ProGAP: Progressive Graph Neural Networks with Differential Privacy Guarantees

Sina Sajadmanesh
sina.sajadmanesh@epfl.ch
Idiap Research Institute and EPFL
Switzerland

Daniel Gatica-Perez
daniel.gatica-perez@epfl.ch
Idiap Research Institute and EPFL
Switzerland

ABSTRACT

Graph Neural Networks (GNNs) have become a popular tool for learning on graphs, but their widespread use raises privacy concerns as graph data can contain personal or sensitive information. Differentially private GNN models have been recently proposed to preserve privacy while still allowing for effective learning over graph-structured datasets. However, achieving an ideal balance between accuracy and privacy in GNNs remains challenging due to the intrinsic structural connectivity of graphs. In this paper, we propose a new differentially private GNN called ProGAP that uses a progressive training scheme to improve such accuracy-privacy trade-offs. Combined with the aggregation perturbation technique to ensure differential privacy, ProGAP splits a GNN into a sequence of overlapping submodels that are trained progressively, expanding from the first submodel to the complete model. Specifically, each submodel is trained over the privately aggregated node embeddings learned and cached by the previous submodels, leading to an increased expressive power compared to previous approaches while limiting the incurred privacy costs. We formally prove that ProGAP ensures edge-level and node-level privacy guarantees for both training and inference stages, and evaluate its performance on benchmark graph datasets. Experimental results demonstrate that ProGAP can achieve up to 5-10% higher accuracy than existing state-of-the-art differentially private GNNs. Our code is available at https://github.com/sisaman/ProGAP.

1 INTRODUCTION

Graph Neural Networks (GNNs) have emerged as a powerful tool for learning from graph-structured data, and their popularity has surged due to their ability to achieve impressive performance in a wide range of applications, including social network analysis, drug discovery, recommendation systems, and traffic prediction [2, 5, 14, 23, 50]. GNNs excel at learning from the structural connectivity of graphs by iteratively updating node embeddings through information aggregation and transformation from neighboring nodes, making them well-suited for tasks such as node classification, graph classification, and link prediction [7, 16, 26, 48, 52, 53]. However, as with many data-driven approaches, GNNs can expose individuals to privacy risks when applied to graph data containing sensitive information, such as social connections, medical records, and financial transactions [38, 44]. Recent studies have shown that various attacks, such as link stealing, membership inference, and node attribute inference, can successfully break the privacy of graph datasets [18, 19, 36, 46], posing a significant challenge for the practical use of GNNs in privacy-sensitive applications.

To address the privacy concerns associated with GNNs, researchers have recently studied *differential privacy (DP)*, a well-established mathematical framework that provides strong privacy guarantees, usually by adding random noise to the data [9, 10]. However, applying DP to GNNs is very challenging due to the complex structural connectivity of graphs, rendering traditional private learning methods, such as differentially private stochastic gradient descent (DP-SGD) [1], infeasible [3, 8, 41]. Recently, the *aggregation perturbation* (AP) approach [41] has emerged as a state-of-the-art technique for ensuring DP in GNNs. Rather than perturbing the model gradients as done in the standard DP-SGD algorithm and its variants, this method perturbs the aggregate information obtained from the GNN neighborhood aggregation step. Consequently, such perturbations can obfuscate the presence of a single edge, which is called *edge-level privacy*, or a single node and all its adjacent edges, referred to as *node-level privacy* [39].

The key limitation of AP is its incompatibility with standard GNN architectures due to the high privacy costs it entails [41]. This is because conventional GNN models constantly query the aggregation functions with every update to the model parameters, which necessitates the re-perturbation of all aggregate outputs at every training iteration to ensure DP, leading to a significant increase in privacy costs. To mitigate this issue, Sajadmanesh *et al.* [41] proposed a method called GAP, which decouples the aggregation steps from the model parameters. In GAP, node features are recursively aggregated first, and then a classifier is learned over the resulting perturbed aggregations, enabling DP to be maintained without incurring excessive privacy costs. Due to having non-trainable aggregations, however, such decoupling approaches reduce the representational power of the GNN [12], leading to suboptimal accuracy-privacy trade-offs.

In the face of these challenges, we present a novel differentially private GNN, called *"**Pro**gressive **G**NN with **A**ggregation **P**erturbation"* (ProGAP). Our new method uses the same AP technique as in GAP to ensure DP. However, instead of decoupling the aggregation steps from the learnable modules, ProGAP adopts a multi-stage, progressive training paradigm to surmount the formidable privacy costs associated with AP. Specifically, ProGAP converts a K-layer GNN model into a sequence of overlapping submodels, where the i-th submodel comprises the first i layers of the model, followed by a lightweight supervision head layer with softmax activation that utilizes node labels to guide the submodel's training. Starting with the shallowest submodel, ProGAP then proceeds progressively to train deeper submodels, each of which is referred to as a training stage. At every stage, the learned node embeddings from the preceding stage are aggregated, perturbed, and then cached to save privacy budget, allowing ProGAP to learn a new set of private node embeddings. Ultimately, the last stage's embeddings are used to generate final node-wise predictions.

The proposed progressive training approach overcomes the high privacy costs of AP by allowing the perturbations to be applied only once per stage rather than at every training iteration. ProGAP also maintains a higher level of representational power compared to GAP, as the aggregation steps now operate on the learned embeddings from the preceding stages, which are more expressive than the raw node features. Moreover, we prove that ProGAP retains all the benefits of GAP, such as edge- and node-level privacy guarantees and zero-cost privacy at inference time. We evaluate ProGAP on five node classification datasets, including Facebook, Amazon, and Reddit, and demonstrate that it can achieve up to 10.4% and 5.5% higher accuracy compared to GAP under edge- and node-level DP with an epsilon of 1 and 8, respectively.

2 RELATED WORK

Several recent studies have investigated differential privacy (DP) to provide formal privacy guarantees in various GNN learning settings. For example, Sajadmanesh and Gatica-Perez [40] propose a locally private GNN for a distributed learning environment, where node features and labels remain private, while the GNN training is federated by a central server with access to graph edges. Lin *et al.* [30] also introduce a locally private GNN, called Solitude, that preserves edge privacy in a decentralized graph, where each node keeps its own private connections. However, both of these approaches use local differential privacy [25], which operates under a different problem setting from our method.

Other approaches propose edge-level DP algorithms for GNNs. Wu *et al.* [46] developed an edge-level private method that modifies the input graph directly through randomized response or the Laplace mechanism, followed by training a GNN on the resulting noisy graph. In contrast, Kolluri *et al.* [28] propose LPGNet, which adopts a tailored neural network architecture. Instead of directly using the graph edges, they encode graph adjacency information in the form of low-sensitivity cluster vectors, which are then perturbed using the Laplace mechanism to preserve edge-level privacy. Unlike our approach, however, neither of these methods provides node-level privacy guarantees.

Olatunji *et al.* [35] propose the first node-level private GNN by adapting the framework of PATE [37]. In their approach, a student GNN model is trained on public graph data, with each node privately labeled using teacher GNN models that are trained exclusively for the corresponding query node. Nevertheless, their approach relies on public graph data and may not be applicable in all situations. Daigavane *et al.* [8] extend the standard DP-SGD algorithm and privacy amplification by subsampling to bounded-degree graph data to achieve node-level DP, but their method fails to provide inference privacy. Finally, Sajadmanesh *et al.* [41] propose GAP, a private GNN learning framework that provides both edge-level and node-level privacy guarantees using the aggregation perturbation approach. They decouple the aggregation steps from the neural network model to manage the privacy costs of their method. Although our method leverages the same aggregation perturbation technique, we take a different approach to limit the privacy costs using a progressive training scheme.

