuracy in group queries. It has been used in the last decade for collection of sensitive private data and answering group queries such as statistical mean and count. Companies like Apple, Google and Microsoft have already started including LDP in their products.

**Definition 2 (Local Differential Privacy).** *An algorithm $\mathcal{A}$ satisfies $\epsilon$-local differential privacy ($\epsilon$-LDP), if and only if for any input $x$ and $x'$, we have*

$$\forall y \in Range(\mathcal{A}) : Pr[\mathcal{A}(x) = y] \leq e^{\epsilon} Pr[\mathcal{A}(x') = y] \tag{3}$$

*where Range($\mathcal{A}$) is the set of all possible outputs of the algorithm $\mathcal{A}$ and $\epsilon \geq 0$.*

## 4 Proposed Method

In this section we describe in detail our method to introduce edge-privacy in Graph Neural Networks.

To preserve the privacy in the nodes' neighborhood, we introduce noise into the edge data. We call our method for noise injection the $\lambda$ Selector. It takes the set of neighbors of the node as input for every node in the graph. It also takes in the percentage of noise ($\lambda$) to introduce as input. The $\lambda$ Selector algorithm gives out the noisy neighborhood data as output. The noise added by this algorithm is $\lambda$ percentage of the actual neighborhood of the node. We calculate the amount of noise to add using the degree of the node.

### 4.1 Privacy in Edges

In a locally private setting, a node ideally should decide how much perturbation of data should be done before it shares a data with the central server. To make sure that a node has a complete control over its private information, we need to privatize the neighborhood information, in addition to its feature and label information.

**Naïve EP-GNN** In order to privatize the neighborhood of a node in a naïve way is to perturb the relations of a node with every other node in the graph. We use randomized response as the mechanism to perturb the same.

**Lemma 1.** *Edge perturbation through randomised response satisfies $\epsilon$-local differential privacy [4].*

This approach of perturbing the relations with all the nodes in graph has its own drawbacks. As the randomized response is applied over the entire vector of nodes in the graph, the neighbors of the nodes after perturbation are quite random. The aggregation of the neighborhood leads to a bad approximation of the node's feature values. In order to mitigate this problem in the naïve way of introducing noise, we control the noise added using a parameter called $\lambda$.

**$\lambda$ Selector** We propose a method to introduce noise into the edge data. We call our method $\lambda$ Selector and is given in Algorithm 2. In this method, based on the parameter $\lambda$ we introduce the noise in the edges of a node. In order to guarantee the utility, we need to put the restriction on how much noise a user/node can introduce into the data it is sharing.

The $\lambda$ selector introduces $\lambda$ percentage noise in neighborhood of every node. We calculate the value $\lambda'$ corresponding to every node by multiplying the parameter $\lambda$ with the degree of every node. This essentially means that every node in this private setting can add noise proportionate to $\lambda$. Once every node has their corresponding value of $\lambda'$, it then samples a set $m$ with $d - \lambda'$ many nodes corresponding to the particular node, uniformly at random from its neighborhood without replacement, where $d$ is the degree of the node. Once the neighboring nodes to add noise are selected, we use different methods to introduce the noise. As we want to strike the right balance between the privacy and utility, there is a need of setting the limit on the noise addition. We do this by setting the parameter $\lambda$.

**All neighbourhood** In randomized response, out of the actual neighborhood of a node, we keep $m$ nodes same and we apply the randomized response on all the nodes except those in the set $m$. We apply randomised response with the following:

$$p(a'|a) = \begin{cases} \frac{e^{\epsilon}}{1+e^{\epsilon}} & \text{if } a' = a \\ \frac{1}{1+e^{\epsilon}} & \text{otherwise} \end{cases} \tag{4}$$

The above approach add privacy to the neighbourhood, controlled both by $\epsilon$ and $\lambda$. It is known that larger neighbourhood leads to better node representations [1]. In order to mitigate the low size of conventional neighborhood of nodes in real networks, we take advantage of the KProp algorithm 1 [12].

**Algorithm 1** K-Prop

**Input:** Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; input vector $x_v, \forall v \in \mathcal{V}$; linear aggregator function AGGREGATE; step parameter $K \geq 0$;
**Output:** Embedding vector $h_v, \forall v \in \mathcal{V}$

**for all** $v \in \mathcal{V}$ **do**
  $h^0_{\mathcal{N}(v)} = x_v$
  **for** $k = 1$ to $K$ **do**
    $h^k_{\mathcal{N}(v)} = \text{AGGREGATE}\left(\{h^{k-1}_{\mathcal{N}(u)}, \forall u \in \mathcal{N}(v) - \{v\}\}\right)$
  **end for**
  $h_v = h^K_{\mathcal{N}(v)}$
**end for**
**return** $\{h_v, \forall v \in \mathcal{V}\}$

**Local neighbourhood** In the above case of considering perturbations in the entire neighborhood, there exists the possibility of perturbations in edge list of a node to extend beyond the margins of the KProp. These would lead to addition of distant nodes as neighbours and their significant contribution to final embeddings. This might be as opposed to the scenario, where they might not have had any impact on the embeddings in the ground truth.

