GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation

Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, Daniel Gatica-Perez

GAP: Differentially Private Graph Neural Networks with Aggregation Perturbation

Sina Sajadmanesh[1,2] Ali Shahin Shamsabadi[3] Aurélien Bellet[4] Daniel Gatica-Perez[1,2]
[1]*Idiap Research Institute* [2]*EPFL* [3]*The Alan Turing Institute* [4]*Inria*

Abstract

In this paper, we study the problem of learning Graph Neural Networks (GNNs) with Differential Privacy (DP). We propose a novel differentially private **G**NN based on **A**ggregation **P**erturbation (GAP), which adds stochastic noise to the GNN's aggregation function to statistically obfuscate the presence of a single edge (edge-level privacy) or a single node and all its adjacent edges (node-level privacy). Tailored to the specifics of private learning, GAP's new architecture is composed of three separate modules: (i) the encoder module, where we learn private node embeddings without relying on the edge information; (ii) the aggregation module, where we compute noisy aggregated node embeddings based on the graph structure; and (iii) the classification module, where we train a neural network on the private aggregations for node classification without further querying the graph edges. GAP's major advantage over previous approaches is that it can benefit from multi-hop neighborhood aggregations, and guarantees both edge-level and node-level DP not only for training, but also at inference with no additional costs beyond the training's privacy budget. We analyze GAP's formal privacy guarantees using Rényi DP and conduct empirical experiments over three real-world graph datasets. We demonstrate that GAP offers significantly better accuracy-privacy trade-offs than state-of-the-art DP-GNN approaches and naive MLP-based baselines. Our code is publicly available at https://github.com/sisaman/GAP.

1 Introduction

Real-world datasets are often represented by graphs, such as social [36], financial [42], transportation [8], or biological [25] networks, modeling the relations (i.e., edges) between a collection of entities (i.e., nodes). Graph Neural Networks (GNNs) have achieved state-of-the-art performance in learning over such relational data in various graph-based machine learning tasks, such as node classification, link prediction, and graph classification [26, 47, 52]. Due to their superior performance, GNNs are now widely used in many applications, such as recommendation systems, credit issuing, traffic forecasting, drug discovery, and medical diagnosis [4, 14, 24, 30, 49].

Privacy concerns. Despite their success, real-world deployments of GNNs raise privacy concerns when graphs contain personal data: for instance, social or financial networks involve sensitive information about individuals and their interactions. Recent works [19, 20, 33, 44] have extended the study of the privacy leakage of standard deep learning models to GNNs, showing the risk of information leakage regarding training data is even higher in GNNs, as they incorporate not only node features and labels but also the graph structure itself [9]. Consequently, GNNs are vulnerable to various privacy attacks, such as node membership inference [20, 33] and edge stealing [19, 44]. For example, a GNN trained on a social network for friendship recommendation could reveal the existing relationships between the users via its predictions. As another example, a GNN trained on the social graph of COVID-19 patients can be used by government authorities to predict the spread of the disease, but an adversary may recover private information about the participating patients.

Problem and motivation. Motivated by these privacy concerns, we investigate the problem of designing privacy-preserving GNNs for private, sensitive graphs. Our goal is to protect the sensitive graph structure and other accompanying data using the framework of Differential Privacy (DP) [10].In the context of graphs, two different variants of DP have been defined: *edge-level* and *node-level* DP [37]. Informally, an edge-level ϵ-DP algorithm have roughly the same output (as measured by ϵ) if one edge is removed from the input graph. This ensures that the algorithm's output does not reveal the existence of a particular edge in the graph. Correspondingly, node-level private algorithms conceal the presence of a particular node together with all its associated edges and attributes. Clearly, node-level DP is a stronger privacy definition, but it is harder to attain because it requires the algorithm's output distribution to hide much larger differences in the input graph.

Challenges. As GNNs utilize the structural information in the graph data, protecting data privacy in such models is more

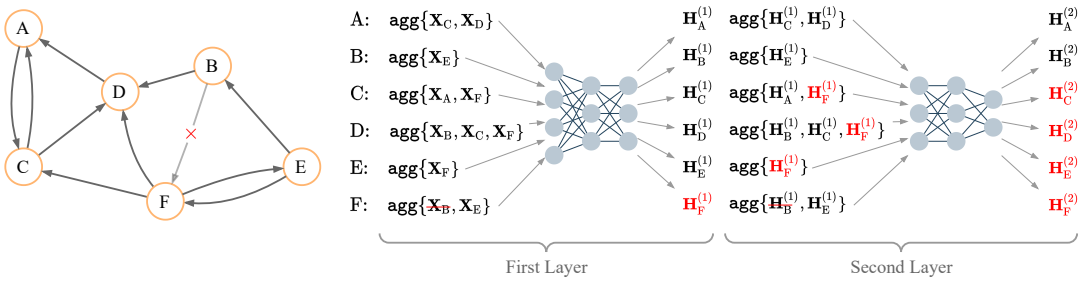
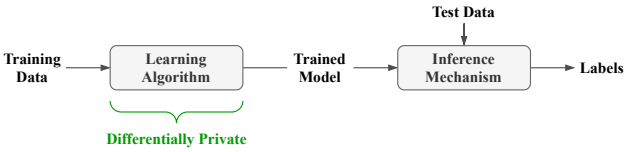


Figure 1: Schema of an unfolded 2-layer GNN taking an example graph as input. At each layer, every node aggregates its neighbors' embedding vectors (initially node features, e.g. $\mathbf{X}_A$ for node A), which is then updated using a neural net into a new vector (e.g., $\mathbf{H}_A$). Removing an arbitrary edge (here, the edge from node B to F) excludes the source node (B) from the aggregation set of the destination node (F). At the first layer, this will only alter the destination node's embedding, but this change is propagated to the neighboring nodes in the next layer. Node embeddings that are affected by the removal of edge (B,F) are indicated in red.

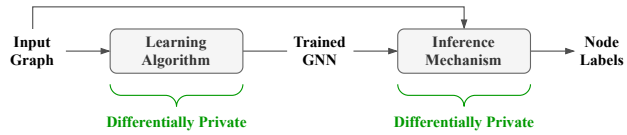
challenging than in standard ones. As shown in Figure 1, one of these challenges is the interdependency between the node embeddings resulting from the GNN's data aggregation mechanism. Specifically, a K-layer GNN iteratively learns node embeddings by aggregating information from every node's K-hop neighborhood (i.e., from nodes that are at a distance at most K in the graph). Hence, the embedding of a node is influenced not only by the node itself but also by all the nodes in its K-hop proximity. This fact voids the privacy guarantees of standard DP learning paradigms, such as DP-SGD [2], as the training loss of GNNs can no longer be decomposed into individual samples. Furthermore, the number of interdependent embeddings grows exponentially with K, hindering the ability of a DP solution to hide the output differences effectively. Therefore, how to get more representational power from higher-order GNN aggregations while ensuring DP is an important challenge to address.

Another major challenge is to guarantee *inference privacy*, i.e., preserving the privacy of graph data not only for training but also at inference time, when the trained GNN model is queried to make predictions for test nodes. Unlike conventional deep learning models, where the training data is not reused at inference time, the inference about any node in a K-layer GNN requires aggregating data from its K-hop neighborhood, which can reveal information about the neighboring nodes. Therefore, private graph data can still be leaked at inference time, even with privately trained model parameters. As a result, it is critical to ensure that both the training and inference stages of a GNN satisfy DP. This is illustrated in Figure 2.

Our contributions. To address the above challenges, we propose GAP, a privacy-preserving GNN model satisfying edge-level privacy, which is also extensible to node-level privacy if combined with standard private learning algorithms such as DP-SGD. As perturbing an edge in the input graph can practically be viewed as changing a sample in a node's neighborhood aggregation set, GAP preserves edge privacy via *aggregation perturbation*: we add calibrated Gaussian noise



(a) Learning standard DNNs with DP



(b) Learning GNNs with DP

Figure 2: Comparison of DP learning with (a) conventional deep neural networks, and (b) graph neural networks. Given the trained model, the inference mechanism of a DNN is independent of the training data, so a DP learning algorithm implies a DP inference mechanism as well. With GNNs however, graph data is queried again at inference time, so the inference step requires specific attention to be made differentially private.

to the output of the aggregation function, which can effectively hide the presence of a single edge (edge-level privacy) or a group of edges (node-level privacy). To avoid accumulating privacy costs at every model update, we propose a custom GNN architecture (Figure 3) comprising three individual components: (i) the encoder module, where we pre-train an encoder to extract lower-dimensional node features without relying on the graph structure; (ii) the aggregation module, where we use aggregation perturbation to privately compute multi-hop aggregated node embeddings using the graph edges and the encoded features; and (iii) the classification module, where we train a neural network on the aggregated data for node classification without further querying the graph edges.