The main concept behind progressive learning is to train the model on simpler tasks first and then gradually move towards more challenging tasks. It was originally introduced to stabilize the training of deep learning models and has been widely adopted in various computer vision applications, such as facial attribute editing [47], image super-resolution [45], image synthesis [24], and representation learning [29]. This technique has also been extended to federated learning, mainly to minimize the communication overhead between clients and the central server [4, 17, 43]. However, the potential benefit of progressive learning in DP applications has not been explored yet. In this paper, we are first to examine the advantages of progressive learning in the context of private GNNs.

3 BACKGROUND

3.1 Differential Privacy

Differential privacy (DP) is a widely accepted framework for measuring the privacy guarantees of algorithms that operate on sensitive data. The main idea of DP is to ensure that the output of an algorithm is not significantly affected by the presence or absence of any particular individual's data in the input. This means that even if an attacker has access to all but one individual's data, they cannot determine whether that individual's data was used in the computation. The formal definition of DP is as follows [10]:

Definition 3.1. Given $\epsilon > 0$ and $\delta \in [0, 1]$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy, if for all adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$ differing by at most one record and for all possible subsets of $\mathcal{A}$'s outputs $S \subseteq \mathit{Range}(\mathcal{A})$:

$$\Pr[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(\mathcal{D}') \in S] + \delta.$$

The value of epsilon, which is often called privacy budget or privacy cost parameter, determines the strength of the privacy guarantee provided by the algorithm, with smaller values of epsilon indicating stronger privacy protection but potentially lower utility of the algorithm. The parameter δ represents the maximum allowable failure probability, i.e., the probability that the algorithm may violate the privacy guarantee, and is usually set to a small value.

The guarantee of differential privacy depends on the notion of adjacency between datasets. In the case of tabular datasets, differential privacy defines adjacency between two datasets as being able to obtain one dataset from the other by removing (or replacing) a single record. However, for graph datasets, adjacency needs to be defined differently due to the presence of links between data records. To adapt the definition of DP for graphs, two different notions of adjacency are defined: edge-level and node-level adjacency. In the former, two graphs are adjacent if they differ only in the presence of a single edge, whereas in the latter, the two graphs differ by a single node with its features, labels, and all attached edges. Accordingly, the definitions of edge-level and node-level DP are derived from these definitions [39]. Specifically, an algorithm $\mathcal{A}$ provides edge-/node-level (ϵ, δ)-DP if for every two edge-/node-level adjacent graph datasets $\mathcal{G}$ and $\mathcal{G}'$ and any set of outputs $S \subseteq \mathit{Range}(\mathcal{A})$, we have $\Pr[\mathcal{A}(\mathcal{G}) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(\mathcal{G}') \in S] + \delta$.

The concepts of edge-level and node-level differential privacy can be intuitively understood as providing privacy protection at different levels of granularity in graph datasets. Edge-level differential privacy is focused on protecting the privacy of edges, which may represent connections between individuals. In contrast, node-level differential privacy aims to protect the privacy of nodes and

their adjacent edges, thus safeguarding all information related to an individual, including their features, labels, and connections.

The difference in granularity between edge-level and node-level DP is crucial because the level of privacy protection needed may depend on the sensitivity of the information being disclosed. For example, protecting only the privacy of individual edges may be sufficient for some applications, while others may require more stringent privacy guarantees that protect the privacy of entire nodes. Our proposed method, which we discuss in detail in Section 4, is capable of providing both edge-level and node-level privacy guarantees.

3.2 Graph Neural Networks

The goal of GNNs is to learn a vector representation, also known as an embedding, for each node in a given graph. These embeddings are learned by taking into account the initial features of the nodes and the structure of the graph (i.e., its edges). The learned embeddings can be applied to various downstream machine learning tasks, such as node classification and link prediction.

A common K-layer GNN is composed of K layers of graph convolution that are applied sequentially. Specifically, layer k takes as input the adjacency matrix $\mathbf{A}$ and the node embeddings produced by layer $k-1$, denoted by $\mathbf{X}^{(k-1)}$, and outputs a new embedding for each node by aggregating the embeddings of its adjacent neighbors, followed by a neural network transformation. In its simplest form, the formal update rule for layer k can be written as follows:

$$\mathbf{X}^{(k)} = \textsc{Upd}\left(\textsc{Agg}(\mathbf{A}, \mathbf{X}^{(k-1)}); \mathbf{\Theta}^{(k)}\right), \tag{1}$$

where Agg is a differentiable permutation-invariant *neighborhood aggregation function*, such as mean, sum, or max pooling, and Upd denotes a learnable transformation, such as a multilayer perceptron (MLP), parameterized by $\mathbf{\Theta}^{(k)}$ that takes the aggregated embeddings as input and produces a new embedding for each node.

3.3 Problem Definition

Consistent with prior work [8, 41], we focus on the node classification task. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed, unweighted graph with a set of nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and edges $\mathcal{E}$ represented by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$. The graph is associated with a set of node features and ground-truth labels. Node features are represented by a matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, where $\mathbf{X}_i$ denotes the d-dimensional feature vector of node v_i. Node labels are denoted by $\mathbf{Y} \in \{0,1\}^{N \times C}$, where C is the number of classes, and $\mathbf{Y}_i$ is a one-hot vector indicating the label of node v_i. We assume that the node labels are known only for a subset of nodes $\mathcal{V}_L \subset \mathcal{V}$. This reflects the transductive (semi-supervised) learning setting, where the goal is to predict the labels of the remaining nodes in $\mathcal{V} \setminus \mathcal{V}_L$.

Consider a GNN-based node classification model $\mathcal{M}(\mathbf{A}, \mathbf{X}; \mathbf{\Theta})$ with parameter set $\mathbf{\Theta}$ that takes the adjacency matrix $\mathbf{A}$ and the node features $\mathbf{X}$, and outputs the corresponding predicted node labels $\widehat{\mathbf{Y}}$:

$$\widehat{\mathbf{Y}} = \mathcal{M}(\mathbf{A}, \mathbf{X}; \mathbf{\Theta}). \tag{2}$$

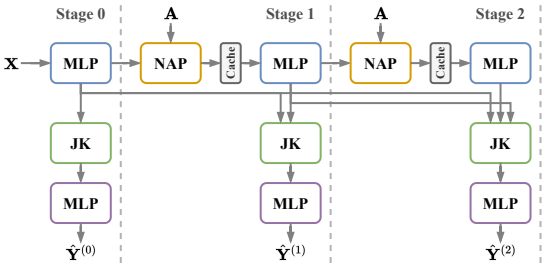

Figure 1: An example ProGAP architecture with three stages (depth = 2). MLP and JK represent multi-layer perceptron and Jumping Knowledge [49] modules, respectively. NAP denotes the normalize-aggregate-perturb module used to ensure the privacy of the adjacency matrix, with its output cached immediately after computation to save privacy budget. Training is done progressively, starting with the first stage and then expanding to the second and third stages, each using its own head MLP. The final prediction is obtained by the head MLP of the last stage.