We use differentially private clustering algorithms [14, 2, 8] to avoid addition of distant nodes as neighbours and keep the perturbed edge list confined to the local cluster. This along with KProp help us reduce the noise. The upper bound on the privacy budget still remains as $\epsilon$.

**Algorithm 2** $\lambda$ Selector Algorithm

**Input:** $v$: A node in vertex set $V$, $\mathbf{a}$: Set of neighboring nodes of node $v$, $\lambda$: Percentage of noise to add
**Output:** $\mathbf{a}^*$: Set of neighboring nodes with added noise

$d \leftarrow |\mathbf{a}|$
$\lambda' \leftarrow \lambda \times d$
$m \leftarrow$ sampled set of $(d - \lambda')$ nodes from $\mathbf{a}$ uniformly at random without replacement
$r \leftarrow$ a set of $\lambda'$ nodes using the noise-inducing-mechanism
**return** set $m + r$

### 4.2 Privacy in Node Features

Node features are privatized using *multi-bit encoder*, which is built upon 1-bit mechanism, on the user-side. The feature vector corresponding to a node is requested by the server, the multi-bit mechanism is applied on it to get the encoded feature vector which is sent to the server. This gives a biased output which is rectified on server-side using a *multi-bit rectifier*. The differential privacy budget for node feature perturbation is $\epsilon_x$. The estimation error is inversely proportional to the number of neighbors of a node, but in real graphs, the size of the neighborhood is usually small. To overcome this issue 1 layer is used as a first GNN layer in order to denoise the input to the GNN. It considers the $k$-hop neighborhood of a node for denoising the perturned node features at server-side.

### 4.3 Privacy in Node Labels

Each node participating in training has to perturb the labels as they are considered private. This can be done using an LDP mechanism, which in this case is Randomized Response. The perturbed labels are sent to the server where K-Prop is applied on node labels. GCN aggregator is used as the aggregate function in K-prop as it leads to lower estimation error. The differential privacy budget for node feature perturbation is $\epsilon_y$. The label denoising with propagation (Drop) [12] is used for training using perturbed node features and node labels.

**Algorithm 3** Training GNN with Locally Private Edges EP-GNN

**Input:** $G = (V_L \cup V_U, E)$: Graph, GNN algorithm, KProp parameters, $\lambda$ : Percentage of noise in the edges, $\epsilon_e$: Privacy budget for edges
**Output:** $W$: Weights of trained GNN
**Server-side**:
$V \leftarrow V_L \cup V_U$
Send $V$ and KProp parameters to every node $v \in V$
**Node-side**:
Obtain perturbed neighbors $\mathbf{a}^*$
Obtain a perturbed vector $\mathbf{x}^*$ by Multi-Bit Algorithm
**if** current node is in $V$ **then**
    Obtain a perturbed label $y'$ by Randomized Response
**else**
    $y' \leftarrow \vec{0}$
**end if**
Send $\mathbf{a}^*$, $\mathbf{x}^*$ and $y'$ to server.
**Server-side**:
Train GNN using Drop
**return** $W$

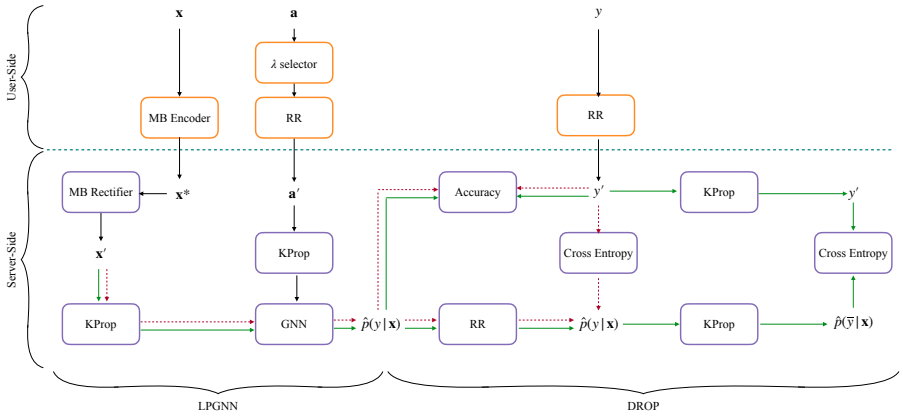


**Fig. 1.** Overview of EP-GNN algorithm with edge privacy in addition to node and label privacy. User's connectivity data is perturbed using $\lambda$ selector and then randomized response can be applied onto it. The new private edges are further passed to the server and used in subsequent steps.