Aggregation perturbation allows us to benefit from higher-

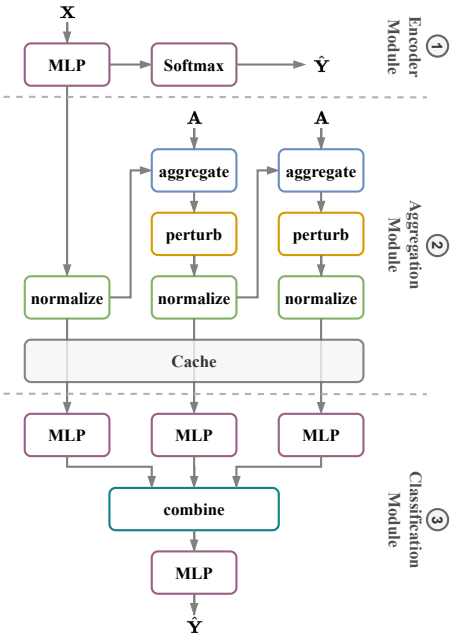


Figure 3: Overview of GAP's architecture: (1) The encoder is trained using only node features ($\mathbf{X}$) and labels ($\mathbf{Y}$). (2) The encoded features are given to the aggregation module to compute private K-hop aggregations (here, $K = 2$) using the graph's adjacency matrix ($\mathbf{A}$). (3) The classification module is trained over the private aggregations for label prediction.

order, multi-hop aggregations by composing individual noisy aggregations, yet the proposed architecture significantly reduces the privacy costs as the perturbed aggregations are computed once on lower-dimensional embeddings, and reused during training and inference. GAP also provides inference privacy, as the inference of any node relies on the perturbed aggregations, which hide information about neighboring nodes. Due to reusing cached aggregations, the inference step does not incur additional privacy costs beyond that of training.

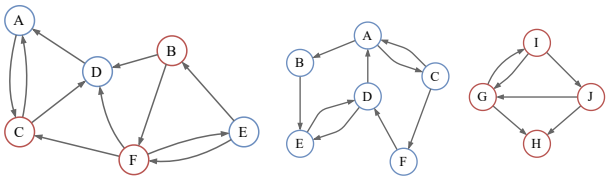
Results. We analyze GAP's formal privacy guarantees using Rényi Differential Privacy [29], and empirically evaluate its accuracy-privacy performance on three medium to large-scale graph datasets, namely Facebook, Reddit, and Amazon. We demonstrate that GAP's accuracy surpasses the competing baselines' at (very) low privacy budgets under both edge-level DP (e.g., $\epsilon \geq 0.1$ on Reddit) and node-level DP (e.g., $\epsilon \geq 1$ on Reddit), and observe that it always performs on par or better than a naive (privately trained) MLP model which does not utilize the graph's structural information.

2 Related Work

Graph neural networks. Deep learning on graphs has emerged in the past few years to tackle different kinds of graph-based learning tasks. A variety of GNN models and various architectures have been proposed, including Graph Convolutional Networks [26], Graph Attention Networks [41], GraphSAGE [16], Graph Isomorphism Networks [47], Jumping Knowledge Networks [48], and Gated Graph Neural Networks [28]. For the latest advances and trends in GNNs, we refer the reader to the available surveys [1, 17, 46, 53, 56].

Privacy attacks on GNNs. Several recent works have investigated the possibility of performing privacy attacks against GNNs and quantified the privacy leakage of publicly released GNN models or node embeddings trained on private graph datasets. Zhang *et al.* [55] study the information leakage in graph embeddings and propose three different inference attacks against GNNs: inferring graph properties (such as number of nodes and edges), inferring whether a given subgraph is contained in the target graph, and graph reconstruction with similar statistics to the target graph. He *et al.* [19] propose a series of black-box link stealing attacks on GNN models, and show that an adversary can accurately infer a link between any pair of nodes in a graph used to train the GNN. Zhang *et al.* [54] study the connection between model inversion risk and edge influence, and show that edges with greater influence are more likely to be inferred. Wu *et al.* [44] also study the link stealing attack via influence analysis, and propose an effective attack against GNNs based on the node influence information. The feasibility of the membership inference attack against GNNs has also been studied and several attacks with different threat models have been proposed in the literature [3, 9, 20, 33]. Overall, these works underline the privacy risks of GNNs trained on sensitive graph data and confirm the vulnerability of these models to various privacy attacks.

Differentially private GNNs. Recently, there have been attempts to use DP to provide formal privacy guarantees in various GNN learning settings. Sajadmanesh and Gatica-Perez [38] propose a locally private GNN model by considering a distributed learning setting, where node features and labels are private but training the GNN is federated by a central server with access to graph edges. However, their method cannot be used in applications where the graph edges are private. Wu *et al.* [44] propose an edge-level DP learning algorithm for GNNs by perturbing the input graph directly using either randomized response (called EdgeRand) or the Laplace mechanism (called LapGraph). Then, a GNN is trained over the resulting noisy graph. However, their method cannot be extended trivially to the node-level privacy setting. Olatunji *et al.* [32] consider a centralized learning setting and propose a node-level private GNN by adapting the framework of PATE [34]. They train the student GNN model using public graph data, which is privately labeled using the teacher GNN models trained exclusively for each query node. However, their dependence on public graph data restricts the applicability of their method. Daigavane *et al.* [7] also propose a node-level private approach for training 1-layer GNNs by extending the



(a) Transductive Learning (b) Inductive Learning

Figure 4: (a) Transductive learning: training and inference steps are conducted on the same graph, but different nodes are used for training and testing. Here, the blue nodes (A, D, and E) are used for training and the red nodes (B, C, and F) for inference. (b) Inductive learning: training and inference steps are performed on different graphs. Here, the left and right graphs are used for training and inference, respectively.

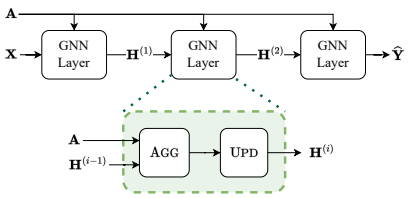


Figure 5: Typical 3-layer GNN for node classification. Each layer i takes the adjacency matrix $\mathbf{A}$ and previous layer's node embedding matrix $\mathbf{H}^{(i-1)}$ (initially, node features $\mathbf{X}$), and outputs a new embedding matrix $\mathbf{H}^{(i)}$ (ultimately, predicted class labels $\widehat{\mathbf{Y}}$). Internally, the input embeddings $\mathbf{H}^{(i-1)}$ are aggregated based on the adjacency matrix $\mathbf{A}$, and then fed to a neural network (UPD) to generate new embeddings $\mathbf{H}^{(i)}$.

standard DP-SGD algorithm and privacy amplification by subsampling results to bounded-degree graph data. However, their approach fails to provide inference privacy and is limited to 1-layer GNNs and thus cannot leverage higher-order aggregations.

Comparison with existing methods. To our best knowledge, GAP is the first approach providing both edge-level or node-level privacy guarantees based on the application requirements. Unlike existing methods, our approach does not rely on public data, can leverage multi-hop aggregations beyond first-order neighbors, and guarantees inference privacy at no additional cost. In Section 7, we also show that GAP outperforms other baselines in terms of accuracy-privacy trade-off.

3 Background and Problem Formulation

3.1 Graph Neural Networks

GNNs aim to learn a representation for every node in the input graph by incorporating the initial node features and the graph structure (edges). The learned node representations, or embeddings, can then be used for the downstream machine learning task. In this paper, we focus on node classification, where the embeddings are used to predict the label of the graph nodes. Node-wise prediction problems can be tackled in either transductive or inductive setting. In the transductive setting, both training and testing are performed on the same graph, but different nodes are used for training and testing. Conversely, in the inductive setting, training and testing are performed on different graphs. This is illustrated in Figure 4.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y})$ be an unweighted directed graph dataset consisting of sets of nodes $\mathcal{V}$ and edges $\mathcal{E}$ represented by a binary adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, where $N = |\mathcal{V}|$ denotes the number of nodes, and $\mathbf{A}_{i,j} = 1$ if there is a directed edge $(i,j) \in \mathcal{E}$ from node i to node j. Nodes are characterized by d-dimensional feature vectors stacked up in an $N \times d$ matrix $\mathbf{X}$, where $\mathbf{X}_v$ denotes the feature vector of the v-th node. $\mathbf{Y} \in \{0,1\}^{N \times C}$ represents the labels of the nodes, where $\mathbf{Y}_v$ is a C-dimensional one-hot vector denoting the label of the v-th node, and C is the number of classes. Note that in the transductive learning setting, only a subset $\mathcal{V}_T \subset \mathcal{V}$ of the nodes is labeled, and thus $\mathbf{Y}_v$ is a zero vector for all $v \notin \mathcal{V}_T$.