We seek to minimize a standard classification loss function $\mathcal{L}$ with respect to the set of model parameters $\mathbf{\Theta}$ over the labeled nodes $\mathcal{V}_L$:

$$\begin{aligned}\mathbf{\Theta}^\star &= \arg\min_{\mathbf{\Theta}} \mathcal{L}(\widehat{\mathbf{Y}}, \mathbf{Y}) \\ &= \arg\min_{\mathbf{\Theta}} \left(\sum_{v_i \in \mathcal{V}_L} \ell(\widehat{\mathbf{Y}}_i, \mathbf{Y}_i) \right),\end{aligned} \tag{3}$$

where $\ell : \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}$ is a loss function, such as cross-entropy, and $\mathbf{\Theta}^\star$ denotes the optimal set of parameters.

Our goal is to ensure the privacy of $\mathcal{G}$ at both the training (Eq. 3) and inference (Eq. 2) phases of the model $\mathcal{M}$, using the differential privacy notions defined for graphs, i.e., edge-level and node-level DP. Note that preserving privacy during the inference stage is of utmost importance since the adjacency information of the graph is still used at inference time to generate the predicted labels, and thus sensitive information about the graph could potentially be leaked even with $\mathbf{\Theta}$ being differentially private [41].

4 PROPOSED METHOD

In this section, we present our proposed ProGAP method, which leverages the aggregation perturbation (AP) technique [41] to ensure differential privacy but introduces a novel progressive learning scheme to restrain the privacy costs of AP incurred during training. The overview of ProGAP architecture is illustrated in Figure 1, and its forward propagation (inference) and training algorithms are presented in Algorithm 1 and Algorithm 2, respectively. In the following, we first describe our method in detail and then analyze its privacy guarantees.

Algorithm 1: ProGAP Forward Propagation $\mathcal{M}_s(\mathbf{A}, \mathbf{X}; \sigma, \mathfrak{P}_s)$

Input : Stage s, adjacency matrix $\mathbf{A}$; node features $\mathbf{X}$; noise standard deviation σ; model parameters $\mathfrak{P}_s = \bigcup_{k=0}^{s}\{\Theta_{base}^{(k)}\} \cup \{\Theta_{jump}^{(s)}, \Theta_{head}^{(s)}\}$

Output : Predicated node labels $\widehat{\mathbf{Y}}^{(s)}$

$\widetilde{\mathbf{X}}^{(0)} \leftarrow \mathbf{X}$
for $k \in \{0, \dots, s\}$ **do**
  if $k > 0$ *and* $\widetilde{\mathbf{X}}^{(k)}$ *is not cached* **then**
    $\widetilde{\mathbf{X}}^{(k)} \leftarrow \text{NAP}(\mathbf{A}, \mathbf{X}^{(k-1)}; \sigma)$
    Cache $\widetilde{\mathbf{X}}^{(k)}$
  end
  $\mathbf{X}^{(k)} \leftarrow \text{MLP}_{base}^{(k)}\left(\widetilde{\mathbf{X}}^{(k)}; \Theta_{base}^{(k)}\right)$
end
$\widehat{\mathbf{Y}}^{(s)} \leftarrow \text{MLP}_{head}^{(s)}\left(\text{JK}^{(s)}(\{\mathbf{X}^{(0)}, \dots, \mathbf{X}^{(s)}\}; \Theta_{jump}^{(s)}); \Theta_{head}^{(s)}\right)$
return $\widehat{\mathbf{Y}}^{(s)}$

Algorithm 2: ProGAP Training

Input : Adjacency matrix $\mathbf{A}$; node features $\mathbf{X}$; node labels $\mathbf{Y}$; model depth K; noise standard deviation σ;

Output : Trained model parameters $\mathfrak{P}_K^{\star}$

initialize $\Theta_{base}^{(0)}, \Theta_{jump}^{(0)}, \Theta_{head}^{(0)}$ randomly
$\mathfrak{P}_0 \leftarrow \{\Theta_{base}^{(0)}, \Theta_{jump}^{(0)}, \Theta_{head}^{(0)}\}$
for $s \in \{0, \dots, K\}$ **do**
  $\mathfrak{P}_s^{\star} \leftarrow \arg\min_{\mathfrak{P}} \mathcal{L}\left(\mathcal{M}_s(\mathbf{A}, \mathbf{X}; \sigma, \mathfrak{P}_s), \mathbf{Y}\right)$
  if $s < K$ **then**
    initialize $\Theta_{base}^{(s+1)}, \Theta_{jump}^{(s+1)}, \Theta_{head}^{(s+1)}$ randomly
    $\mathfrak{P}_{s+1} \leftarrow \mathfrak{P}_s^{\star} \cup \{\Theta_{base}^{(s+1)}, \Theta_{jump}^{(s+1)}, \Theta_{head}^{(s+1)}\} \setminus \{\Theta^{\star}{}_{jump}^{(s)}, \Theta^{\star}{}_{head}^{(s)}\}$
  end
end
return $\mathfrak{P}_K^{\star}$

4.1 Model Architecture and Training

We start by considering a simple non-private sequential GNN model $\mathcal{M}$ with K aggregation layers as the following:

$$\mathbf{X}^{(0)} = \text{MLP}_{base}^{(0)}\left(\mathbf{X}; \Theta_{base}^{(0)}\right), \tag{4}$$

$$\mathbf{X}^{(k)} = \text{MLP}_{base}^{(k)}\left(\text{Agg}(\mathbf{A}, \mathbf{X}^{(k-1)}); \Theta_{base}^{(k)}\right) \quad \forall k \in \{1, \dots, K\}, \tag{5}$$

$$\widehat{\mathbf{Y}} = \text{MLP}_{head}\left(\mathbf{X}^{(K)}; \Theta_{head}\right), \tag{6}$$

where $\mathbf{X}^{(k)}$ is the node embeddings generated at layer k by $\text{MLP}_{base}^{(k)}$ having parameters $\Theta_{base}^{(k)}$, and MLP_{head} is a multi-layer perceptron parameterized by Θ_{head} with the softmax activation function that maps the final embeddings $\mathbf{X}^{(K)}$ to the predicted class probabilities $\widehat{\mathbf{Y}}$.

To make this model differentially private, we follow the aggregation perturbation technique proposed by Sajadmanesh*et al.* [41] and add noise to the output of the aggregation function. Specifically, we replace the original aggregation function Agg in Eq. 5 with a *Normalize-Aggregate-Perturb* mechanism defined as:

$$\text{NAP}(\mathbf{A}, \mathbf{X}; \sigma) = \left[\sum_{j=1}^{N} \frac{\mathbf{X}_j}{\|\mathbf{X}_j\|_2}\mathbf{A}_{j,i} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_d) \mid \forall i \in \{1, \dots, N\}\right], \tag{7}$$

where N is the number of nodes, d is the dimension of the input node embeddings, and σ is the standard deviation of the Gaussian noise. Concretely, the NAP mechanism row-normalizes the input embeddings to limit the contribution of each node to the aggregated output, then applies the sum aggregation function followed by adding Gaussian noise to the results.

It can be easily shown that the resulting model provides edge-level DP as every query to the adjacency matrix $\mathbf{A}$ is immediately perturbed with noise. However, training such a model comes at the cost of a significant increase in the privacy budget, which is proportional to the number of queries to the adjacency matrix. Concretely, with T training iterations, the NAP mechanism is queried KT times (at each forward pass and each layer), leading to an excessive accumulated privacy cost of $O(\sqrt{KT})$.

