NODE-LEVEL DIFFERENTIALLY PRIVATE GRAPH NEURAL NETWORKS

Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, Prateek Jain

Google Research
{ameyasd,gaganmadan,sinhaaditya,athakurta,gagg,prajain}@google.com

ABSTRACT

Graph Neural Networks (GNNs) are a popular technique for modelling graph-structured data and computing node-level representations via aggregation of information from the neighborhood of each node. However, this aggregation implies increased risk of revealing sensitive information, as a node can participate in the inference for multiple nodes. This implies that standard privacy preserving machine learning techniques, such as differentially private stochastic gradient descent (DP-SGD) – which are designed for situations where each data point participates in the inference for one point only – either do not apply, or lead to inaccurate solutions. In this work, we formally define the problem of learning GNN parameters with node-level privacy, and provide an algorithmic solution with a strong differential privacy guarantee. We employ a careful sensitivity analysis and provide a non-trivial extension of the privacy-by-amplification technique. An empirical evaluation on standard benchmarks datasets and architectures demonstrates that our method is indeed able to learn accurate privacy–preserving GNNs which outperform standard non-private methods that completely ignore graph information.

1 INTRODUCTION

Graph Neural Networks (GNNs) (Kipf & Welling, 2016; Veličković et al., 2018; Hamilton et al., 2017; Gilmer et al., 2017) are powerful modeling tools that capture structural information provided by a graph. Consequently, they have become popular in a wide array of domains such as the computational sciences (Ktena et al., 2018; Ahmedt-Aristizabal et al., 2021; McCloskey et al., 2019), computer vision (Wang et al., 2019), and natural language processing (Yao et al., 2019). GNNs allow aggregation of data from the neighbors of a given node in the graph, therefore providing an attractive solution for modeling users – each node of the graph is represented by the user and the connections represent interactions between users – for a variety of recommendation/ranking tasks, where it is challenging to obtain and store user data (Fan et al., 2019; Budhiraja et al., 2020; Levy et al., 2021). However, such solutions are challenging to deploy as they are susceptible to leaking highly sensitive private information of users. Standard ML models – without GNN-style data aggregation – are already known to be highly susceptible to leakage of sensitive information about the training data (Carlini et al., 2019). But, the risk of leakage is even higher in GNNs as each prediction is based on the node itself and aggregated data from it's neighborhood. In fact, there are two types of highly-sensitive information about an individual node that can be leaked: a) the features associated with each node/user, b) the connectivity information of an individual node/user.

In this work, we study the problem of designing algorithms to learn GNNs while preserving *node*-level privacy. We use differential privacy as the notion of privacy (Dwork et al., 2006) of a node, which requires that the algorithm should learn roughly similar GNNs despite perturbation of an entire node and *all* the data points associated with that node. Our proposed method preserves the privacy of the features of each node ('user'), their labels as well as their connectivity information. Our method adapts the standard DP-SGD method (Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016) to the node-level privacy setting. But, analysis of the standard DP-SGD method does not directly extend to GNNs, as each *gradient* term in GNNs can depend on multiple nodes. The key technical contribution of our work is two-fold: **(1)** we propose a neighborhood sampling scheme

in-line with a careful sensitivity analysis for multi-layer GNNs, **(2)** we extend the standard privacy by amplification technique to GNNs where one gradient term can depend on multiple users.

2 Related Work

Differentially Private SGD (DP-SGD) (Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016) has been used successfully to train neural network models to classify images (Abadi et al., 2016) and text (Anil et al., 2021), by augmenting the standard paradigm of gradient-based training to be differentially private. Edge-level privacy in GNN's ensures that the existence of an edge between user i and user j does not impact the output significantly (Wu et al., 2021b). However, such methods do not cover the case of perturbing entire user i's data, which is practical as in many cases we need to preserve privacy of *all* of a user's data. Private GNNs have also been studied from the perspective of local privacy (Sajadmanesh & Gatica-Perez, 2020), where each node performs its share of the GNN computation locally and sends noisy versions of its data to neighbouring nodes so as to learn shared weights; such algorithm needs to correct for the bias in both the features and labels. The analysis of this method only applies to GNNs with linear neighborhood aggregation functions. In contrast, the methods we propose can be employed with several practical GNN classes. (Wu et al., 2021a) utilizes private GNNs for recommendation systems, but their method assumes a bipartite graph structure, and cannot naturally handle homogeneous graphs. Other approaches employ federated learning (Zhou et al., 2020), but only guarantee that the GNN neighbourhood aggregation step is differentially private, which is insufficient to guarantee privacy of each node's neighborhood. Finally, several papers provide privacy-preserving GNNs (Shan et al., 2021) but these do not use the formal notion of DP and provide significantly weaker privacy guarantees. In different contexts, there has been extensive work on node-level DP (Raskhodnikova & Smith, 2016; Karwa et al., 2011; Borgs et al., 2015; 2018). But these methods generally estimate 'global' graph-level statistics and do not support learning methods such as GNNs. In contrast, our approach *predicts* 'local' node-level statistics (such as the label of a node) while preserving node-level privacy.