A typical K-layer GNN consists of K sequential graph convolution layers. Layer i receives node embeddings from layer $i-1$ and outputs a new embedding for each node by aggregating the current embeddings of its adjacent neighbors followed by a learnable transformation, as defined below:

$$\mathbf{H}_v^{(i)} = \mathtt{upd}\left(\mathtt{agg}\left(\{\mathbf{H}_u^{(i-1)} : \forall u \in \mathfrak{N}_v\}\right); \mathbf{\Theta}^{(i)}\right),$$

where $\mathfrak{N}_v = \{u : \mathbf{A}_{u,v} \neq 0\}$ denotes the set of adjacent nodes to node v (i.e., nodes with outbound edges toward v), and $\mathbf{H}_u^{(i-1)}$ is the embedding of an adjacent node u at layer $i-1$. $\mathtt{agg}(\cdot)$, is a (sub)differentiable, permutation invariant aggregator function, such as SUM, MEAN, or MAX. Finally, $\mathtt{upd}(\cdot)$ is a learnable function, such as a multi-layer perceptron (MLP), parameterized by $\mathbf{\Theta}^{(i)}$ that takes the aggregated vector and outputs the new embedding $\mathbf{H}_v^{(i)}$. For convenience, we define the matrix-based version of $\mathtt{agg}(\cdot)$ and $\mathtt{upd}(\cdot)$ by stacking the corresponding vectors of all the nodes into a matrix as:

$$\textsc{Agg}(\mathbf{H}, \mathbf{A}) = \left[\mathtt{agg}\left(\{\mathbf{H}_u : \forall u \in \mathfrak{N}_v\}\right) : \forall v \in \mathcal{V}\right]^T,$$

$$\textsc{Upd}(\mathbf{M}; \mathbf{\Theta}) = \left[\mathtt{upd}\left(\mathbf{M}_v; \mathbf{\Theta}\right) : \forall v \in \mathcal{V}\right]^T,$$

where we omitted the layer indicator superscripts for simplicity. Initially, we have $\mathbf{H}^{(0)} = \mathbf{X}$ (i.e., node features) as the input to the GNN's first layer. The last layer generates an output embedding vector for each node, which can be used in different ways depending on the downstream task. For node classification, a softmax layer is applied to the final embeddings $\mathbf{H}^{(K)}$ to obtain the posterior class probabilities $\widehat{\mathbf{Y}}$. The illustration of a typical 3-layer GNN is depicted in Figure 5.

3.2 Differential Privacy

Differential privacy (DP) [11] is the gold standard for formalizing the privacy guarantees of algorithms that process

sensitive data. Informally, DP requires that the algorithm's output distribution be roughly the same regardless of the presence of an individual's data in the dataset. As such, an adversary having access to the data of all but the target individual cannot distinguish whether the target's record is among the input data. The formal definition of DP is as follows.

Definition 1 (Differential Privacy [11]). *Given $\epsilon > 0$ and $\delta > 0$, a randomized algorithm $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy, if for all possible pairs of adjacent datasets X and X' differing by at most one record, denoted as $X \sim X'$, and for any possible set of outputs $S \subseteq Range(\mathcal{A})$, we have:*

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

Here, the parameter ϵ is called the *privacy budget* (or privacy cost) and is used to tune the privacy-utility trade-off of the algorithm: a lower privacy budget leads to stronger privacy guarantees but reduced utility. The parameter δ is informally treated as a failure probability, and is usually chosen to be very small. DP has the following important properties that help us design complex algorithms from simpler ones [10]:

- *Robustness to post-processing:* Any post-processing of the output of an (ϵ, δ)-DP algorithm remains (ϵ, δ)-DP.
- *Sequential composition:* If an (ϵ, δ)-DP algorithm is applied k times on the same data, the result is at most $(k\epsilon, k\delta)$-DP.
- *Parallel composition:* Executing an (ϵ, δ)-DP algorithm on disjoint chunks of data yields an (ϵ, δ)-DP algorithm.

In this paper, we use an alternative definition of DP, called *Rényi Differential Privacy* (RDP) [29], which allows obtaining tighter sequential composition results:

Definition 2 (Rényi Differential Privacy [29]). *A randomized algorithm $\mathcal{A}$ is (α, ϵ)-RDP for $\alpha > 1, \epsilon > 0$ if for every adjacent datasets $X \sim X'$, we have $D_\alpha(\mathcal{A}(X) \| \mathcal{A}(X')) \leq \epsilon$, where $D_\alpha(P\|Q)$ is the Rényi divergence of order α between probability distributions P and Q defined as:*

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

As RDP is a generalization of DP, it can be easily converted back to standard (ϵ, δ)-DP using the following proposition:

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

A basic method to achieve RDP is the *Gaussian mechanism*, where Gaussian noise is added to the output of the algorithm we want to make private. Specifically, let $f : \mathcal{X} \to \mathbb{R}^d$ be the non-private algorithm taking a dataset as input and outputting a d-dimensional vector. Let the *sensitivity* of f be the maximum L_2 distance achievable when applying $f(\cdot)$ to adjacent datasets X and X' as $\Delta_f = \max_{X \sim X'} \|f(X) - f(X')\|_2$. Then, adding Gaussian noise with variance σ^2 to f as $\mathcal{A}(X) = f(X) + \mathcal{N}(\sigma^2 \mathbb{I}_d)$, with $\mathbb{I}_d$ being $d \times d$ identity matrix, yields an (α, ϵ)-RDP algorithm for all $\alpha > 1$ with $\epsilon = {\Delta_f^2 \alpha}/{2\sigma^2}$ [29].

3.3 Problem Definition

Let $\widehat{\mathbf{Y}} = \mathcal{F}(\mathbf{X}, \mathbf{A}; \boldsymbol{\Theta})$ be a GNN-based node classification model with parameter set $\boldsymbol{\Theta}$ that takes node features $\mathbf{X}$ and the graph's adjacency matrix $\mathbf{A}$ as input, and outputs the corresponding predicted labels $\widehat{\mathbf{Y}}$. To learn the model parameters $\boldsymbol{\Theta}$, we minimize a standard classification loss function (e.g., cross-entropy) with respect to $\boldsymbol{\Theta}$ as follows:

$$\boldsymbol{\Theta}^{\star} = \arg\min_{\boldsymbol{\Theta}} \sum_{v \in \mathcal{V}_T} \ell(\widehat{\mathbf{Y}}_v, \mathbf{Y}_v), \tag{1}$$

where $\ell(\cdot, \cdot)$ is the loss function, $\mathbf{Y}$ is the ground-truth labels, and $\mathcal{V}_T \subseteq \mathcal{V}$ is the set of labeled training nodes. After training, in the transductive setting, the learned GNN is used to infer the labels of unlabeled nodes in $\mathcal{G}$:

$$\widehat{\mathbf{Y}} = \mathcal{F}(\mathbf{X}, \mathbf{A}; \boldsymbol{\Theta}^{\star}), \tag{2}$$

Otherwise, in the inductive setting, a new graph dataset $\mathcal{G}_{test}$ is given to the learned GNN for label inference.

The goal of this paper is to preserve the privacy of graph datasets for both the training step (Eq. 1) and the inference step (Eq. 2) using differential privacy. Note that preserving privacy in the inference step is critical as the adjacency information is still used in this step for obtaining the predicted labels.

However, as graph datasets are different from standard tabular datasets due to the existence of links between data records, one needs to adapt the definition of DP to graphs. As the semantic interpretation of DP relies on the definition of adjacent datasets, we first define two different notions of adjacency in graphs, namely edge-level and node-level adjacent graph datasets [18]:

Definition 3 (Edge-level adjacent graphs). *Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are edge-level adjacent if one can be obtained by removing a single edge from the other. Therefore, $\mathcal{G}$ and $\mathcal{G}'$ differ by at most one edge.*

Definition 4 (Node-level adjacent graphs). *Two graphs $\mathcal{G}$ and $\mathcal{G}'$ are node-level adjacent if one can be obtained by removing a single node (with its features, labels, and all attached edges) from the other. Therefore, $\mathcal{G}$ and $\mathcal{G}'$ differ by at most one node.*

Accordingly, the definition of edge-level and node-level DP follows from the above definitions: an algorithm $\mathcal{A}$ is edge-level (respectively, node-level) (ϵ, δ)-DP if for every two edge-level (respectively, node-level) adjacent graph datasets $\mathcal{G}$ and $\mathcal{G}'$ and any set of outputs $S \subseteq Range(\mathcal{A})$, we have $\Pr[\mathcal{A}(\mathcal{G}) \in S] \leq e^{\epsilon} \Pr[\mathcal{A}(\mathcal{G}') \in S] + \delta$.

Intuitively, edge-level DP protects edges (which could represent connections between people), while node-level DP protects nodes together with their adjacent edges (i.e., all information pertaining to an individual, including features, labels, and connections).