To reduce this cost, we propose a progressive training approach as the following: We first split the model $\mathcal{M}$ into $K+1$ overlapping submodels, where submodel $\mathcal{M}_s$, $s \in \{0, 1, \dots, K\}$, is defined as:

$$\widetilde{\mathbf{X}}^{(s)} = \text{NAP}\left(\mathbf{A}, \mathbf{X}^{(s-1)}; \sigma\right), \tag{8}$$

$$\mathbf{X}^{(s)} = \text{MLP}_{base}^{(s)}\left(\widetilde{\mathbf{X}}^{(s)}; \Theta_{base}^{(s)}\right), \tag{9}$$

$$\widehat{\mathbf{Y}}^{(s)} = \text{MLP}_{head}^{(s)}\left(\text{JK}^{(s)}(\bigcup_{k=0}^{s}\{\mathbf{X}^{(k)}\}; \Theta_{jump}^{(s)}); \Theta_{head}^{(s)}\right), \tag{10}$$

where $\widetilde{\mathbf{X}}^{(s)}$ is the noisy aggregate embeddings of $\mathcal{M}_s$, with $\widetilde{\mathbf{X}}^{(0)} = \mathbf{X}$. $\text{JK}^{(s)}$ is a Jumping Knowledge module [49] with parameters $\Theta_{jump}^{(s)}$ that combines the embeddings generated by submodels $\mathcal{M}_0$ to $\mathcal{M}_s$, and $\text{MLP}_{head}^{(s)}$ is a lightweight, 1-layer head MLP with parameters $\Theta_{head}^{(s)}$ used to train $\mathcal{M}_s$. Finally, $\widehat{\mathbf{Y}}^{(s)}$ is the output predictions of $\mathcal{M}_s$. Then, we progressively train the model in $K+1$ stages, starting from the shallowest submodel $\mathcal{M}_0$ and gradually expanding to the deepest submodel $\mathcal{M}_K$ (which is equivalent to the full model $\mathcal{M}$) as explained by Algorithm 2. For the final inference after training, we simply use the labels predicted by the last submodel $\mathcal{M}_K$, i.e., $\widehat{\mathbf{Y}} = \widehat{\mathbf{Y}}^{(K)}$.

The key point in this training strategy is that we immediately save the outputs of NAP modules on their first query and reuse them throughout the training. More specifically, at each stage s, the perturbed aggregate embedding matrix $\widetilde{\mathbf{X}}^{(s)}$ computed in the first forward pass of $\mathcal{M}_s$ (via Eq. 8) is stored in the cache and reused in all further queries. This caching mechanism allows us to reduce the privacy costs of the model by a factor of T, as the NAP module in this case is only queried K times (once per stage) instead of KT times. At the same time, the aggregations $\widetilde{\mathbf{X}}^{(s)}$ are computed over the embeddings $\mathbf{X}^{(s-1)}$ that are already learned in the preceding stage $s-1$, which provide more expressive power than the raw node features as they also encode information from the adjacency matrix and node labels, and thus lead to better performance.

4.2 Privacy Analysis

With the following theorem, we show that the proposed training strategy provides edge-level DP. The proof is provided in the supplementary material.

Theorem 4.1. *Given the model depth $K \geq 0$ and noise variance σ^2, for any $\delta \in (0,1)$ Algorithm 2 satisfies edge-level (ϵ, δ)-DP with $\epsilon = \frac{K}{2\sigma^2} + \sqrt{2K\log(1/\delta)}/\sigma$.*

To ensure node-level DP, however, we must train every submodel using DP-SGD or its variants, as in this case node features and labels are also private and can be leaked with non-private training. Theorem 4.2 establishes the node-level DP guarantee of ProGAP's training algorithm when combined with DP-SGD:

Theorem 4.2. *Given the number of nodes N, batch-size $B < N$, number of per-stage training iterations T, gradient clipping threshold $C > 0$, model depth $K \geq 0$, maximum cut-off degree $D \geq 1$, noise variance for aggregation perturbation $\sigma^2_{AP} > 0$, and noise variance for gradient perturbation $\sigma^2_{GP} > 0$, Algorithm 2 satisfies node-level (ϵ, δ)-DP for any $\delta \in (0,1)$ with:*

$$\epsilon \leq \min_{\alpha>1} \frac{(K+1)T}{\alpha-1} \log\Bigg\{\left(1-\frac{B}{N}\right)^{\alpha-1}\left(\alpha\frac{B}{N}-\frac{B}{N}+1\right) + \binom{\alpha}{2}\left(\frac{B}{N}\right)^2\left(1-\frac{B}{N}\right)^{\alpha-2} e^{\frac{C^2}{\sigma^2_{GP}}} + \sum_{l=3}^{\alpha}\binom{\alpha}{l}\left(1-\frac{B}{N}\right)^{\alpha-l}\left(\frac{B}{N}\right)^{l} e^{(l-1)\left(\frac{C^2 l}{2\sigma^2_{GP}}\right)}\Bigg\} + \frac{DK\alpha}{2\sigma^2_{AP}} + \frac{\log(1/\delta)}{\alpha-1}, \quad (11)$$

providing that the optimization in line 4 of Algorithm 2 is done using DP-SGD.

The proof is available in the supplementary material. Note that to decrease the node-level sensitivity of the NAP mechanism (i.e., the impact of adding/removing a node on the output of the NAP mechanism), we assume an upper bound D on node degrees, and randomly sample edges from the graph to ensure that each node has no more than D outgoing edges. This is a standard technique to ensure bounded-degree graphs [8, 41].

In addition to training privacy, ProGAP also guarantees privacy during inference at both edge and node levels without any further privacy costs. This is because the entire noisy aggregate matrices $\widetilde{\mathbf{X}}^{(k)}$ corresponding to all the nodes –both training and test ones– are already computed and cached during training and reused for inference (i.e., lines 4 and 5 of Algorithm 1 is not executed at inference time). Therefore, the inference for a node i no longer depends on the adjacency matrix and neighboring node features, but only on its own aggregated features $\widetilde{\mathbf{X}}^{(k)}_i$ for all $k \in \{0,\dots,K\}$, which are already computed with DP during training. As a result, the inference only post-processes differentially private outputs, which does not incur any additional privacy costs.

Table 1: Dataset Statistics

Dataset	# Nodes	# Edges	# Features	# Classes	Degree
Facebook	26,406	2,117,924	501	6	62
Reddit	116,713	46,233,380	602	8	209
Amazon	1,790,731	80,966,832	100	10	22
FB-100	1,120,280	86,304,478	537	6	57
WeNet	37,576	22,684,206	44	4	286

5 EXPERIMENTAL SETUP

We test our proposed method on node-wise classification tasks and evaluate its effectiveness in terms of classification accuracy and privacy guarantees.

5.1 Datasets

We conduct experiments on three real-world datasets that have been used in previous work [8, 35, 41], namely Facebook [42], Reddit [16], and Amazon [6], and also two new datasets: FB-100 [42] and WeNet [13, 31]. The Facebook dataset is a collection of anonymized social network data from UIUC students, where nodes represent users, edges indicate friendships, and the task is to predict students' class year. The Reddit dataset comprises a set of Reddit posts as nodes, where edges represent if the same user commented on both posts, and the goal is to predict the posts' subreddit. The Amazon dataset is a product co-purchasing network, with nodes representing products and edges indicating if two products are purchased together, and the objective is to predict product category. FB-100 is an extended version of the Facebook dataset combining the social network of 100 different American universities. WeNet is a mobile sensing dataset collected from university students in four different countries. Nodes represent eating events, which are linked based on the similarity of location and Wi-Fi sensor readings. Node features are extracted based on cellular and application sensors, and the goal is to predict the country of the events.A summary of the datasets is provided in Table 1.