3 Problem Formulation and Preliminaries

Consider a graph dataset $G = (V, E, \mathbf{X}, \mathbf{Y})$ with *directed* graph $\mathcal{G} = (V, E)$ represented by a adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$. n is the number of nodes in $\mathcal{G}$, V denotes the node set, E denotes the edge set. Each node v in the graph is equipped with a feature vector $\mathbf{X}_v \in \mathbb{R}^d$; $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the feature matrix. $\mathbf{Y} \in \mathbb{R}^{n \times Q}$ is the label matrix and $\mathbf{y}_v$ is the label for the v-th node over Q classes. Note that many of the labels in the label vector can be missing, which models the semi-supervised setting. In particular, we assume that node labels $\mathbf{y}_v$ are only provided for a subset of nodes $V_{tr} \subset V$, called the training set. Given the graph dataset G, the goal is to learn parameters of a GNN while preserving privacy of individual nodes. A 1-layer GNN can be represented by the following operations:

$$\widehat{\mathbf{y}}_v = \mathsf{GNN}(\mathbf{A}, \mathbf{X}, v; \mathbf{\Theta}) := f_{\text{dec}}\left(f_{\text{agg}}\left(\{f_{\text{enc}}(\mathbf{X}_u) \mid \mathbf{A}_{vu} \neq 0\}\right)\right) \tag{1}$$

where $\widehat{\mathbf{y}}_v$ is the prediction from the GNN for a given node v, f_{enc} is the encoder function that encodes node features with parameters Θ_{enc}, f_{agg} is the neighborhood aggregation function with parameters Θ_{agg}, f_{dec} is the prediction decoder function with parameters Θ_{dec}, and $\Theta := (\Theta_{\text{enc}}, \Theta_{\text{agg}}, \Theta_{\text{dec}})$.

While our results apply to most classes of GNN models (Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2018), for simplicity, we focus on 1-layer and 2-layer Graph Convolutional Network (GCN) (Kipf & Welling, 2016) with additional results for the Graph Isomorphism Network (GIN) (Xu et al., 2018) and the Graph Attention Network (GAT) (Veličković et al., 2018) in the Appendix. Thus, 'learning' a GNN is equivalent to finding parameters $\Theta := (\Theta_{\text{enc}}, \Theta_{\text{agg}}, \Theta_{\text{dec}})$ that minimize a suitable loss:

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}(G, \Theta) := \sum_{v \in V} \ell(\widehat{\mathbf{y}}_v; \mathbf{y}_v), \tag{2}$$

where $\ell : \mathbb{R}^{Q \times Q} \to \mathbb{R}$ is a standard loss function such as categorical or sigmoidal cross-entropy.

Definition 1 (Adjacent Graph Datasets)**.** *Two graph datasets G and G' are said to be* **node-level adjacent** *if one can be obtained by adding or removing a node (with its features, labels and associated edges) to the other. That is, G and G' are exactly the same except for the v-th node, i.e., $\mathbf{X}_v$, $\mathbf{y}_v$ and $\mathbf{A}_v$ differ in the two datasets.*

Informally, $\mathcal{A}$ is said to be node-level differentially-private if the addition or removal of a node in $\mathcal{A}$'s input does not affect $\mathcal{A}$'s output significantly.

Definition 2 (Node-level Differential Privacy)**.** *Consider any randomized algorithm $\mathcal{A}$ that takes as input a graph dataset. $\mathcal{A}$ is said to be (α, γ)* **node-level Rényi differentially-private** *(Mironov, 2017) if, for every pair of node-level adjacent datasets G and G': $D_\alpha(\mathcal{A}(G) \parallel \mathcal{A}(G')) \leq \gamma$ where* **Rényi divergence** *D_α of order α between two random variables P and Q is defined as $D_\alpha(P \parallel Q) = \frac{1}{\alpha-1} \ln \mathbb{E}_{x\sim Q} \left[\frac{P(x)}{Q(x)}\right]^\alpha$.*

Note that we use Rényi differentially-private (RDP) (Mironov, 2017) as the formal notion of differential privacy (DP), as it allows for tighter composition of DP across multiple steps. This notion is closely related to the standard (ε, δ)-differential privacy (Dwork et al., 2006); Proposition 3 of Mironov (2017) states that any (α, γ)-RDP mechanism also satisfies $(\gamma + \frac{\log 1/\delta}{\alpha-1}, \delta)$-differential privacy for any $\delta \in (0, 1)$. Thus, we seek to find Θ by optimizing Equation 2 while ensuring RDP.

Definition 3. *The K-***restricted node-level sensitivity** *$\Delta_K(f)$ of a function f defined on graph datasets is $\Delta_K(f) = \max_{\substack{G,G' \text{ node-level adjacent} \\ \text{in-deg}(G),\ \text{in-deg}(G') \leq K}} \|f(G) - f(G')\|_2$.*

4 Learning Graph Neural Networks (GNN) via DP-SGD

In this section, we provide a variant of DP-SGD (Bassily et al., 2014) designed specifically for GNNs, and show that our method guarantees node-level DP (Definition 2). Assuming we are running a r-layer GNN, first we subsample the local r-hop neighborhood of each node to ensure that each node has a bounded number of neighbors and influences a small number of nodes. Next, similar to the standard mini-batch SGD technique, we sample a subset $\mathcal{B}_t$ of m subgraphs chosen uniformly at random from the set $\mathcal{S}_{tr}$ of training subgraphs. In contrast to the standard mini-batch SGD, that samples points with replacement for constructing a mini-batch, our method samples mini-batch $\mathcal{B}_t$ uniformly from the set of all training subgraphs. This distinction is important for our privacy amplification result. Once we sample the mini-batch, we apply the standard DP-SGD procedure of computing the gradient over the mini-batch, clipping the gradient and adding noise to it, and then use the noisy gradients for updating the parameters.