4 Proposed Method: GAP

In this section, we explain our proposed differentially private method, called GNN with Aggregation Perturbation (GAP), which guarantees both edge-level and node-level privacy for training and inference on sensitive graph data.

4.1 Overview

As mentioned in Section 1, the two primary challenges in the design of private GNNs come from the use of higher-order aggregations and the need to ensure inference privacy. To tackle these challenges, we propose a new architecture for GAP, which is different from the conventional GNN architectures presented in Section 3.1. The key distinction is that GAP decouples the graph-based aggregations from the neural network-based transformations, which is similar in spirit to the Inception model and scalable networks [13, 39, 45]. As illustrated in Figure 3, GAP is composed of the following three components:

(i) *Encoder Module (EM):* This module encodes the input node features into a lower-dimensional representation without using the private graph structure.

(ii) *Aggregation Module (AM):* This module takes the encoded low-dimensional node features and recursively computes private multi-hop aggregations using the *aggregation perturbation* approach, i.e., by adding noise to the output of each aggregation step.

(iii) *Classification Module (CM):* This module takes the privately aggregated node features and predicts the corresponding labels without querying the edges any further.

GAP's privacy mechanism. Our proposed mechanism for preserving the privacy of graph edges in AM is the *aggregation perturbation* approach: we use the Gaussian mechanism to add stochastic noise to the output of the aggregation function proportional to its sensitivity. This approach is motivated by the fact that perturbing an edge in the input graph can practically be viewed as changing a sample in the neighborhood aggregation function of the edge's destination node. Therefore, by adding an appropriate amount of noise to the aggregation function, we can effectively hide the presence of a single edge, which ensures edge-level privacy, or a group of edges, which is necessary for node-level privacy. To fully guarantee node-level privacy, however, in addition to the edges, we need to also protect node features and labels, which is simply done by training EM and CM using standard DP learning algorithms such as DP-SGD. We discuss this point further in Section 5.

Challenges addressed. Our GAP method can benefit from multi-hop aggregations by composing individual noisy aggregation steps. As the sensitivity of a single-step aggregation is easily determined, AM applies the Gaussian mechanism immediately after each aggregation step, avoiding the growing interdependency between node embeddings. GAP also provides inference privacy as the inference of a node relies on the aggregated data from its neighbors, which is privately computed by AM. As the subsequent CM only post-processes these private aggregations, GAP ensures inference-time privacy. This is explained in more details in Section 5.

In the rest of this section, we first discuss each of the GAP's components thoroughly and then describe the inference mechanism.

4.2 Encoder Module

GAP uses a multi-layer perceptron (MLP) model as an encoder to transform the original node features into an intermediate representation given to AM. The main goal of this module is to reduce the dimensionality of AM's input, as the magnitude of the Gaussian noise injected into the aggregations grows with data dimensionality. Therefore, reducing the dimensionality helps achieve better aggregation utility under DP.

Note that in order to save the privacy budget spent in AM, we do not train the encoder end-to-end with CM. Instead, we attach a linear softmax layer to the encoder MLP for label prediction, and then pre-train this model separately using node features and labels. Specifically, we use the following model:

$$\widehat{\mathbf{Y}} = \text{softmax}\left(\text{MLP}_{\text{enc}}(\mathbf{X}; \boldsymbol{\Theta}_{\text{enc}}) \cdot \mathbf{W}\right), \tag{3}$$

where MLP_{enc} is the encoder MLP with parameter set $\boldsymbol{\Theta}_{\text{enc}}$, $\mathbf{W}$ is the weight matrix of the linear softmax layer, $\mathbf{X}$ is the original node features, and $\widehat{\mathbf{Y}}$ is the corresponding posterior class probabilities. In order to train this model, we minimize the cross-entropy (or any other classification-related) loss function $\ell(\cdot, \cdot)$ with respect to the model parameters $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_{\text{enc}}, \mathbf{W}\}$:

$$\boldsymbol{\Theta}^\star = \arg\min_{\boldsymbol{\Theta}} \sum_{v \in \mathcal{V}_T} \ell(\widehat{\mathbf{Y}}_v, \mathbf{Y}_v), \tag{4}$$

where $\mathbf{Y}$ is the ground-truth labels and $\mathcal{V}_T \subseteq \mathcal{V}$ is the set of training nodes. After pre-training, we use the encoder MLP to extract low-dimensional node features, $\mathbf{X}^{(0)}$, for AM:

$$\mathbf{X}^{(0)} = \text{MLP}_{\text{enc}}(\mathbf{X}; \boldsymbol{\Theta}^\star_{\text{enc}}). \tag{5}$$

Remark. As will be discussed in Section 4.3, this encoder pre-training approach significantly reduces the model's privacy costs as the private aggregations in AM no longer need to be updated with the encoder's parameters. Besides, compared to the original features, this approach provides better node features to AM as the encoded representations incorporate label information as well.

4.3 Aggregation Module

The goal of AM is to privately release multi-hop aggregated node features using the aggregation perturbation method. Algorithm 1 presents our mechanism, the Private Multi-hop

Aggregation (PMA). It relies on the Sum aggregation function, which is simply equivalent to the multiplication of the adjacency matrix $\mathbf{A}$ by the input feature matrix $\mathbf{X}$, as $\textsc{Agg}(\mathbf{X}, \mathbf{A}) = \mathbf{A}^T \cdot \mathbf{X}$. The PMA mechanism takes $\check{\mathbf{X}}^{(0)}$, the row-normalized version of the encoder's extracted features as:

$$\check{\mathbf{X}}_v^{(0)} = \mathbf{X}_v^{(0)} / \|\mathbf{X}_v^{(0)}\|_2, \quad \forall v \in \mathcal{V}. \tag{6}$$

It then outputs a set of K normalized, privately aggregated node features $\check{\mathbf{X}}^{(1)}$ to $\check{\mathbf{X}}^{(K)}$ corresponding to different hops from 1 to K. Specifically, given $\sigma > 0$, the PMA mechanism performs the following steps to recursively compute and perturb the aggregations in k-th hop from $(k-1)$-th:

1. **Aggregation:** First, we compute k-th non-private aggregations using the normalized aggregations at step $k-1$:

$$\mathbf{X}^{(k)} = \mathbf{A}^T \cdot \check{\mathbf{X}}^{(k-1)}. \tag{7}$$

2. **Perturbation:** Next, we perturb the aggregations using the Gaussian mechanism, i.e., by adding noise with variance σ^2 to every row of $\mathbf{X}^{(k)}$ independently:

$$\widetilde{\mathbf{X}}_v^{(k)} = \mathbf{X}_v^{(k)} + \mathcal{N}(\sigma^2 \mathbb{I}), \quad \forall v \in \mathcal{V}. \tag{8}$$

3. **Normalization:** Finally, it is essential to bound the effect of each feature vector on the subsequent aggregations. Therefore, we again row-normalize the private aggregated features, such that the L2-norm of each row is 1:

$$\check{\mathbf{X}}_v^{(k)} = \widetilde{\mathbf{X}}_v^{(k)} / ||\widetilde{\mathbf{X}}_v^{(k)}||_2, \quad \forall v \in \mathcal{V}. \tag{9}$$

Remark. The recursive computation of aggregations in the PMA mechanism has one advantage: each aggregation step acts as a denoising mechanism, averaging out the DP noise added in the previous step (to some extent). Therefore, part of the injected noise is dampened by the PMA mechanism itself, leading to better aggregation utility. This noise-reducing effect of GNN aggregations is also observed in prior work [38].

Effect of EM. Note that EM plays a critical role in improving AM's privacy-utility trade-off: First, it increases the utility of noisy aggregations by reducing the dimensionality of AM's input, resulting in less noise added to the aggregations. Second, its pre-training strategy makes AM agnostic to model training, which remarkably reduces the total privacy costs as the PMA mechanism is called only once and its output is cached to be reused for entire training and inference. Technically, this implies that with T training iterations, the Gaussian mechanism is composed only K times, which would otherwise be KT in the case of end-to-end training. Since K is small ($1 \leq K \leq 5$) compared to T (in the order of hundreds), this leads to a substantial reduction in the privacy budget.