5.2 Baselines

We compare ProGAP against the following baselines:

GraphSAGE [16]. This is one of the most popular GNN models, which we use for non-private performance comparison with our method. Moreover, it serves as the backbone model for the following EdgeRand and DP-GNN baselines. The number of message-passing layers is tuned in the range of 1 to 5 for each dataset. We also use one preprocessing and one postprocessing layer, and equip the model with jumping knowledge modules [49] to get better performance.
MLP and DP-MLP. We use a simple 3-layer MLP model which does not use any graph structural information and therefore is perfectly edge-level private. DP-MLP is the node-level private variant of MLP, which is trained using the DP version of the Adam optimizer (DP-Adam).
EdgeRand [46]. This edge-level private method directly adds noise to the adjacency matrix. We use the enhanced version of

Table 2: Comparison of Experimental Results (Mean Accuracy ± 95% CI)

Privacy Level	Method	ϵ	Facebook	Reddit	Amazon	FB-100	WeNet
Non-Private	GraphSAGE [16]	∞	**84.7 ± 0.09**	99.4 ± 0.01	93.2 ± 0.07	74.0 ± 0.80	71.6 ± 0.54
	GAP [41]	∞	80.5 ± 0.42	**99.5 ± 0.01**	92.0 ± 0.10	66.4 ± 0.35	69.7 ± 0.14
	ProGAP (**Ours**)	∞	84.5 ± 0.24	99.3 ± 0.03	**93.3 ± 0.04**	**74.4 ± 0.14**	**73.9 ± 0.25**
Edge-Level Private	MLP	0.0	50.8 ± 0.20	82.5 ± 0.08	71.1 ± 0.18	34.9 ± 0.02	51.5 ± 0.22
	EdgeRand [46]	1.0	50.2 ± 0.50	82.8 ± 0.05	72.7 ± 0.1	34.9 ± 0.05	52.1 ± 0.48
	GAP [41]	1.0	69.4 ± 0.39	97.5 ± 0.06	78.8 ± 0.26	46.5 ± 0.58	62.4 ± 0.28
	ProGAP (**Ours**)	1.0	**77.2 ± 0.33**	**97.8 ± 0.05**	**84.2 ± 0.07**	**56.9 ± 0.30**	**68.8 ± 0.23**
Node-Level Private	DP-MLP [1]	8.0	50.2 ± 0.25	81.5 ± 0.12	73.6 ± 0.05	34.5 ± 0.13	50.8 ± 0.37
	DP-GNN [8]	8.0	62.6 ± 0.86	**95.6 ± 0.31**	78.5 ± 0.86	46.5 ± 0.57	54.2 ± 0.73
	GAP [41]	8.0	63.9 ± 0.49	93.9 ± 0.09	77.6 ± 0.07	43.0 ± 0.20	58.2 ± 0.39
	ProGAP (**Ours**)	8.0	**69.3 ± 0.33**	94.0 ± 0.04	**79.1 ± 0.10**	**48.5 ± 0.36**	**61.0 ± 0.34**

EdgeRand from [41] that uses the Asymmetric Randomized Response [20] with the GraphSAGE model as the backbone GNN.

DP-GNN [8]. This node-level private approach extends the DP-SGD algorithm to GNNs. Note, however, that this method does not support inference privacy, but we nevertheless include it in our comparison. Similar to EdgeRand, we use the GraphSAGEmodel as the backbone GNN for this method as well.

GAP [41]. GAP is the state-of-the-art approach that supports both edge-level and node-level DP. We use GAP's official implementation on GitHub[1] and follow the same experimental setup as reported in the original paper.

We do not include other available differentially private GNN baselines (e.g., [3, 33, 36, 40]) as they have different problem settings that make them not directly comparable to our method.

5.3 Implementation Details

We follow the same experimental setup as GAP [41], and randomly split the nodes in all the datasets into training, validation, and test sets with 75/10/15% ratio, respectively. We vary ϵ within $\{0.25, 0.5, 1, 2, 4, \infty\}$ for the edge-level privacy ($\epsilon = \infty$ corresponds to the non-private setting) and within $\{2, 4, 8, 16, 32\}$ for the node-level privacy setting. For each ϵ value, we tune the following hyperparameters based on the mean validation set accuracy computed over 10 runs: MLP_{base} layers in $\{1, 2\}$, model depth K in $\{1, 2, 3, 4, 5\}$, and learning rate in $\{0.01, 0.05\}$. The value of δ is fixed per each dataset to be smaller than the inverse number of private units (i.e., edges for edge-level privacy, nodes for node-level privacy). For all cases, we set the number of MLP_{head} layers to 1 and use concatenation for the JK modules. Additionally, we set the number of hidden units to 16 and use the SeLU activation function [27]. We use batch normalization except for the node-level setting, for which we use group normalization with one group. Under the edge-level setting, we train the models with full-sized batches for 100 epochs using the Adam optimizer and perform early stopping based on the validation set accuracy. For the node-level setting, we use randomized neighbor sampling to bound the maximum degree D to 50 for Amazon, 100 for Facebook and FB-100, and 400 for Reddit and WeNet. We use DP-Adam [15] with a clipping threshold of 1.0. We tune the number of per-stage epochs in $\{5, 10\}$ and set the batch size to 256, 1024, 2048, 4096, and 4096 for Facebook, WeNet, Reddit, Amazon, and FB-100, respectively. Finally, we report the average test accuracy over 10 runs with 95% confidence intervals calculated by bootstrapping with 1000 samples. We open-source our implementation on GitHub.[2]

5.4 Software and Hardware

We use PyTorch Geometric [11] for implementing the models, `autodp`[3] for privacy accounting, and Opacus [51] for DP training. We run all the experiments on an HPC cluster with NVIDIA Tesla V100 and GeForce RTX 3090 GPUs with maximum 32GB memory.

6 RESULTS AND DISCUSSION

6.1 Accuracy-Privacy Trade-off

Table 2 presents the test accuracy of ProGAP against other baselines at three different privacy levels: non-private with $\epsilon = \infty$, edge-level privacy with $\epsilon = 1$, and node-level privacy with $\epsilon = 8$. The results are reported as mean accuracy ± 95% confidence interval. We observe that ProGAP outperforms other approaches most of the time, and often by a substantial margin. In the non-private setting, ProGAP performs comparably with GraphSAGE, but achieves higher test accuracies than GAP on all datasets except Reddit, where GAP performs only slightly better. This shows that our ProGAP method is also a strong predictor in the non-private learning setting.