However, DP-SGD requires each update to be differentially private. In standard settings where each gradient term in the mini-batch corresponds to only one point, we only need to add $O(C)$ noise – where C is the clipping norm of the gradient – to ensure privacy. However, in the case of GNNs with node-level privacy, perturbing one node/point $\widehat{\mathbf{v}}$ can have impact on the loss terms corresponding to all its neighbors. Thus, to ensure the privacy of each update, we add noise according to the sensitivity of aggregated gradient: $\nabla_{\mathbf{\Theta}} \mathcal{L}(\mathcal{B}_t; \mathbf{\Theta}_t) := \sum_{S_v \in \mathcal{B}_t} \text{Clip}_C(\nabla_{\mathbf{\Theta}} \ell\left(\mathsf{GNN}(\mathbf{A}, \mathbf{X}, v; \mathbf{\Theta}_t); \mathbf{y}_v\right))$ with respect to any individual node $\widehat{\mathbf{v}}$, which we bound via careful subsampling of the input graph. In traditional DP-SGD, a crucial component in getting a better privacy/utility trade-off over just adding noise according to the sensitivity of the minibatch gradient, is privacy amplification by sampling (Kasiviswanathan et al., 2008; Bassily et al., 2014). This says that if an algorithm $\mathcal{A}$ is ε-DP on a data set D_1, then on a random subset $D_2 \subseteq D_1$ it satisfies roughly $\frac{|D_2|}{|D_1|}(e^\varepsilon - 1)$-DP. Unlike traditional ERM problems, we cannot directly use this result in the context of GNNs. The reason is again that on two adjacent data sets, multiple loss terms corresponding to $\widehat{\mathbf{v}}$ and its r-hop neighbors $\mathcal{N}^{(r)}_{\widehat{\mathbf{v}}}$ get modified. To complicate things further, the minibatch $\mathcal{B}_t$ that gets selected may only contain a small random subset of $\mathcal{N}^{(r)}_{\widehat{\mathbf{v}}}$. To address these issues, we provide a new privacy amplification theorem (Theorem 1). To prove the theorem, we adapt (Feldman et al., 2018, Lemma 25) – that shows a weak form of convexity of Renyi divergence – for our specific instance, and provide a tighter bound by exploiting the special structure in our setting along with the above bound on sensitivity.

Theorem 1 (Amplified Privacy Guarantee for any 1-Layer GNN)**.** *Consider the loss function $\mathcal{L}$ of the form: $\mathcal{L}(G, \mathbf{\Theta}) = \sum_{v \in V_{tr}} \ell\left(\mathsf{GNN}(\mathbf{A}, \mathbf{X}, v; \mathbf{\Theta}_t); \mathbf{y}_v\right)$.*

For any choice of the noise standard deviation $\sigma > 0$ and clipping threshold C, every iteration t of Algorithm 1 is (α, γ) node-level Rényi DP, where:

$$\gamma = \frac{1}{\alpha - 1} \ln \mathbb{E}_\rho \left[\exp\left(\alpha(\alpha-1) \cdot \frac{2\rho^2 C^2}{\sigma^2}\right)\right],\ \rho \sim \text{Hypergeometric}\left(N, \frac{K^{r+1}-1}{K-1}, m\right).$$

Algorithm 1: DP-GNN (SGD): Training Differentially Private Graph Neural Networks with SGD

Data: Graph $G = (V, E, \mathbf{X}, \mathbf{Y})$, GNN model GNN, Number of GNN layers r, Training set V_{tr}, Loss function $\mathcal{L}$, Batch size m, Maximum in-degree K, Learning rate η, Clipping threshold C, Noise standard deviation σ, Maximum training iterations T.

Result: GNN parameters $\mathbf{\Theta}_T$.

Note that V_{tr} is the subset of nodes for which labels are available (see Paragraph 1 of Section 3).

Construct the set of training subgraphs with Algorithm 4:

$\mathcal{S}_{tr} \leftarrow \mathsf{SAMPLE-SUBGRAPHS}(G, V_{tr}, K, r)$.

Initialize $\mathbf{\Theta}_0$ randomly.

for $t = 0$ **to** T **do**

  Sample set $\mathcal{B}_t \subseteq \mathcal{S}_{tr}$ of size m uniformly at random from all subsets of $\mathcal{S}_{tr}$.

  Compute the update term $\mathbf{u}_t$ as the sum of the clipped gradient terms in the mini-batch $\mathcal{B}_t$:

  $\mathbf{u}_t \leftarrow \sum_{S_v \in \mathcal{B}_t} \mathrm{Clip}_C(\nabla_{\mathbf{\Theta}} \ell\left(\mathsf{GNN}(\mathbf{A}, \mathbf{X}, v; \mathbf{\Theta}_t); \mathbf{y}_v\right))$

  Add independent Gaussian noise to the update term: $\tilde{\mathbf{u}}_t \leftarrow \mathbf{u}_t + \mathcal{N}(0, \sigma^2 \mathbb{I})$

  Update the current estimate of the parameters with the noisy update: $\mathbf{\Theta}_{t+1} \leftarrow \mathbf{\Theta}_t - \frac{\eta}{m} \tilde{\mathbf{u}}_t$

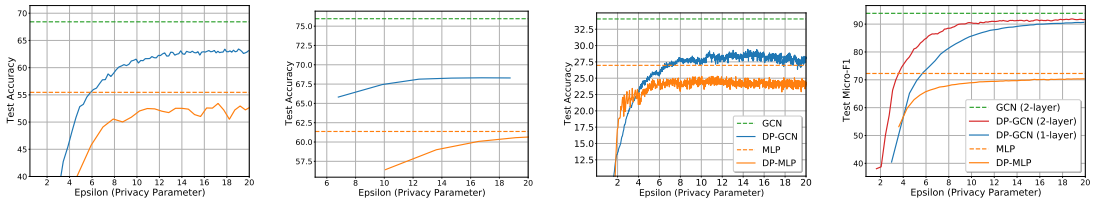
end

Hypergeometric *denotes the standard hypergeometric distribution (Forbes et al., 2011). By the standard composition theorem for Rényi Differential Privacy (Mironov, 2017), over T iterations, Algorithm 1 is $(\alpha, \gamma T)$ node-level Rényi DP, where γ and α are defined above.*

We similarly adapt a DP version of the Adam (Kingma & Ba, 2014) optimizer to the GNN setting, called DP-GNN (Adam), with the same privacy guarantees as DP-GNN (SGD).