4.4 Classification Module

Given the list of private aggregated features $\{\check{\mathbf{X}}^{(0)}, \ldots, \check{\mathbf{X}}^{(K)}\}$ provided by AM, the goal of CM is to predict node labels without further relying on the graph edges. To this end, for each $k \in \{0, 1, \ldots, K\}$, we first obtain the k-hop representation $\mathbf{H}^{(k)}$ using a corresponding base MLP, denoted as $\text{MLP}_{\text{base}}^{(k)}$:

$$\mathbf{H}^{(k)} = \text{MLP}_{\text{base}}^{(k)}(\check{\mathbf{X}}^{(k)}; \boldsymbol{\Theta}_{\text{base}}^{(k)}), \tag{10}$$

where $\boldsymbol{\Theta}_{\text{base}}^{(k)}$ is the parameters of $\text{MLP}_{\text{base}}^{(k)}$. Next, we combine these representations to get an integrated node embedding $\mathbf{H}$:

$$\mathbf{H} = \textsc{Combine}\left(\{\mathbf{H}^{(0)}, \mathbf{H}^{(1)}, \ldots, \mathbf{H}^{(K)}\}; \boldsymbol{\Theta}_{\text{comb}}\right), \tag{11}$$

where Combine is any differentiable combination strategy, with common choices being summation, concatenation, or attention, potentially with parameter set $\boldsymbol{\Theta}_{\text{comb}}$. Finally, we feed the integrated representation into a head MLP, denoted as MLP_{head}, to get posterior class probabilities for the nodes:

$$\widehat{\mathbf{Y}} = \text{MLP}_{\text{head}}(\mathbf{H}; \boldsymbol{\Theta}_{\text{head}}), \tag{12}$$

where $\boldsymbol{\Theta}_{\text{head}}$ denotes the parameters of MLP_{head}. To train CM, we minimize a similar loss function as Eq. 4 but with respect to CM's parameters: $\boldsymbol{\Theta} = \{\boldsymbol{\Theta}_{\text{base}}^{(0)}, \ldots, \boldsymbol{\Theta}_{\text{base}}^{(K)}, \boldsymbol{\Theta}_{\text{comb}}, \boldsymbol{\Theta}_{\text{head}}\}$. The overall training procedure of GAP is presented in Algorithm 2.

Remark. CM independently processes the information encoded in the graph-agnostic node features $\check{\mathbf{X}}^{(0)}$ and the private, graph-based aggregated features $\check{\mathbf{X}}^{(1)}$ to $\check{\mathbf{X}}^{(K)}$, combining them together to get an integrated node representation. Therefore, even if the DP noise overwhelms the signal in the higher-level aggregations, the information in the lower-level aggregations and/or the graph-agnostic features is still preserved and exploited for classification. As a result, regardless of the privacy budget, GAP is expected to always perform on par or better than pure MLP-based models that do not rely on the graph structure. We will empirically demonstrate this point in our experiments.

Algorithm 1: Private Multi-hop Aggregation

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$; initial normalized features $\check{\mathbf{X}}^{(0)}$; max hop K; noise variance σ^2;
Output : Private aggregated node feature matrices $\check{\mathbf{X}}^{(1)}, \ldots, \check{\mathbf{X}}^{(K)}$

```
for k ∈ {1, ..., K} do
    X^(k) ← A^T · X̌^(k−1)                          // aggregate
    X̃^(k) ← X^(k) + N(σ²𝕀)                          // perturb
    for v ∈ V do
        X̌_v^(k) ← X̃_v^(k) / ||X̃_v^(k)||_2          // normalize
    end
end
return X̌^(1), ..., X̌^(K)
```

4.5 Inference Mechanism

GAP is compatible with both the transductive and the inductive inference, as discussed below.

Algorithm 2: GAP Training

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$; node features $\mathbf{X}$; node labels $\mathbf{Y}$; max hop K; noise variance σ^2;

Output : Trained model parameters $\{\mathbf{\Theta}^\star_{\text{enc}}, \mathbf{\Theta}^{\star\,(0)}_{\text{base}}, \ldots, \mathbf{\Theta}^{\star\,(K)}_{\text{base}}, \mathbf{\Theta}^\star_{\text{comb}}, \mathbf{\Theta}^\star_{\text{head}}\}$;

1 Pre-train EM (Eq. 3) to obtain $\mathbf{\Theta}^\star_{\text{enc}}$.
2 Use the pre-trained encoder (Eq. 5) to obtain encoded features $\mathbf{X}^{(0)}$.
3 Row-normalize the encoded features (Eq. 6) to obtain $\breve{\mathbf{X}}^{(0)}$.
4 Use Algorithm 1 to obtain private aggregations $\tilde{\mathbf{X}}^{(1)}, \ldots, \tilde{\mathbf{X}}^{(K)}$.
5 Train CM (Eq. 10-12) to get $\mathbf{\Theta}^{\star\,(0)}_{\text{base}}, \ldots, \mathbf{\Theta}^{\star\,(K)}_{\text{base}}, \mathbf{\Theta}^\star_{\text{comb}}, \mathbf{\Theta}^\star_{\text{head}}$.
6 **return** $\{\mathbf{\Theta}^\star_{\text{enc}}, \mathbf{\Theta}^{\star\,(0)}_{\text{base}}, \ldots, \mathbf{\Theta}^{\star\,(K)}_{\text{base}}, \mathbf{\Theta}^\star_{\text{comb}}, \mathbf{\Theta}^\star_{\text{head}}\}$

Transductive setting. In this setting, both training and inference are conducted on the same graph, but using different nodes for training and inference steps (Figure 4a). As the entire graph is available at training time, AM computes the private aggregations of all the nodes, including both training and test ones. Therefore, at inference time, we only give the cached aggregations of the test nodes to the trained CM to predict their labels.

Inductive setting. Here, we use a new graph for inference different from the one used for training (Figure 4b). In this case, we first extract low-dimensional node features for the new graph using the pre-trained encoder and then feed them to AM to obtain the private aggregations. Finally, we input the private aggregations to the trained CM to get the node labels.

5 Privacy Analysis

5.1 Edge-Level Privacy

In the following, we provide a formal analysis of GAP's edge-level privacy guarantees at training and inference stages.

Training privacy. The following arguments establish the DP guarantees of the PMA mechanism and the GAP training algorithm. The detailed proofs can be found in Appendix A.

Theorem 1. *Given the maximum hop $K \geq 1$ and noise variance σ^2, the PMA mechanism presented in Algorithm 1 satisfies edge-level $(\alpha, {}^{K\alpha}/_{2\sigma^2})$-RDP for any $\alpha > 1$.*

Proposition 2. *For any $\delta \in (0, 1)$, maximum hop $K \geq 1$, and noise variance σ^2, Algorithm 2 satisfies edge-level (ϵ, δ)-DP with $\epsilon = \frac{K}{2\sigma^2} + {}^{\sqrt{2K\log(1/\delta)}}/_{\sigma}$.*

Proposition 2 shows that the privacy cost grows with the number of hops (K), but is independent of the number of training steps thanks to our GAP architecture.

Inference privacy. A major advantage of GAP is that querying the model at inference time preserves DP *without consuming additional privacy budget*. This is true for both the transductive and the inductive settings:

- *Transductive setting:* In this setting, the inference is performed by feeding the privately trained CM with the cached aggregations of the test nodes, which have already been computed privately at training time. As this computation does not query the private graph structure and only post-processes the previous DP operations, due to the robustness of DP to post-processing, GAP provides inference privacy with no additional cost.

- *Inductive setting:* In this case, first the new graph's node features are given to the encoder to obtain low-dimensional features, which are fed to AM to compute private aggregations. Then, the private aggregations are given to CM to obtain the final predictions. The only part where the private graph structure is queried is the AM, in which the PMA mechanism is applied to the new graph data, and thus the output is private. Furthermore, since the training and test graphs are disjoint, this application of the PMA mechanism is subject to the parallel composition of differentially private mechanisms, and thus it does not increase the privacy costs beyond that of training's. The other parts, the encoder and CM, perform graph-agnostic computations and only post-process previous DP outputs, leading to GAP ensuring inference privacy without extra privacy costs.

5.2 Node-Level Privacy

Equipped with aggregation perturbation, the proposed GAP architecture guarantees edge-level privacy by default. However, it is readily extensible to provide node-level privacy guarantees as well, providing that we have bounded-degree graphs, i.e., the degree of each node should be bounded above by a constant D. This allows to bound the sensitivity of the aggregation function in the PMA mechanism when adding/removing a node, as in this case each node can influence at most D other nodes. If the input graph has nodes with very high degrees, we can use neighbor sampling (as proposed in [7]) to randomly sample at most D neighbors per node.

For bounded-degree graphs, adding or removing a node corresponds (in the worst case) to adding or removing D edges. Therefore, our PMA mechanism also ensures node-level privacy, albeit with increased privacy costs compared to the edge-level setting (see Theorem 2 below).

However, since the node features and labels are also private under node-level DP, both EM and CM need to be trained privately as they access node features/labels. To this end, we can simply use standard DP-SGD [2] or any other differentially private learning algorithm for pre-training the encoder as well as training CM with DP. In other words, steps 1 and 5 of Algorithm 2 must be done with DP instead of regular non-private training. This way, since each of the three GAP modules become node-level private, the entire GAP model, as an adaptive composition of several node-level private mechanisms, satisfies node-level DP. The formal node-level privacy

analysis of GAP's training and inference is provided below.