Moving to edge-level privacy, ProGAP consistently outperforms other approaches across all datasets, with the largest performance gap of 10.4% accuracy points compared to GAP observed on FB-100. Under node-level DP, ProGAP is still superior to the other baselines across all datasets except Reddit, where DP-GNN achieves a slightly higher accuracy. Note, however, that DP-GNN only guarantees node-level privacy for the model parameters and fails to provide

[1] https://github.com/sisaman/GAP

[2] https://github.com/sisaman/ProGAP

[3] https://github.com/yuxiangw/autodp

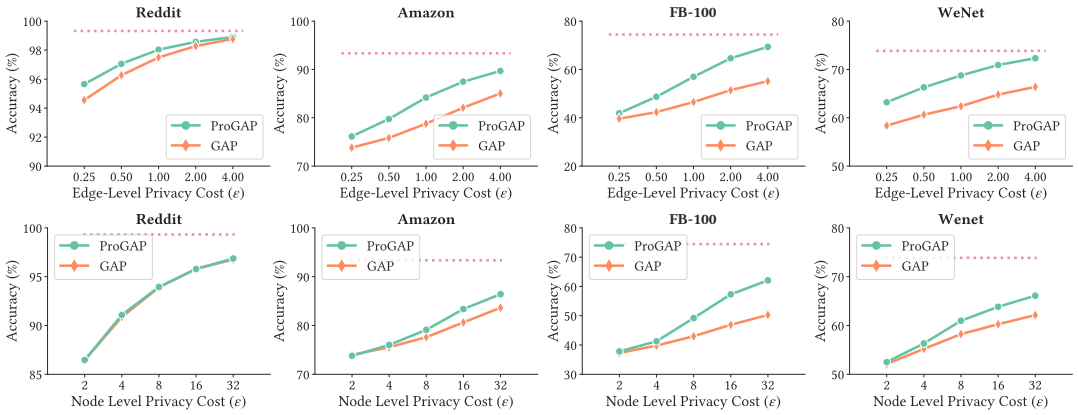

Figure 2: Accuracy-privacy trade-off of edge-level (top) and node-level (bottom) private methods. The dotted line represents the accuracy of the non-private ProGAP.

inference privacy, while our ProGAP method provides both training and inference privacy guarantees. Compared to GAP, the largest margin in this category is also observed on FB-100, where ProGAP achieves 5.5% more accuracy points.

To examine the performance of ProGAP at different privacy budgets, we varied ϵ between 0.25 to 4 for edge-level privacy and 2 to 32 for node-level private algorithms. We then recorded the accuracy of ProGAP for each privacy budget and compare it with GAP as the most similar baseline. The outcome for both edge-level and node-level privacy settings is depicted in Figure 2. Notably, we observe that ProGAP achieves higher accuracies than GAP across all ϵ values tested and approaches the non-private accuracy more quickly under both privacy settings. This is because in ProGAP, each aggregation step is computed on the node embeddings learned in the previous stage, providing greater representational power than GAP, which just recursively computes the aggregations on the initial node representations.

It is worth noting that the performance discrepancy between ProGAP and GAP is not consistent across all datasets. For instance, this gap in accuracy is less pronounced with the Reddit dataset compared to FB-100. This is due to the specific characteristics and the learning task of each dataset, which require different levels of graph representational power. In Reddit, where the goal is to predict the community of nodes (representing Reddit posts), most of the pertinent information needed is already present in the node features, making the relationships between the posts less crucial for this prediction task. In contrast, the learning task of the FB-100 dataset (predicting students' class year) relies more heavily on the graph structure, necessitating more powerful graph representations. Therefore, the performance difference between ProGAP and GAP is more noticeable in this dataset. This connection between the learning task and the graph structure will be revisited in Section 6.3.

6.2 Convergence Analysis

We examine the convergence of ProGAP to further understand its behavior under the two privacy settings. We report the training and validation accuracy of ProGAP per training step under edge-level privacy with $\epsilon = 1$ and node-level privacy with $\epsilon = 8$. For all datasets, ProGAP is trained for 100 and 10 epochs per stage under edge and node-level privacy, respectively. We fix $K = 5$ in all settings. The results are shown in Figure 3. We observe that both training and validation accuracies increase as ProGAP moves from stage 0 to 5, with diminishing returns for more stages, which indicates the higher importance of the nearby neighbors to each node, since the receptive field of nodes grows with the number of stages. Moreover, we observe negligible discrepancies between training and validation accuracy when the model converges, which suggests higher resilience to privacy attacks, such as membership inference, which typically rely on large generalization gaps. This result is in line with previous work showing the effectiveness of DP against privacy attacks [21, 22, 34, 41].

6.3 Effect of the Model Depth

We explore how the performance of ProGAP is influenced by modifying the model depth K, or equivalently, the number of stages $K + 1$. We experiment with different values of K ranging from 1 to 5 and evaluate ProGAP's accuracy under varying privacy budgets of $\epsilon \in \{0.25, 1, 4\}$ for edge-level DP and $\epsilon \in \{2, 8, 32\}$ for node-level privacy. The results are demonstrated in Figure 4. We observe that ProGAP can generally gain advantages from increasing the depth, but there is a compromise depending on the privacy budget: deeper models lead to better accuracy under higher privacy budgets, while lower privacy budgets require shallower models to achieve optimal performance. This is because ProGAP can leverage data from more remote nodes with a higher value of K, which can boost the final accuracy, but it also increases the amount of noise in the aggregations, which has a detrimental effect on the model's accuracy. When the privacy budget is lower and the amount of noise is greater, ProGAP has the best performance at smaller values of K. But as the privacy budget grows, the magnitude of the noise is lowered, enabling the model to take advantage of greater K values.

An intriguing aspect to note is the potential improvement in ProGAP's accuracy across various datasets as its depth increases. Observing the Reddit and FB-100 datasets as an example, it is clear that the boost in performance from increasing the parameter K is considerable for FB-100 under moderate privacy budgets, while

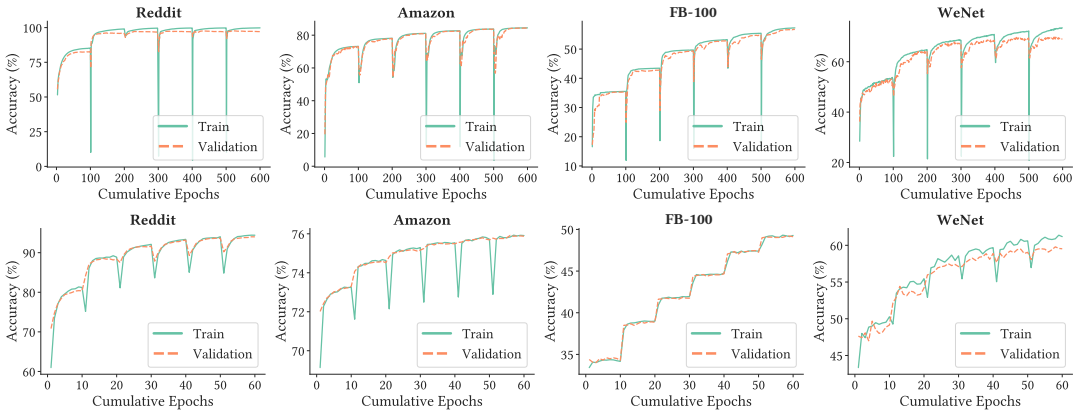

Figure 3: Convergence of ProGAP with $K = 5$ under edge-level (top) and node-level (bottom) privacy, with $\epsilon = 1$ and $\epsilon = 8$, respectively.