5 Experimental Results

In this section, we present empirical evaluation of our method on standard benchmark datasets for large graphs such as datasets from the Open Graph Benchmark (OGB) suite (Hu et al., 2020) and GraphSAGE (Hamilton et al., 2017), and evaluate our method in both transductive and inductive settings. The goal is to demonstrate that our method (DP-GNN) can indeed learn privacy preserving GNNs accurately. The main benchmark of our evaluation is to demonstrate that DP-GNN is able to provide *more accurate solutions* than standard methods that completely discard the graph information. In particular, we benchmark the following methods: a) **DP-GCN**: Our DP-GNN method (Algorithm 1) applied to a 1-layer GCN (in the transductive and inductive settings) and a 2-layer GCN (in the inductive settings) with an MLP as the encoder and the decoder, b) **GCN**: A 1-layer GCN (in transductive and inductive settings) and a 2-layer GCN (in inductive settings) with MLP as the encoder and decoder. This would in general be more accurate than our method but due to privacy concerns, might not be suitable for several real-world applications, c) **MLP**: A standard multi-layer perceptron (MLP) architecture on the raw node features as proposed in prior works (Hu et al., 2020), which does not utilize the graph information, d) **DP-MLP**: A DP version of MLP (with standard architecture) trained using DP-Adam.



(a) **(b)** **(c)** **(d)**

Figure 1: (a), (b), (c): Performance of the 1-layer DP-GCN models and baselines with respect to privacy budget ε on ogbn-arxiv, ogbn-products and ogbn-mag datasets. (d): Performance of the 1-layer and 2-layer DP-GCN models on the reddit-disjoint dataset.

Table 1: **Test performance of DP-GCN in the transductive setting, with privacy budget $\varepsilon \leq 20$.**

Model	ogbn-arxiv	ogbn-products	ogbn-mag	reddit
GCN (1-layer)	67.758 ± 0.418	75.965 ± 0.374	34.074 ± 0.445	94.074 ± 0.074
DP-GCN (Adam)	62.917 ± 0.308	68.374 ± 0.106	29.078 ± 0.202	92.807 ± 0.050
DP-GCN (SGD)	63.015 ± 0.562	67.230 ± 0.031	29.354 ± 0.298	92.787 ± 0.139
MLP	55.236 ± 0.317	61.364 ± 0.132	26.969 ± 0.361	72.359 ± 0.141
DP-MLP	52.942 ± 0.413	60.572 ± 0.104	25.259 ± 0.321	70.273 ± 0.087

5.1 Results in the Transductive Setting

We first study the transductive setting where at inference time, each test node has access to the features of its neighbors. Recall that we focus on 1-layer GNNs in this setting. Table 1 compares the performance of DP-GCN against baselines on the ogbn-arxiv, ogbn-products, ogbn-mag and reddit-transductive datasets. Overall, we observe that our proposed method DP-GCN significantly outperforms the Non-Private MLP (without any usage of the graphs) and DP-MLP (trained using standard DP-Adam) baselines on all of the datasets and with a reasonable privacy budget of $\varepsilon \leq 20$. For example, for ogbn-arxiv dataset, our method DP-GCN (SGD) is about 8% more accurate than MLP and 10% more accurate than DP-MLP. Similarly, for ogbn-products our method is about 5% more accurate than both MLP and DP-MLP and for reddit, our method is about 20% more accurate than MLP and 22% more accurate than DP-MLP. Figure 1 provides a comparison of epsilon versus test set performance for the three benchmark datasets. On all datasets, DP-GCN (Adam) outperforms both MLP and DP-MLP for a privacy budget of $\varepsilon \leq 10$.

5.2 Results in the Inductive Setting

Additionally, we consider the 'inductive' setting where the test dataset (i.e. the nodes and the graph) are completely disjoint from the training data nodes and the associated graph. This models situations such as multi-enterprise models where graph over users of one enterprise is completely disjoint from the graph over another enterprise's users. To conduct these experiments, we divide the nodes into three splits – training, validation and test – and remove all inter-split edges to partition the graph into disjoint subgraphs. We report results on three datasets: ogbn-arxiv-disjoint (where inter-split edges in ogbn-arxiv have been removed), ogbn-arxiv-clustered (where agglomerative clustering is perfomed on the original ogbn-arxiv dataset to partition the nodes) and reddit-disjoint (where inter-split edges in reddit-transductive have been removed). We also investigate 2-layer DP-GCNs in this setting. Once the DP-GNN parameters have been learnt privately over the training graph, we assume that the test graph and test nodes are available non-privately to the inference algorithm. Table 2 presents accuracy of our DP-GCN method with 1-layer and 2-layer GCN models on three datasets. We observe that DP-GCN is significantly more accurate than MLP as well as DP-MLP method which completely ignore the graph features. Furthermore, we observe that an additional GCN layer is able to provide a significant gain in accuracy on all three datasets for both the non-private as well as private GCN methods.