Training privacy. The node-level privacy guarantees of the PMA mechanism and the GAP training algorithm are as follows. Detailed proofs are deferred to Appendix A.

Theorem 2. *Given the maximum degree $D \geq 1$, maximum hop $K \geq 1$, and noise variance σ^2, Algorithm 1 (PMA mechanism) satisfies node-level $(\alpha, {}^{DK\alpha}/_{2\sigma^2})$-RDP for any $\alpha > 1$.*

Proposition 3. *For any $\alpha > 1$, let encoder pre-training (Step 1 of Algorithm 2) and CM training (Step 5 of Algorithm 2) satisfy $(\alpha, \epsilon_1(\alpha))$-RDP and $(\alpha, \epsilon_5(\alpha))$-RDP, respectively. Then, for any $0 < \delta < 1$, maximum hop $K \geq 1$, maximum degree $D \geq 1$, and noise variance σ^2, Algorithm 2 satisfies node-level (ϵ, δ)-DP with $\epsilon = \epsilon_1(\alpha) + \epsilon_5(\alpha) + {}^{DK\alpha}/_{2\sigma^2} + {}^{\log(1/\delta)}/_{\alpha-1}$.*

Note that in Proposition 3, we cannot optimize α in closed form as we do not know the precise form of $\epsilon_1(\alpha)$ and $\epsilon_5(\alpha)$. However, in our experiments, we numerically optimize the choice of α on a per-case basis.

Inference privacy. The arguments stated for edge-level inference privacy also hold for node-level privacy. Note that in the inductive setting, the test graph should also have bounded degree for the node-level inference privacy guarantees to hold.

6 Discussion

Choice of aggregation function. In this paper, we used Sum as the default choice of aggregation function. Although other choices of aggregation functions are also possible, we empirically found that Sum is the most efficient choice to privatize, as its sensitivity does not depend on the size of the aggregation set (i.e., number of neighbors), which is itself a quantity that should be computed privately. For example, the calculation of both Mean and GCN [26] aggregation functions depend on the node degrees, and thus requires additional privacy budget to be spent on perturbing node degrees. In any case, Sum is recognized as one of the most expressive aggregation functions in the GNN literature [6, 47].

Normalization instead of clipping. The PMA mechanism uses normalization to bound the effect of each individual feature on the Sum aggregation function. While clipping is more common in the private learning literature (e.g., gradient clipping in DP-SGD [2]), we empirically found that normalization is a better choice for aggregation perturbation: CM is then trained on normalized data, which tends to facilitate learning. Normalizing the node embeddings is actually commonly done in non-private GNNs as well to stabilize training [16, 50].

Limitations. As the PMA mechanism adds random noise to the aggregation function, its utility naturally depends on the size of the node's aggregation set, i.e., the node's degree. Specifically, with a certain amount of noise, the more inbound neighbors a node has, the more accurate its noisy aggregated vector will be. This implies that graphs with higher average degree per node can tolerate larger noise in the aggregation function, and thus GAP can achieve a better privacy-accuracy trade-off on such graphs. Conversely, GAP's performance will suffer if the average degree of the graph is too low, requiring higher privacy budgets to achieve acceptable accuracy. Note however that this is an expected behavior: nodes with fewer inbound neighbors are more easily influenced by a change in their neighborhood compared to nodes with higher degrees, and thus the privacy of low-degree nodes is harder to preserve than high-degree ones. Furthermore, this limitation is not specific to GAP: it is shared by all DP algorithms, whose performance generally suffer from lack of sufficient data.

Edge-level vs. node-level privacy. While GAP can work in either edge-level or node-level privacy settings, it must be emphasized that the former setting is suitable only for the use cases where the node-level information (e.g, features or labels) is not sensitive or is publicly available (e.g., the vertically partitioned graph setting described in [44]). Whenever node-level information is private as well (e.g., user profiles in a social network), however, edge-level privacy fails to provide appropriate privacy protection, and thus node-level privacy setting has to be enforced.

7 Experiments

In this section, we conduct extensive experiments to empirically evaluate GAP's privacy-accuracy performance and its resilience under privacy attacks. As GAP's privacy guarantees are the same under both transductive and inductive settings, we only focus on the former, which has also more pertinent use cases (e.g., social networks).

7.1 Datasets

We evaluate the proposed method on three publicly available node classification datasets, which are medium to large scale in terms of the number of nodes and edges:

Facebook [40]. This dataset contains the anonymized Facebook social network between UIUC students collected in September 2005. Nodes represent Facebook users and edges indicate friendship. Each node (user) has the following attributes: student/faculty status, gender, major, minor, and housing status, and the task is to predict the class year of users.

Reddit [16]. This dataset consist of a set of posts from the Reddit social network, where each node represents a post and an edge indicates if the same user commented on both posts. Node features are extracted based on the embedding of the post contents, and the task is to predict the community (subreddit) that a post belongs to.

Amazon [5]. The largest dataset used in this paper represents Amazon product co-purchasing network, where nodes

Table 1: Overview of dataset statistics.

Dataset	Nodes	Edges	Degree	Features	Classes
Facebook	26,406	2,117,924	62	501	6
Reddit	116,713	46,233,380	209	602	8
Amazon	1,790,731	80,966,832	22	100	10

represent products sold on Amazon and an edge indicates if two products are purchased together. Node features are bag-of-words vectors of the product description followed by PCA, and the task is to predict the category of the products.

We preprocess the datasets by limiting the classes to those having 1k, 10k, and 100k nodes on Facebook, Reddit, and Amazon, respectively. We then randomly split the remaining nodes into training, validation, and test sets with 75/10/15% ratios, respectively. Table 1 summarizes the statistics of the datasets after preprocessing.

7.2 Competing Methods

Edge-level private methods. The following methods are evaluated under edge-level privacy:

- *GAP-EDP:* Our proposed edge-level DP algorithm.
- *SAGE-EDP:* This is the method of Wu *et al*. [44] that uses the graph perturbation approach, with the popular GraphSAGE architecture [16] as its backbone GNN model. We perturb the graph's adjacency matrix using the Asymmetric Randomized Response (ARR) [21], which performs better than EdgeRand [44] by limiting the output sparsity.
- *MLP:* A simple MLP model that does not use the graph edges, and thus provides perfect edge-level privacy ($\epsilon = 0$).

Node-level private methods. We compare the following node-level private algorithms:

- *GAP-NDP:* Our proposed node-level DP approach.
- *SAGE-NDP:* This is the method of Daigavane *et al*. [7] that adapts the standard DP-SGD method for 1-layer GNNs, with the same GraphSAGE architecture as its backbone model. Since this method does not inherently ensure inference privacy, as suggested by its authors, we add noise to the aggregation function based on its node-level sensitivity at test time and account for the additional privacy cost.
- *MLP-DP:* Similar to MLP, but trained with DP-SGD so as to provide node-level DP without using the graph edges.

We do not consider the approach of [32] as it requires public graph data and is thus not directly comparable to the others.

Non-private methods. To quantify the accuracy loss of private approaches, we use the following non-private methods ($\epsilon = \infty$):

- *GAP-∞:* a non-private counterpart of the GAP method, where we do not perturb the aggregations.
- *SAGE-∞:* a non-private GraphSAGE model.

7.3 Experimental Setup

Model implementation details. For our GAP models (GAP-EDP, GAP-NDP, and GAP-∞), we set the number of MLP_{enc}, MLP_{base}, and MLP_{head} layers to be 2, 1, and 1, respectively. We use concatenation as the Combine function (Eq. 11) and tune the number of hops K in $\{1, 2, \ldots, 5\}$. For the GraphSAGE models (SAGE-EDP, SAGE-NDP, and SAGE-∞), we use the Sum aggregation function and tune the number of message-passing layers in $\{1, 2, \ldots, 5\}$, except for SAGE-NDP that only supports one message-passing layer. We use a 2-layer and a 1-layer MLP as preprocessing and post-processing before and after the message-passing layers, respectively. For the MLP baselines (MLP and MLP-DP), we set the number of layers to 3. In addition, for both the GAP-NDP and SAGE-NDP methods, we use randomized neighbor sampling to bound the maximum degree D and search for the best D within $\{100, 200, 300, 400\}$. For all methods, we set the number of hidden units to 16 (including the dimension of GAP's encoded representation) and use the SeLU activation function [27] at every layer. Batch-normalization is used for all methods except the node-level private ones (GAP-NDP, SAGE-NDP, and MLP-DP), for which batch-normalization is not supported.