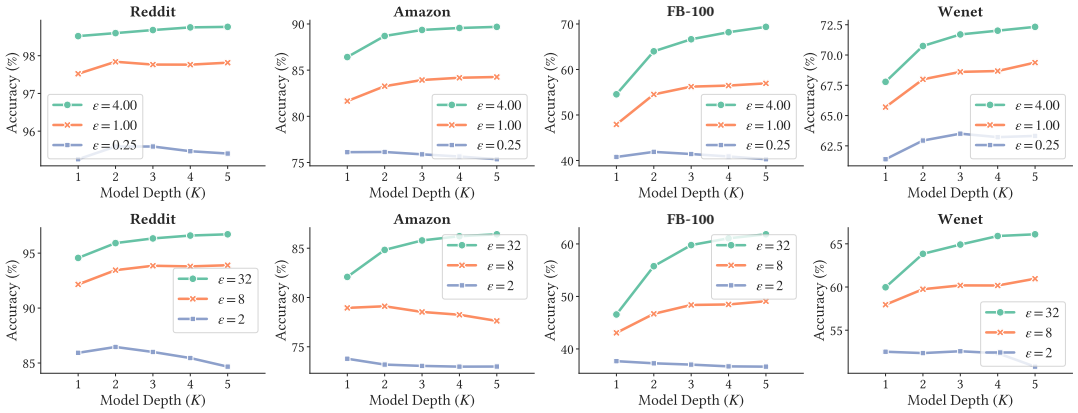

Figure 4: Effect of the model depth on ProGAP's accuracy under edge-level (top) and node-level (bottom) privacy.

for Reddit, the gain in accuracy is minimal. This discrepancy can again be explained by the nature and the specific learning tasks of the datasets. In Reddit, where the task is less reliant on the graph structure and the node features hold more predictive value, the depth of ProGAP doesn't significantly influence its performance. On the contrary, in FB-100, where the learning task heavily depends on the graph structure, ProGAP's performance is more sensitive to the model's depth. These observations align with the previous discussion about the wider performance gap between ProGAP and GAP in FB-100 compared to Reddit.

7 CONCLUSION

In this paper, we introduced ProGAP, a novel differentially private GNN that improves the challenging accuracy-privacy trade-off in learning from graph data. Our approach uses a progressive training scheme that splits the GNN into a sequence of overlapping submodels, each of which is trained over privately aggregated node embeddings learned and cached by the previous submodels. By combining this technique with the aggregation perturbation method, we formally proved that ProGAP can ensure edge-level and node-level privacy guarantees for both training and inference stages. Empirical evaluations on benchmark graph datasets demonstrated that ProGAP can achieve state-of-the-art accuracy by outperforming existing methods. Future work could include exploring new architectures or training strategies to further improve the accuracy-privacy trade-off of differentially private GNNs, especially in the more challenging node-level privacy setting.

ACKNOWLEDGMENTS

This work was supported by the European Commission's H2020 Program ICT-48-2020, AI4Media Project, under grant number 951911. It was also supported by the European Commission's H2020 WeNet Project, under grant number 823783.

REFERENCES

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

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

[3] Morgane Ayle, Jan Schuchardt, Lukas Gosch, Daniel Zügner, and Stephan Günnemann. 2023. Training Differentially Private Graph Neural Networks with Random Walk Sampling. *arXiv preprint arXiv:2301.00738* (2023).

[4] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. 2020. Decoupled Greedy Learning of CNNs. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 736–745.

[5] Mark Cheung and José M. F. Moura. 2020. Graph Neural Networks for COVID-19 Drug Discovery. In *2020 IEEE International Conference on Big Data (Big Data)*. 5646–5648. https://doi.org/10.1109/BigData50022.2020.9378164

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.

[7] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal Neighbourhood Aggregation for Graph Nets. In *Advances in Neural Information Processing Systems*.

[8] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. 2022. Node-Level Differentially Private Graph Neural Networks. In *ICLR 2022 Workshop on PAIR^2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*. https://openreview.net/forum?id=BCfgOLx3gb9

[9] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*. Springer, 1–19.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[11] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[12] Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. 2021. GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*. PMLR, 3294–3304.

[13] Fausto Giunchiglia, Ivano Bison, Matteo Busso, Ronald Chenu-Abente, Marcelo Rodas, Mattia Zeni, Can Gunel, Giuseppe Veltri, Amalia De Götzen, Peter Kun, et al. 2021. A worldwide diversity pilot on daily routines and social practices (2020). (2021).

[14] Nikolaos Gkalelis, Andreas Goulas, Damianos Galanopoulos, and Vasileios Mezaris. 2021. ObjectGraphs: Using Objects and a Graph Convolutional Network for the Bottom-Up Recognition and Explanation of Events in Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3375–3383.

[15] Roan Gylberth, Risman Adnan, Setiadi Yazid, and T Basaruddin. 2017. Differentially private optimization algorithms for deep neural networks. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE, 387–394.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[17] Chaoyang He, Shen Li, Mahdi Soltanolkotabi, and Salman Avestimehr. 2021. Pipetransformer: Automated elastic pipelining for distributed training of transformers. *arXiv preprint arXiv:2102.03161* (2021).

[18] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing Links from Graph Neural Networks. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[19] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. 2021. Node-Level Membership Inference Attacks Against Graph Neural Networks. *arXiv preprint arXiv:2102.05429* (2021).

[20] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2022. Communication-Efficient Triangle Counting under Local Differential Privacy. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/usenixsecurity22/presentation/imola

[21] Matthew Jagielski, Jonathan R. Ullman, and Alina Oprea. 2020. Auditing Differentially Private Machine Learning: How Private is Private SGD?. In *Proceedings of the Advances in Neural Information Processing (NeurIPS)*. Virtual Event.

[22] Bargav Jayaraman and David Evans. 2019. Evaluating Differentially Private Machine Learning in Practice. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 1895–1912. https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman

[23] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* (2022), 117921.

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[25] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2011. What can we learn privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[26] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[27] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*. 972–981.

[28] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. 2022. LPGNet: Link Private Graph Networks for Node Classification. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1813–1827. https://doi.org/10.1145/3548606.3560705

[29] Zhiyuan Li, Jaideep Vitthal Murkute, Prashnna Kumar Gyawali, and Linwei Wang. 2020. Progressive learning and disentanglement of hierarchical representations. *arXiv preprint arXiv:2002.10549* (2020).

[30] Wanyu Lin, Baochun Li, and Cong Wang. 2022. Towards Private Learning on Decentralized Graphs With Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 17 (2022), 2936–2946. https://doi.org/10.1109/TIFS.2022.3198283

[31] Lakmal Meegahapola, William Droz, Peter Kun, Amalia de Götzen, Chaitanya Nutakki, Shyam Diwakar, Salvador Ruiz Correa, Donglei Song, Hao Xu, Miriam Bidoglia, et al. 2023. Generalization and Personalization of Mobile Sensing-Based Mood Inference Models: An Analysis of College Students in Eight Countries. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–32.

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

[33] Tamara T Mueller, Johannes C Paetzold, Chinmay Prabhakar, Dmitrii Usynin, Daniel Rueckert, and Georgios Kaissis. 2022. Differentially Private Graph Neural Networks for Whole-Graph Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[34] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. 2021. Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. San Francisco, CA, USA.

[35] Iyiola E Olatunji, Thorben Funke, and Megha Khosla. 2021. Releasing Graph Neural Networks with Differential Privacy Guarantees. *arXiv preprint arXiv:2109.08907* (2021).

[36] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 11–20.

[37] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755* (2016).

[38] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2110–2119.

[39] Sofya Raskhodnikova and Adam Smith. 2016. Differentially Private Analysis of Graphs. *Encyclopedia of Algorithms* (2016).

[40] Sina Sajadmanesh and Daniel Gatica-Perez. 2021. Locally private graph neural networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2130–2145.

[41] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. 2023. GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA.

[42] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.

[43] Hui-Po Wang, Sebastian Stich, Yang He, and Mario Fritz. 2022. ProgFed: Effective, Communication, and Computation Efficient Federated Learning by Progressive Training. In *International Conference on Machine Learning*. PMLR, 23034–23054.

[44] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. 2021. A Review on Graph Neural Network Methods in Financial Applications. *arXiv preprint arXiv:2111.15367* (2021).

[45] Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. 2018. A fully progressive approach to single-image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 864–873.

[46] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. 2022. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2005–2024.

[47] Rongliang Wu, Gongjie Zhang, Shijian Lu, and Tao Chen. 2020. Cascade ef-gan: Progressive facial expression editing with local focuses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5021–5030.