Table 2: **Test performance of DP-GCN in the inductive setting, with privacy budget $\varepsilon \leq 20$.**

Model	ogbn-arxiv-disjoint	ogbn-arxiv-clustered	reddit-disjoint
GCN (2-layer)	60.641 ± 0.417	56.932 ± 0.864	93.775 ± 0.116
GCN (1-layer)	60.570 ± 0.438	54.188 ± 0.761	92.358 ± 0.815
DP-GCN (2-layer)	57.304 ± 0.240	43.870 ± 2.303	91.538 ± 0.321
DP-GCN (1-layer)	56.714 ± 0.281	43.522 ± 0.642	90.618 ± 0.073
MLP	55.460 ± 0.188	37.764 ± 1.014	72.272 ± 0.132
DP-MLP	52.822 ± 0.476	34.768 ± 2.524	70.298 ± 0.118

6 Conclusions and Future Work

In this work, we proposed a method to privately learn r-layer GNN parameters, that outperforms both private and non-private baselines that do not utilize graph information. Our method ensures node-level differential privacy, by a careful combination of sensitivity analysis of the gradients and a privacy amplification result extended to the GNN setting. We believe that our work is a first step in the direction of designing powerful GNNs while preserving privacy. Some promising avenues for future work include extending the DP-GNN method to learn non-local GNNs, and understanding utility bounds for GNNs with node-level privacy.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. Graph-Based Deep Learning for Medical Diagnosis and Analysis: Past, Present and Future. *arXiv preprint arXiv:2105.13137*, 2021.

Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-Scale Differentially Private BERT. *CoRR*, abs/2108.01624, 2021. URL https://arxiv.org/abs/2108.01624.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pp. 464–473, 2014.

Christian Borgs, Jennifer T. Chayes, and Adam Smith. Private Graphon Estimation for Sparse Graphs, 2015.

Christian Borgs, Jennifer Chayes, Adam Smith, and Ilias Zadik. Revealing Network Structure, Confidentially: Improved Rates for Node-Private Graphon Estimation, 2018.

Amar Budhiraja, Gaurush Hiranandani, Darshak Chhatbar, Aditya Sinha, Navya Yarrabelly, Ayush Choure, Oluwasanmi Koyejo, and Prateek Jain. Rich-Item Recommendations for Rich-Users: Exploiting Dynamic and Static Side Information, 2020.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 267–284, Santa Clara, CA, August 2019. USENIX Association. ISBN 978-1-939133-06-9. URL https://www.usenix.org/conference/usenixsecurity19/presentation/carlini.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In Shai Halevi and Tal Rabin (eds.), *Theory of Cryptography*, pp. 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-32732-5.

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

Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy Amplification by Iteration. In *59th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 521–532, 2018.

C. Forbes, M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions*. Wiley, 2011. ISBN 9781118097823. URL https://books.google.co.in/books?id=YhF1osrQ4psC.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. *CoRR*, abs/1704.01212, 2017. URL http://arxiv.org/abs/1704.01212.

Josef Hadar and William R. Russell. Rules for ordering uncertain prospects. *The American Economic Review*, 1969. ISSN 00028282.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. *CoRR*, abs/1706.02216, 2017. URL http://arxiv.org/abs/1706.02216.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Vishesh Karwa, Sofya Raskhodnikova, Adam Davison Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, August 2011. ISSN 2150-8097.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What Can We Learn Privately? In *49th Annual IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 531–540, 2008.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs/1609.02907, 2016. URL http://arxiv.org/abs/1609.02907.

Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169:431–442, 2018.

Daniel Levy, Ziteng Sun, Kareem Amin, Satyen Kale, Alex Kulesza, Mehryar Mohri, and Ananda Theertha Suresh. Learning with User-Level Privacy. *CoRR*, abs/2102.11845, 2021. URL https://arxiv.org/abs/2102.11845.

Kevin McCloskey, Ankur Taly, Federico Monti, Michael P Brenner, and Lucy J Colwell. Using attribution to decode binding mechanism in neural network models for chemistry. *Proceedings of the National Academy of Sciences*, 116(24):11624–11629, 2019.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275. IEEE, 2017.

Sofya Raskhodnikova and Adam Smith. Lipschitz Extensions for Node-Private Graph Statistics and the Generalized Exponential Mechanism. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 495–504, 2016. doi: 10.1109/FOCS.2016.60.

Sina Sajadmanesh and Daniel Gatica-Perez. When Differential Privacy Meets Graph Neural Networks. *CoRR*, abs/2006.05535, 2020. URL https://arxiv.org/abs/2006.05535.

Chuanqiang Shan, Huiyun Jiao, and Jie Fu. Towards Representation Identical Privacy-Preserving Graph Neural Network via Split Learning. *CoRR*, abs/2107.05917, 2021. URL https://arxiv.org/abs/2107.05917.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pp. 245–248. IEEE, 2013.

Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pp. 347–450. Springer, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks, 2018.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *Acm Transactions On Graphics)*, 38(5): 1–12, 2019.

Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. FedGNN: Federated Graph Neural Network for Privacy-Preserving Recommendation. *CoRR*, abs/2102.04925, 2021a. URL https://arxiv.org/abs/2102.04925.

Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis, 2021b.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? *CoRR*, abs/1810.00826, 2018. URL http://arxiv.org/abs/1810.00826.

Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7370–7377, 2019.

Jun Zhou, Chaochao Chen, Longfei Zheng, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. Privacy-Preserving Graph Neural Network for Node Classification. *CoRR*, abs/2005.11903, 2020. URL https://arxiv.org/abs/2005.11903.