### 4.4 EP-GNN

We combine perturbations in node features, labels and neighbourhood (using the naïve method) with privacy budgets $\epsilon_x$, $\epsilon_y$ and $\epsilon_e$ respectively in Algorithm 3.

**Theorem 1.** *Algorithm 3 satisfies $\epsilon_x + \epsilon_y + \epsilon_e$-local differential privacy.*

Theorem 1 follows from the post-processing composition theorem [13]. We process only the output of the LDP mechanisms on features, labels and edges of the node. LDP mechanism is applied only one time on the nodes and hence any post-processing does not affect differential privacy [4].

## 5 Experiments

#### Dataset Description

We perform the experiments on four popular datasets, namely Cora and Pubmed [19], Facebook [10] and LastFM [11]. The dataset statistics are as given in the table 1.

1. *Cora and Pubmed*[19]: These are citation networks, where each node represent a document and if a document $i$ cites document $j$, then $(i, j) \in E$ and $(j, i) \in E$. The feature vector corresponding to a node is the bag-of-words

representation of the document. The labels are the categories the document belong to.
2. *Facebook*[10]: Nodes in this dataset are verified Facebook pages and edges are the mutual likes between them. The classification is on the site category and the feature vector are extracted from the site description.
3. *LastFM*[11]: Nodes represent the users of radio streaming service LastFM, while the edges represent the friendships among the users. The nodes' features are constituted based on the artists a user likes. The classification task is to predict the home country of the user. The current usage of the dataset is only limited to top 10 countries.

**Table 1.** The statistics of the datasets

| Dataset | Classes | Nodes | Edges | Features | Average Degree |
|---|---|---|---|---|---|
| Cora | 7 | 2708 | 5278 | 1433 | 3.90 |
| Pubmed | 3 | 19717 | 44324 | 500 | 4.50 |
| Facebook | 4 | 22470 | 170912 | 4714 | 15.21 |
| LastFM | 10 | 7083 | 25814 | 7842 | 7.29 |

### 5.1 Experimental Setup

We conduct experiments on the all four datasets to study the effect of the addition of noise in the edge data and the effect of varying privacy budgets for edges in the graph. The experimental setup for privacy of node features and node labels remains similar to that in LPGNN [12].

The nodes in all the datasets are split in the ratios 50:25:25 in training, validation and test sets. The node features are normalized between [0, 1]. The privacy is applied on the graph neural network models namely GCN [7], GAT [15], and GraphSAGE [5]. Feature perturbation as well as edge perturbation is applied during training, validation and testing and the label perturbation is applied during the training and validation only. The privacy budgets for edge perturbation $\epsilon_e$ are taken from the set $\{0.01, 0.1, 0.5, 1, 2\}$. We vary the amount of noise (in percentage) $\lambda$ added to the edges for perturbing the neighborhood of the nodes. We different values of $\lambda$ that we experiment with are from $\{1, 2, 3, 5, 8, 10, 20, 30, 50, 80\}$. The other parameters for training using the Drop are same as that of LPGNN [12]. We use Adam optimizer to train the model and based on validation loss we pick the best model for testing.

### 5.2 Experimental Results

We vary the added noise $\lambda$ and the privacy budget $\epsilon_e$ for the edge perturbation. The privacy-accuracy trade-off is well observed in the results. The noise $\lambda$ decides the amount of neighborhood that will be perturbed. The amount of perturbation

in the selected edges is given by the privacy budget $\epsilon_e$. We see the drop in accuracy as we increase the amount of neighborhood that is perturbed. This is a result of the fact that the $\lambda$ contributes to the privacy of the neighborhood data.