Training and evaluation details. We train the non-private and edge-level private methods using the Adam optimizer over 100 epochs with full-sized batches. For the node-level private algorithms (GAP-NDP, SAGE-NDP, MLP-DP), we use DP-Adam [15] with maximum gradient norm set to 1, and train each model for 10 epochs with a batch size of 256, 2048, 4096 on Facebook, Reddit, and Amazon, respectively. For our GAP models (GAP-∞, GAP-EDP, and GAP-NDP), we use the same parameter setting for training both the encoder and classification modules. We train all the methods with a learning rate of 0.01 and repeat each combination of possible hyperparameter values 10 times. We pick the best performing model based on validation accuracy, and report the average test accuracy with 95% confidence interval calculated by bootstrapping with 1000 samples.

Privacy accounting and calibration. Privacy budget accounting is done via the Analytical Moments Accountant [43]. We numerically calibrate the noise scale (i.e., the noise standard deviation σ divided by the sensitivity) of PMA (for GAP-EDP and GAP-NDP), ARR (for SAGE-EDP), DP-SGD (for GAP-NDP, SAGE-NDP, and MLP-DP) and the Gaussian mechanism (for inference privacy in SAGE-NDP) to achieve the desired (ϵ, δ)-DP. We report results for several values of ϵ, while δ is set to be smaller than the inverse number of private entities (i.e., edges for edge-level privacy, nodes for node-level privacy). For both GAP-NDP and SAGE-NDP, we use the same noise scale for perturbing the gradients (in DP-SGD) and the aggregations (in PMA and Gaussian mechanisms).

Software and hardware. All the models are implemented in PyTorch [35] using PyTorch-Geometric (PyG) [12]. We use

Table 2: Test accuracy of different methods on the three datasets. The best performing method in each category — none-private, edge-level DP and node-level DP — is highlighted.

	Method	ϵ	Facebook	Reddit	Amazon
None	GAP-∞	∞	80.0 ± 0.48	**99.4 ± 0.02**	91.2 ± 0.07
	SAGE-∞	∞	**83.2 ± 0.68**	99.1 ± 0.01	**92.7 ± 0.09**
Edge DP	GAP-EDP	4	**76.3 ± 0.21**	**98.7 ± 0.03**	**83.8 ± 0.26**
	SAGE-EDP	4	50.4 ± 0.69	84.6 ± 1.63	68.3 ± 0.99
	MLP	0	50.8 ± 0.17	82.4 ± 0.10	71.1 ± 0.18
Node DP	GAP-NDP	8	**63.2 ± 0.35**	**94.0 ± 0.14**	**77.4 ± 0.07**
	SAGE-NDP	8	37.2 ± 0.96	60.5 ± 1.10	27.5 ± 0.83
	MLP-DP	8	50.2 ± 0.25	81.5 ± 0.12	73.6 ± 0.05

the `autodp` library[1] which implements analytical moments accountant, and utilize Opacus [51] for training the node-level private models with differential privacy. Experiments are conducted on Sun Grid Engine with NVIDIA GeForce RTX 3090 and NVIDIA Tesla V100 GPUs, Intel Xeon 6238 CPUs, and 32 GB RAM.

7.4 Experimental Results

7.4.1 Trade-offs between Privacy and Accuracy

We first compare the accuracy of our proposed methods against the non-private, edge-level private, and node-level private baselines. We fix the privacy budget to $\epsilon = 8$ for the node-level private methods and $\epsilon = 4$ for the edge-level private ones (except for MLP, which does not use the graph structure and thus achieves $\epsilon = 0$). The results are presented in Table 2. We observe that in the non-private setting, the proposed GAP architecture is competitive with SAGE, with only a slight decrease in accuracy on Facebook and Amazon. Under both edge-level and node-level privacy settings, however, our proposed methods GAP-EDP and GAP-NDP significantly outperform their competitors. Particularly, under edge-level privacy, GAP-EDP's accuracy is roughly 26, 14, and 15 points higher than the best competitor over Facebook, Reddit, and Amazon, respectively. Under node-level privacy, our proposed GAP-NDP method outperforms the best performing competitor by approximately 13, 13, and 4 accuracy points, respectively.

Next, to investigate how different methods perform under different privacy budgets, we vary ϵ from 0.1 to 8 for edge-level private methods and from 1 to 16 for node-level private algorithms and report the accuracy of the methods under each privacy budget. The result for both edge-level and node-level privacy settings is depicted in Figure 6.

Under edge-level privacy (Figure 6, left side), we observe that GAP-EDP consistently outperforms its direct competitor, SAGE-EDP, especially at lower privacy costs. The relative gap between GAP-EDP and SAGE-EDP is influenced by the average degree of the dataset. For example, on Facebook and Reddit with higher average degrees, SAGE-EDP requires a high privacy budget of $\epsilon \geq 8$ to achieve reasonable accuracy, but on Amazon, which has the lowest average degree, it cannot even beat the MLP baseline. In comparison, the accuracy of GAP-EDP approaches the non-private GAP-∞ at much lower privacy budgets, and always performs better than a vanilla MLP. This is because SAGE-EDP perturbs the adjacency matrix, which is extremely high-dimensional and sparse, while GAP-EDP perturbs the aggregated node embeddings, which has much lower dimensions and is not sparse compared to the adjacency matrix. The amount of accuracy loss with respect to the non-private method also depends on the average degree of the graph. For example, on Reddit at $\epsilon = 2$, GAP-∞'s accuracy is only 1 point higher than GAP-EDP's, while on Amazon at $\epsilon = 8$, GAP-EDP's accuracy fall behind GAP-∞ by around 5 points. These observations are in line with our discussion of Section 6.

We can observe similar trends under node-level privacy (Figure 6, right side). We see that our GAP-NDP method always performs on par or better than the MLP-DP baseline, and also significantly outperforms SAGE-NDP under all the considered privacy budgets. We attribute this to two factors: first, SAGE-NDP is limited to 1-layer models and thus cannot exploit higher-order aggregations; second, the naive noisy aggregation patch for supporting inference privacy severely hurts the performance of SAGE-NDP. As expected, since the node-level private GAP-NDP hides more information (e.g., node features, labels, and all the adjacent edges to a node) than the edge-level private GAP-EDP, it requires larger privacy budgets to achieve a reasonable accuracy. Still, the accuracy loss with respect to the non-private method is higher in the node-level private method as we have further information loss due to neighborhood sampling (to bound the graph's maximum degree) and gradient clipping (to bound the sensitivity in DP-SGD/Adam).

7.4.2 Resilience Against Privacy Attacks

As mentioned above, the node-level private methods require a higher privacy budget than the edge-level private ones as they attempt to hide much more information. In order to assess the practical implications of choosing rather large privacy budgets (e.g., $\epsilon = 8$ in Table 2), we empirically measure the privacy guarantees of GAP-NDP and other node-level private methods by conducting node-level membership inference attack [20,33] as the most relevant adapted privacy attack to GNNs.

Attack overview. The attack is modeled as a binary classification task, where the goal is to infer whether an arbitrary node v is a member of the training set $\mathcal{V}_T$ of the target GNN. The key intuition is that due to overfitting, GNNs give more confident probability scores to training nodes than to test ones, which can be exploited by the attacker to distinguish members of the training set. Having access to a shadow graph dataset coming from the same distribution as the target graph, the attacker first trains a shadow GNN to mimic the behavior of the target

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

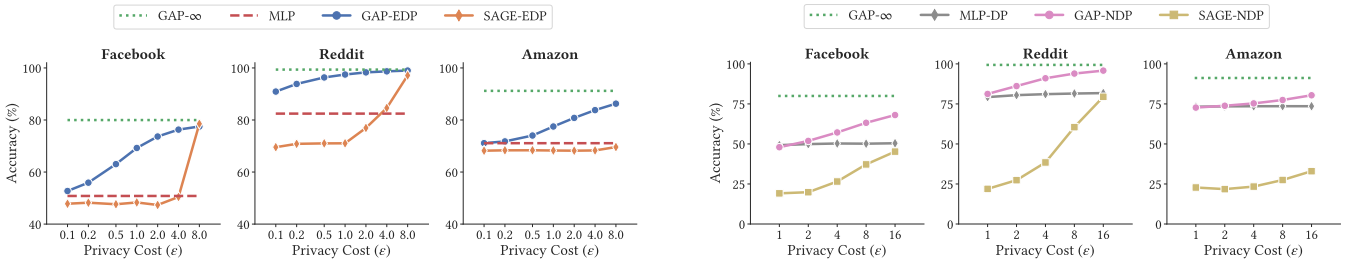


Figure 6: Accuracy vs. privacy cost (ϵ) of edge-level private algorithms (left) and node-level private methods (right).

GNN, but for which the membership ground truth is known. Then, the attacker trains an attack model over the probability scores of the shadow graph nodes and their corresponding membership labels. Finally, the attacker uses the trained attack model to infer the membership of the target graph nodes.