APPENDIX

A SAMPLING SUBGRAPHS WITH OCCURRENCE CONSTRAINTS

To bound the sensitivity of the mini-batch gradient in Algorithm 1, we must carefully bound the maximum number of occurrences of any node in the graph across all training subgraphs. To ensure that these constraints are met for any r-layer GNN, we propose $\mathsf{SAMPLE}-\mathsf{SUBGRAPHS}$ (Algorithm 4) to output a set of training subgraphs. Note that the common practice (Hamilton et al., 2017) of sampling to restrict the out-degree of every node is insufficient to provide such a guarantee, as the in-degree of a node (and hence, the number of occurrences of that node in other subgraphs) can be very large. Note that once the model parameters have been learnt, no restrictions are needed at inference time. This means GNN predictions for the 'test' nodes can use the entire neighbourhood.

Algorithm 2: $\mathsf{SAMPLE}-\mathsf{EDGELISTS}$: Sampling the Adjacency Matrix with In-Degree Constraints

Data: Graph $G = (V, E, \mathbf{X}, \mathbf{Y})$, Training set V_{tr}, Maximum in-degree K.
Result: Set of sampled edgelists $\mathbf{E}_v$ for each node $v \in V$.
for $v \in V$ **do**
  Construct the incoming edgelist over training set: $\mathbf{RE}_v \leftarrow \{u \mid (u, v) \in E \text{ and } u \in V_{tr}\}$
  Sample incoming edgelists. Each edge is sampled independently with a probability $p = \frac{K}{2|\mathbf{RE}_v|}$: $\mathbf{RE}_v \leftarrow \text{sample}(\mathbf{RE}_v)$
  The nodes with in-degree greater than K are dropped.[1]
end
for $v \in V$ **do**
  Reverse incoming edgelists to get sampled edgelists $\mathbf{E}_v$: $\mathbf{E}_v \leftarrow \{u \mid v \in \mathbf{RE}_u\}$
end
return $\{\mathbf{E}_v \mid v \in V\}$.

Algorithm 3: $\mathsf{DFS}-\mathsf{TREE}$: Depth-First-Search Tree with Depth Constraints

Data: Root node v, Edgelists $\mathbf{E}$, Maximum depth r.
Result: Subgraph S_v representing the r-depth DFS tree rooted at v.
If $r = 0$, **return** $\{v\}$.
Add edges from v to its neighbours' subgraphs:
$S_v \leftarrow \{v\} \cup \{\mathsf{DFS}-\mathsf{TREE}(u, \mathbf{E}, r-1) \mid u \in \mathbf{E}_v\}$
return S_v.

Algorithm 4: $\mathsf{SAMPLE}-\mathsf{SUBGRAPHS}$: Sampling Local Neighborhoods with Occurrence Constraints

Data: Graph $G = (V, E, \mathbf{X}, \mathbf{Y})$, Training set V_{tr}, Maximum in-degree K, GNN layers r.
Result: Set of subgraphs S_v for each node $v \in V_{tr}$.
Obtain set of sampled edgelists: $\mathbf{E} \leftarrow \mathsf{SAMPLE}-\mathsf{EDGELIST}(G, V_{tr}, K)$
for $v \in V_{tr}$ **do**
  $S_v \leftarrow \mathsf{DFS}-\mathsf{TREE}(v, \mathbf{E}, r)$
end
return $\{S_v \mid v \in V_{tr}\}$.

Lemma 1 ($\mathsf{SAMPLE}-\mathsf{SUBGRAPHS}$ Satisfies Occurrence Constraints)**.** *Let G be any graph with set of training nodes V_{tr}. Then, for any $K, r \geq 0$, the number of occurrences of any node in the set of training subgraphs $\mathsf{SAMPLE}-\mathsf{SUBGRAPHS}(G, V_{tr}, K, r)$ is bounded above by $N(K, r)$, where:*

$$N(K, r) = \sum_{i=0}^{r} K^i = \frac{K^{r+1} - 1}{K - 1} \in \Theta(K^r)$$

[1] For simplicity, we implement this step in a non-DP manner, but it can be easily made DP by employing the Propose-Test-Release mechanism (Vadhan, 2017) assuming public knowledge of the degree.

A.1 Results with other GNN Architectures

As mentioned in Section 4, the DP-GNN training mechanisms can be used with most r-layer GNN architectures. We experiment with two more GNN architectures, namely GIN (Xu et al., 2018) and GAT (Veličković et al., 2018) in both transductive (Table 3) and inductive (Table 4) settings.

We observe that DP-GNN performs well across different architectures in both privacy settings, outperforming MLP and DP-MLP baselines in all cases. For both the inductive and transductive settings, we observe that GIN performs similarly to GCN and DP-GIN again has similar performance as DP-GCN. On the ogbn-arxiv-clustered dataset, however, both 1-layer and 2-layer DP-GIN models perform much better than their DP-GCN counterparts.

Table 3: **Test accuracy of DP-GIN and DP-GAT on the transductive ogbn-arxiv dataset with a privacy budget of** $\varepsilon \leq 20$.