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[49] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 5453–5462.

[50] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[51] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. *arXiv preprint arXiv:2109.12298* (2021).

[52] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* 31 (2018), 5165–5175.

[53] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[54] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission Subsampled Rényi Differential Privacy. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 7634–7642. https://proceedings.mlr.press/v97/zhu19c.html

ProGAP: Progressive Graph Neural Networks with Differential Privacy Guarantees

Supplementary Material

Sina Sajadmanesh
sina.sajadmanesh@epfl.ch
Idiap Research Institute and EPFL
Switzerland

Daniel Gatica-Perez
daniel.gatica-perez@epfl.ch
Idiap Research Institute and EPFL
Switzerland

A RÉNYI DIFFERENTIAL PRIVACY

The proofs presented in the following section are based on *Rényi Differential Privacy* (RDP) [32], which is an alternative definition of DP that gives tighter sequential composition results. The formal definition of RDP is as follows:

Definition A.1 (Rényi Differential Privacy [32]). Given $\alpha > 1$ and $\epsilon > 0$, a randomized algorithm $\mathcal{A}$ satisfies (α, ϵ)-RDP if for every adjacent datasets X and X', we have:

$$D_\alpha \left(\mathcal{A}(\mathcal{D}) \| \mathcal{A}(\mathcal{D}') \right) \leq \epsilon, \tag{12}$$

where $D_\alpha(P\|Q)$ is the Rényi divergence of order α between probability distributions P and Q defined as:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{x \sim Q} \left[\frac{P(x)}{Q(x)} \right]^\alpha .$$

Proposition 3 of [32] measures the privacy guarantee of the composition of RDP algorithms as the following:

Proposition A.2 (Composition of RDP mechanisms [32]). *Let f be (α, ϵ_1)-RDP and g be (α, ϵ_2)-RDP, then the mechanism defined as (X, Y), where $X \sim f(\mathcal{D})$ and $Y \sim g(X, \mathcal{D})$, satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

A key property of RDP is that it can be converted to standard (ϵ, δ)-DP using the Proposition 3 of [32], as follows:

Proposition A.3 (From RDP to (ϵ, δ)-DP [32]). *If $\mathcal{A}$ is an (α, ϵ)-RDP algorithm, then it also satisfies $(\epsilon + \log(1/\delta)/\alpha - 1, \delta)$-DP for any $\delta \in (0, 1)$.*

B DEFERRED THEORETICAL ARGUMENTS

B.1 Proof of Theorem 4.1

Proof. In Algorithm 2, the graph's adjacency is only used when the NAP mechanism is invoked during the forward propagation of submodels $\mathcal{M}_1$ to $\mathcal{M}_K$. According to Lemma 1 of [41], the edge-level sensitivity of the NAP mechanism is 1, and thus based on Corollary 3 of [32], each individual query to the NAP mechanism is $(\alpha, \alpha/2\sigma^2)$-RDP. Due to ProGAP's caching system, the NAP mechanism is only invoked K times during training (once for each submodel), and the rest of the training process does not query the graph edges. As a result, Algorithm 2 can be seen as an adaptive composition of K NAP mechanisms, which based on Proposition A.2, is $(\alpha, K\alpha/2\sigma^2)$-RDP. According to Proposition A.3, this is equivalent to edge-level (ϵ, δ)-DP with $\epsilon = \frac{K\alpha}{2\sigma^2} + \frac{\log(1/\delta)}{\alpha - 1}$. Minimizing this expression over $\alpha > 1$ gives $\epsilon = \frac{K}{2\sigma^2} + \sqrt{2K \log(1/\delta)}/\sigma$. □

B.2 Proof of Theorem 4.2

Proof. Algorithm 2 is composed of $K + 1$ stages, where each stage $s \in \{1, \ldots, K\}$ starts by computing and perturbing the aggregate embeddings (Eq. 8), which is the only part where the graph adjacency information is involved. As this part is privatized by the NAP mechanism, the rest of the process in stage $s \geq 1$ is just normal graph-agnostic training over tabular-like data, which is made private using DP-SGD. The exception is stage 0, which does not use the graph's adjacency at all, and thus it is just privatized using DP-SGD. Therefore, Algorithm 2 can be seen as an adaptive composition of K NAP mechanisms and $K + 1$ DP-SGD algorithms. According to Lemma 3 of [41], the NAP mechanism is node-level $(\alpha, D\alpha/2\sigma_{AP}^2)$-RDP. The DP-SGD algorithm itself is a composition of T subsampled Gaussian mechanisms, which according to Theorem 11 of [54] and Proposition A.2 is $(\alpha, \epsilon_{\text{DPSGD}})$-RDP, where:

$$\epsilon_{\text{DPSGD}} \leq \frac{T}{\alpha - 1} \log \left\{ \left(1 - \frac{B}{N}\right)^{\alpha - 1} \left(\alpha \frac{B}{N} - \frac{B}{N} + 1\right) + \binom{\alpha}{2} \left(\frac{B}{N}\right)^2 \left(1 - \frac{B}{N}\right)^{\alpha - 2} e^{\frac{C^2}{\sigma_{GP}^2}} + \sum_{l=3}^{\alpha} \binom{\alpha}{l} \left(1 - \frac{B}{N}\right)^{\alpha - l} \left(\frac{B}{N}\right)^l e^{(l-1)\left(\frac{C^2 l}{2\sigma_{GP}^2}\right)} \right\}.$$

Overall, according to Proposition A.2, the composition of K NAP mechanisms and $K+1$ DP-SGD algorithms is $(\alpha, \epsilon_{\text{total}})$-RDP, where:

$$\epsilon_{\text{total}} \leq \frac{(K+1)T}{\alpha - 1} \log \left\{ \left(1 - \frac{B}{N}\right)^{\alpha - 1} \left(\alpha \frac{B}{N} - \frac{B}{N} + 1\right) + \binom{\alpha}{2} \left(\frac{B}{N}\right)^2 \left(1 - \frac{B}{N}\right)^{\alpha - 2} e^{\frac{C^2}{\sigma_{GP}^2}} + \sum_{l=3}^{\alpha} \binom{\alpha}{l} \left(1 - \frac{B}{N}\right)^{\alpha - l} \left(\frac{B}{N}\right)^l e^{(l-1)\left(\frac{C^2 l}{2\sigma_{GP}^2}\right)} \right\} + \frac{DK\alpha}{2\sigma_{AP}^2}.$$

The proof is completed by applying Proposition A.3 to the above expression and minimizing the upper bound over $\alpha > 1$:

$$\epsilon \leq \min_{\alpha>1} \frac{(K+1)T}{\alpha-1} \log \Bigg\{ \left(1-\frac{B}{N}\right)^{\alpha-1} \left(\alpha\frac{B}{N} - \frac{B}{N} + 1\right) + \binom{\alpha}{2} \left(\frac{B}{N}\right)^2 \left(1-\frac{B}{N}\right)^{\alpha-2} e^{\frac{C^2}{\sigma^2_{GP}}} + \sum_{l=3}^{\alpha} \binom{\alpha}{l} \left(1-\frac{B}{N}\right)^{\alpha-l} \left(\frac{B}{N}\right)^{l} e^{(l-1)\left(\frac{C^2 l}{2\sigma^2_{GP}}\right)} \Bigg\} + \frac{DK\alpha}{2\sigma^2_{AP}} + \frac{\log(1/\delta)}{\alpha-1},$$

□