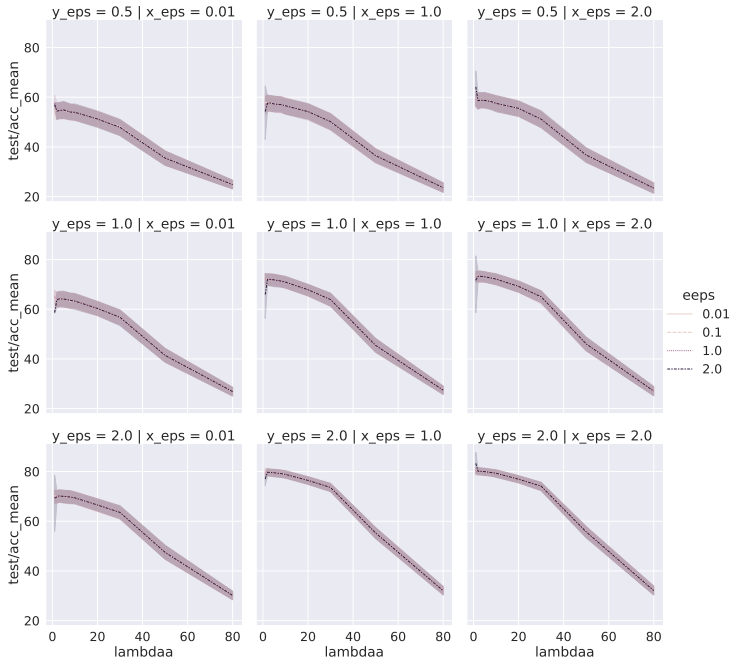


**Fig. 2.** Plots showing the test accuracy for different values of $\lambda$ and $\epsilon_e$

The trend that we observe from the results is similar to that shown in the 2 for different values of $\epsilon_x$ and $\epsilon_y$. From the experiments that we performed, we do not observe a significant drop in accuracy with the increase in privacy budget corresponding to edge data in the graph. The values that we chose for our experiments are small as compared to the size of the neighborhood of the nodes in the datasets we are using. The privacy budget gets divided across the neighborhood. This value in our case is very small and hence there is not a significant drop in accuracy.

## 6 Conclusion

Graphs have been increasingly used to model real world systems including the social interactions. GNNs on graphs create embeddings which enable machine learning algorithms to infer useful insights on such systems. However, real world systems such as social interactions, mobile computing carry a lot of personal and

sensitive data. In order to safeguard the privacy of users, we use one of powerful techniques called differential privacy. Perturbations in node features and labels have been introduced earlier leaving neighbourhood data in open. In this work, we show how perturbations in node neighbourhood affects the accuracy in both embeddings and further machine learning tasks. We introduce $\lambda$ selector to help us tune trade off in amount of perturbations/privacy with the accuracy of the models and perform evaluations across multiple datasets and models. However, this leaves room for future work as initial experiments show drastic drop in accuracy for perturbations. In the future, we would like to work on improving the accuracy while introducing privacy in the edges of the nodes in the graph.

## References


1. Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., Galstyan, A.: Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: international conference on machine learning. pp. 21–29. PMLR (2019)
2. Bun, M., Elias, M., Kulkarni, J.: Differentially private correlation clustering. In: International Conference on Machine Learning. pp. 1136–1146. PMLR (2021)
3. Duddu, V., Boutet, A., Shejwalkar, V.: Quantifying privacy leakage in graph embedding. In: Mobiquitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. pp. 76–85 (2020)
4. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
5. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
6. He, X., Jia, J., Backes, M., Gong, N.Z., Zhang, Y.: Stealing links from graph neural networks. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 2669–2686 (2021)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
8. Mülle, Y., Clifton, C., Böhm, K.: Privacy-integrated graph clustering through differential privacy. In: EDBT/ICDT Workshops. vol. 157 (2015)
9. Olatunji, I.E., Nejdl, W., Khosla, M.: Membership inference attack on graph neural networks. arXiv preprint arXiv:2101.06570 (2021)
10. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding (2021)
11. Rozemberczki, B., Sarkar, R.: Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management (2020)
12. Sajadmanesh, S., Gatica-Perez, D.: Locally private graph neural networks. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2130–2145 (2021)
13. Sengupta, P., Paul, S., Mishra, S.: Learning with differential privacy. In: Handbook of Research on Cyber Crime and Information Privacy, pp. 372–395. IGI Global (2021)
14. Stemmer, U.: Locally private k-means clustering. In: SODA. pp. 548–559 (2020)

15. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
16. Wu, C., Wu, F., Cao, Y., Huang, Y., Xie, X.: Fedgnn: Federated graph neural network for privacy-preserving recommendation. arXiv preprint arXiv:2102.04925 (2021)
17. Wu, F., Long, Y., Zhang, C., Li, B.: Linkteller: Recovering private edges from graph neural networks via influence analysis. In: Proceedings of the Symposium on Security and Privacy (2021)
18. Xu, D., Yuan, S., Wu, X., Phan, H.: Dpne: Differentially private network embedding. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 235–246. Springer (2018)
19. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings (2016)
20. Zhou, J., Chen, C., Zheng, L., Wu, H., Wu, J., Zheng, X., Wu, B., Liu, Z., Wang, L.: Vertically federated graph neural network for privacy-preserving node classification. arXiv preprint arXiv:2005.11903 (2020)
21. Zügner, D., Borchert, O., Akbarnejad, A., Günnemann, S.: Adversarial attacks on graph neural networks: Perturbations and their patterns. ACM Trans. Knowl. Discov. Data **14**(5) (jun 2020). https://doi.org/10.1145/3394520, https://doi.org/10.1145/3394520