Model	Non-Private GNN	DP-GNN
GCN	67.758 ± 0.418	62.917 ± 0.308
GIN	66.934 ± 0.529	63.777 ± 0.300
GAT	65.490 ± 0.454	59.044 ± 0.183
MLP	55.236 ± 0.317	52.942 ± 0.413

Table 4: **Test performance of DP-GIN in the inductive setting, with privacy budget** $\varepsilon \leq 20$**.**

Model	ogbn-arxiv-disjoint	ogbn-arxiv-clustered	reddit-disjoint
GIN (2-layer)	60.020 ± 0.697	54.239 ± 1.801	93.102 ± 0.211
GIN (1-layer)	59.215 ± 0.332	51.834 ± 1.362	92.682 ± 0.191
DP-GIN (2-layer)	57.475 ± 0.198	47.598 ± 0.661	91.308 ± 0.220
DP-GIN (1-layer)	57.372 ± 0.112	45.017 ± 1.021	90.777 ± 0.084
MLP	55.460 ± 0.188	37.764 ± 1.014	72.272 ± 0.132
DP-MLP	52.822 ± 0.476	34.768 ± 2.524	70.298 ± 0.118

A.2 Proof of Theorem 1

Lemma 2 (Node-Level Sensitivity of any r-Layer GNN)**.** *Let G be any graph such that the maximum in-degree of G is bounded by $K \geq 0$. Let V_{tr} be the training set of nodes. Let $\mathcal{B}_t$ be any choice of m unique subgraphs from $\mathcal{S}_{tr} = \mathsf{SAMPLE} - \mathsf{SUBGRAPHS}(G, V_{tr}, K, r)$. For each node $v \in V_{tr}$, let $\hat{y}_v$ be the prediction from an r-layer GNN when run on the subgraph $S_v \in \mathcal{S}_{tr}$. Now, $\widehat{\mathbf{y}}_v := \mathsf{GNN}(S_v, \mathbf{X}, v; \mathbf{\Theta})$. Consider the loss function $\mathcal{L}$ of the form: $\mathcal{L}(G, \mathbf{\Theta}) = \sum_{v \in V} \ell\left(\widehat{\mathbf{y}}_v; \mathbf{y}_v\right)$. Consider the following quantity $\mathbf{u}_t$ from Algorithm 1:*

$$\mathbf{u}_t(G) = \sum_{v \in \mathcal{B}_t} \text{Clip}_C(\nabla_{\mathbf{\Theta}} \ell\left(\widehat{\mathbf{y}}_v; \mathbf{y}_v\right))$$

Then, the following inequality holds: $\Delta_K(\mathbf{u}_t) < 2C \cdot \frac{K^{r+1}-1}{K-1}$.

Lemma 3 (Un-amplified Privacy Guarantee for Each Iteration of Algorithm 1)**.** *Every iteration t of Algorithm 1 is (α, γ) node-level Rényi DP where $\gamma = \frac{\alpha \cdot (\Delta_K(\mathbf{u}_t))^2}{2\sigma^2}$.*

Proof. Follows directly from (Mironov, 2017, Corollary 3). □

Lemma 4 (Distribution of Loss Terms Per Minibatch)**.** *For any iteration t in Algorithm 1, consider the minibatch $\mathcal{B}_t$ of subgraphs. For any subset $\mathcal{S}$ of d unique subgraphs, define the random variable ρ as $|\mathcal{S} \cap \mathcal{B}_t|$. Then, the distribution of ρ follows the hypergeometric distribution* Hypergeometric(N, d, m)*:*

$$\rho_i = P[\rho = i] = \frac{\binom{d}{i}\binom{N-d}{m-i}}{\binom{N}{m}},$$

where N is the total number of nodes in the training set V_{tr} and $|\mathcal{B}_t| = m$ is the batch size.

Lemma 5 (Adaptation of Lemma 25 from Feldman et al. (2018)). *Let $\mu_0, \ldots, \mu_n$ and $\nu_0, \ldots, \nu_n$ be probability distributions over some domain Z such that: $D_\alpha(\mu_0 \parallel \nu_0) \leq \varepsilon_0$, …, $D_\alpha(\mu_n \parallel \nu_n) \leq \varepsilon_n$, for some given $\varepsilon_0, \ldots, \varepsilon_n$.*

Let ρ be a probability distribution over $[n] = \{0, \ldots, n\}$. Denote by μ_ρ (respectively, ν_ρ) the probability distribution over Z obtained by sampling i from ρ and then outputting a random sample from μ_i (respectively, ν_i). Then:

$$D_\alpha(\mu_\rho \parallel \nu_\rho) \leq \ln \mathbb{E}_{i\sim\rho}\left[e^{\varepsilon_i(\alpha-1)}\right] = \frac{1}{\alpha-1}\ln\sum_{i=0}^{n}\rho_i e^{\varepsilon_i(\alpha-1)}.$$

Lemma 6. *Let ρ, ρ' be sampled from the hypergeometric distribution: $\rho \sim$ Hypergeometric(N, k, m), $\rho' \sim$ Hypergeometric(N, k', m), such that $k \geq k'$. Then, ρ stochastically dominates ρ': $F_{\rho'}(i) \geq F_\rho(i)$ for all $i \in \mathbb{R}$, where F_ρ (respectively, $F_{\rho'}$) is the cumulative distribution function (CDF) of ρ (respectively, ρ').*

Proof of Theorem 1. At a high-level, Lemma 2 tells us that a node can participate in $N(K, r) = \frac{K^{r+1}-1}{K-1}$ training subgraphs from $\mathcal{S}_{tr}$, in the worst case. However, on average, only a fraction of these subgraphs will be sampled in the mini-batch $\mathcal{B}_t$, as Lemma 4 indicates. We use the knowledge of the exact distribution of the number of subgraphs sampled in $\mathcal{B}_t$ provided by Lemma 4 with Lemma 5 to get a tighter bound on the Rényi divergence between the distributions of $\tilde{\mathbf{u}}_t$ over node-level adjacent graphs. Finally, Lemma 6 allows us to make the above bound independent of the actual node being removed, giving our final result.

Let G be any graph with training set V_{tr}. Let G' be formed by removing a single node $\widehat{\mathbf{v}}$ from G, so G and G' are node-level adjacent.

For convenience, for any node v, we denote the corresponding gradient terms $\nabla_{\mathbf{\Theta}}\ell_v$ and $\nabla_{\mathbf{\Theta}}\ell'_v$ as:

$$\nabla_{\mathbf{\Theta}}\ell_v = \nabla_{\mathbf{\Theta}}\ell\left(\mathsf{GNN}(S_v, \mathbf{X}, v; \mathbf{\Theta}); \mathbf{y}_v\right) = \nabla_{\mathbf{\Theta}}\ell\left(\widehat{\mathbf{y}}_v; \mathbf{y}_v\right)$$
$$\nabla_{\mathbf{\Theta}}\ell'_v = \nabla_{\mathbf{\Theta}}\ell\left(\mathsf{GNN}(S'_v, \mathbf{X}', v; \mathbf{\Theta}); \mathbf{y}_v\right) = \nabla_{\mathbf{\Theta}}\ell\left(\widehat{\mathbf{y}}'_v; \mathbf{y}_v\right)$$

Let $\mathcal{S}^r(v)$ be the set of all subgraphs in $\mathsf{SAMPLE}-\mathsf{SUBGRAPHS}(G, V_{tr}, K, r)$ in which v occurs.

$$\mathbf{u}_t(G) - \mathbf{u}_t(G') = \sum_{S_v \in (\mathcal{B}_t \cap \mathcal{S}^r(\widehat{\mathbf{v}}))} \text{Clip}_C(\nabla_{\mathbf{\Theta}}\ell_v) - \text{Clip}_C(\nabla_{\mathbf{\Theta}}\ell'_v).$$

Using the notation from Algorithm 1, we have: $\tilde{\mathbf{u}}_t(G) = \mathbf{u}_t(G) + \mathcal{N}(0, \sigma^2\mathbb{I})$, $\tilde{\mathbf{u}}_t(G') = \mathbf{u}_t(G') + \mathcal{N}(0, \sigma^2\mathbb{I})$. We need to show that $D_\alpha(\tilde{\mathbf{u}}_t(G) \parallel \tilde{\mathbf{u}}_t(G')) \leq \gamma$.

From the above equation, we see that the sensitivity of $\mathbf{u}_t$ depends on the number of subgraphs in $\mathcal{S}^r(v)$ that are present in $\mathcal{B}_t$. Let ρ' be the distribution over $\{0, 1, \ldots |\mathcal{S}^r(v)|\}$ of the number of subgraphs in $\mathcal{S}^r(v)$ present in $\mathcal{B}_t$, that is, $\rho' = |\mathcal{S}^r(v) \cap \mathcal{B}_t|$. Lemma 4 then gives us that the distribution of ρ' is: $\rho' \sim$ Hypergeometric$(N, \mathcal{S}^r(v), m)$. In particular, when $\rho' = i$, exactly i subgraphs are sampled in $\mathcal{B}_t$. Then, following Lemma 2, $\Delta_K(\mathbf{u}_t \mid \rho' = i) < 2iC$. Thus, conditioning on $\rho' = i$, we see that every iteration is (α, γ_i) node-level Rényi DP, by Lemma 3 where $\gamma_i = \alpha \cdot 2i^2C^2/\sigma^2$.

Define the distributions μ_i and ν_i for each $i \in \{0, \ldots, |\mathcal{S}^r(v)|\}$, as follows: $\mu_i = [\tilde{\mathbf{u}}_t(G) \mid \rho' = i], \nu_i = [\tilde{\mathbf{u}}_t(G') \mid \rho' = i]$. Then $D_\alpha(\mu_i \parallel \nu_i) \leq \gamma_i$. For the mixture distributions $\mu_{\rho'} = \tilde{\mathbf{u}}_t(G)$ and $\nu_{\rho'} = \tilde{\mathbf{u}}_t(G')$, Lemma 5 now tells us that:

$$D_\alpha(\tilde{\mathbf{u}}_t(G) \parallel \tilde{\mathbf{u}}_t(G')) = D_\alpha(\mu_{\rho'} \parallel \nu_{\rho'}) \quad \leq \frac{1}{\alpha-1}\ln\mathbb{E}_{i\sim\rho'}\left[\exp\left(\gamma_i(\alpha-1)\right)\right] = \frac{1}{\alpha-1}\ln\mathbb{E}\left[f(\rho')\right].$$

where $f(\rho') = \exp\left(\alpha(\alpha-1)\cdot\frac{2\rho'^2C^2}{\sigma^2}\right)$. Define another distribution ρ as: $\rho \sim$ Hypergeometric$(N, N(K, r), m)$, where $N(K, r) = \frac{K^{r+1}-1}{K-1}$ is the upper bound on $|\mathcal{S}^r(v)|$ from Lemma 1. By Lemma 6, ρ stochastically dominates ρ'. As non-decreasing functions preserve stochastic dominance (Hadar & Russell, 1969), $f(\rho)$ stochastically dominates $f(\rho')$, and hence $\mathbb{E}\left[f(\rho')\right] \leq \mathbb{E}\left[f(\rho)\right]$. It follows that:

$$D_\alpha(\tilde{\mathbf{u}}_t(G) \parallel \tilde{\mathbf{u}}_t(G')) \leq \frac{1}{\alpha-1}\ln\mathbb{E}_\rho\left[\exp\left(\alpha(\alpha-1)\cdot\frac{2\rho^2C^2}{\sigma^2}\right)\right] = \gamma.$$

The theorem now follows from the fact that the above holds for an arbitrary pair of node-level adjacent graphs G and G'. □