Attack settings. We follow the TSTF (train on subgraph, test on full graph) approach of [33] for the node-level membership inference attack. Specifically, we consider a strong adversary with access to a shadow graph dataset with 1000 nodes per class, which are sampled uniformly at random from the target dataset. For the shadow model, we use the same architecture and hyperparameters as the target model (described in Section 7.3). Similar to prior work [33], we use a 3-layer MLP with 64 hidden units as the attack model, and use the area under the receiver operating characteristic curve (AUC) averaged over 10 runs as the evaluation metric.

Results. Table 3 reports the mean AUC of the attack on different node-level private methods trained with the same setting as in Figure 6 (right). As we see, the attack is quite effective on the non-private methods ($\epsilon = \infty$), especially on Facebook and Amazon datasets. The success of the attack on each method mainly depends on its generalization gap (the difference between the training and test accuracy): the higher the generalization gap, the more confident the model is on the training nodes and the easier it is to distinguish them from the test nodes. Hence, the lower attack performance on the non-private SAGE method is due to its lower generalization gap compared to the other methods. Nevertheless, for all private GNN methods, we observe that DP with privacy budgets as large as $\epsilon = 16$ can effectively defend against the attack, reducing the AUC to about 50% (random baseline) on all datasets. This result is in line with the work of [22, 23, 31], showing that DP with large privacy budgets can still effectively mitigate realistic membership inference attacks.

Table 3: Mean AUC of node membership inference attack.

Dataset	Method	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 16$	$\epsilon = \infty$
Facebook	GAP-NDP	50.16	50.25	50.61	51.11	52.66	81.67
	SAGE-NDP	50.25	50.20	50.23	50.17	50.20	62.49
	MLP-DP	50.32	50.72	52.13	53.44	54.77	81.57
Reddit	GAP-NDP	50.04	50.39	51.20	52.23	52.54	54.97
	SAGE-NDP	49.97	49.97	49.95	50.00	49.98	50.05
	MLP-DP	51.25	53.09	55.13	56.72	58.32	71.35
Amazon	GAP-NDP	50.06	50.23	50.54	51.53	51.72	66.68
	SAGE-NDP	49.93	49.93	49.93	49.92	49.97	59.41
	MLP-DP	50.30	50.58	51.43	52.31	53.34	72.97

7.4.3 Ablation Studies

Effectiveness of the encoder module (EM). In this experiment, we investigate the effect of EM on the accuracy/privacy performance of the proposed methods, GAP-EDP and GAP-NDP. We compare the case in which EM is used as usual with the case where we remove EM and just input the original node features to the aggregation module. The results under different privacy budgets are given in Figure 7. We can observe that in all cases, the accuracy of GAP-EDP and GAP-NDP is higher with EM than without it. For example, leveraging EM results in a gain of around 20, 2, and 5 accuracy points for GAP-EDP with $\epsilon = 1$ on Facebook, Reddit, and Amazon datasets, respectively. GAP-NDP with EM also benefits from a gain of more than 10, 10, and 5 points with $\epsilon = 4$ on Facebook, Reedit, and Amazon datasets, respectively. As discussed in Section 4.2, the improved performance with EM is mainly due to the reduced dimensionality of the aggregation module's input, which leads to adding less noise to the aggregations. Also, the effect of EM is more significant on GAP-NDP, as the amount of noise injected into the aggregations is generally larger for node-level privacy, hence dimensionality reduction becomes more critical to mitigate the impact of noise.

Effect of the number of hops. In this experiment, we investigate how changing the number of hops K affects the accuracy/privacy performance of our proposed methods, GAP-EDP and GAP-NDP. We vary K within $\{1, 2, 3, 4, 5\}$ and report the accuracy under different privacy budgets: $\epsilon \in \{1, 4\}$ for GAP-EDP and $\epsilon \in \{8, 16\}$ for GAP-NDP. The result is depicted in Figure 8. We observe that both of our methods can effectively benefit from allowing multiple hops, but there is a trade-off in increasing the number of hops. As we increment K, the accuracy of both GAP-EDP and GAP-NDP method increase up to a point and then steady or decrease in almost

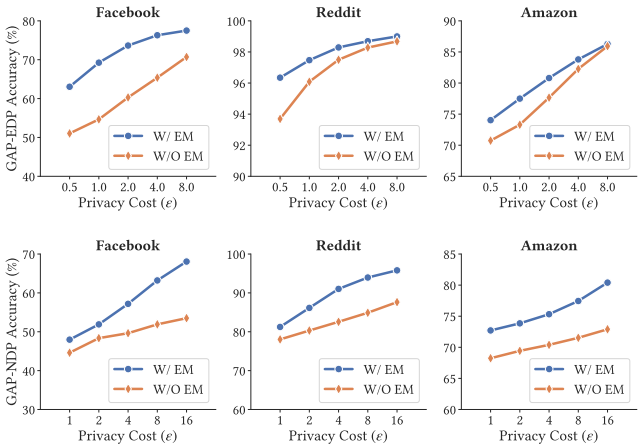


Figure 7: Effect of the encoder module (EM) on the accuracy/privacy performance of the edge-level private GAP-EDP (top) and the node-level private GAP-NDP (bottom).

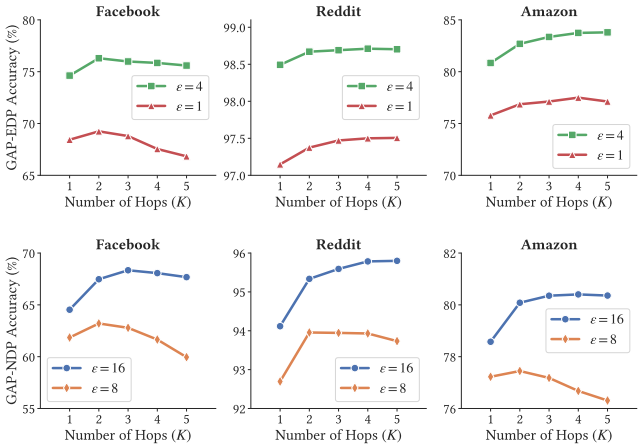


Figure 8: Effect of the number of hops K on the accuracy/privacy performance of the edge-level private GAP-EDP (top) and the node-level private GAP-NDP (bottom).

all cases. The reason is that with a larger K the model is able to utilize information from more distant nodes (all the nodes within the K-hop neighborhood of a node) for prediction, which can increase the final accuracy. However, as more hops are involved, the amount of noise in the aggregations is also increased, which adversely affects the model's accuracy. We can see that with the lower privacy budgets where the noise is more severe, both GAP-EDP and GAP-NDP achieve their peak accuracy at smaller K values. But as the privacy budget increases, the magnitude of the noise is reduced, enabling the models to benefit from larger K values.

Effect of the maximum degree. We now analyze the effect of D on the performance of our node-level private method. We vary D from 10 to 400 and report GAP-NDP's accuracy under two different privacy budgets $\epsilon \in \{4, 16\}$. Figure 9 shows

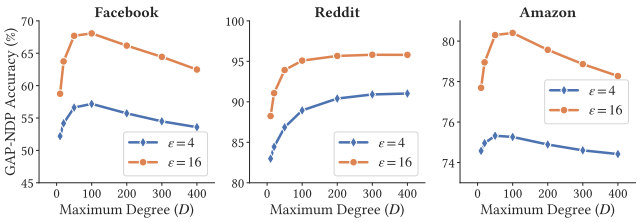


Figure 9: Effect of the degree bound D on the accuracy/privacy performance of the node-level private GAP-NDP method.

that the accuracy keeps growing with D on Reddit (which has a high average degree), while on Facebook and Amazon (lower average degrees) the accuracy increases with D up to a peak point, and drops afterwards. This is due to the trade-off between having more samples for aggregation and the amount of noise injected: the larger D, the fewer neighbors are excluded from the aggregations (i.e., less information loss), but on the other hand, the larger the sensitivity of the aggregation function, leading to more noise injection. We also observe that the accuracy gain as a result of increasing D gets bigger as the privacy budget is increased from 5 to 20, since a higher privacy budget compensates for the higher sensitivity by reducing the amount of noise.

8 Conclusion

In this paper, we presented GAP, a privacy-preserving GNN architecture that ensures both edge-level and node-level differential privacy for training and inference over sensitive graph data. We used aggregation perturbation, where the Gaussian mechanism is applied to the output of the GNN's aggregation function, as a fundamental technique to achieve DP in our approach. We proposed a new GNN architecture tailored to the specifics of private learning over graphs, aiming to achieve better privacy-accuracy trade-offs while tackling the intricate challenges involved in the design of differentially private GNNs. Experimental results over real-world graph datasets showed that our approach achieves favorable privacy/accuracy trade-offs and significantly outperforms existing methods. Promising future directions include: (i) investigating robust aggregation functions that provide specific benefits for private learning; (ii) exploiting the redundancy of information in recursive aggregations to achieve tighter composition when the number of hops K gets large, which might prove useful for specific applications; (iii) extending the framework to other tasks and scenarios, such as link-wise prediction or learning over dynamic graphs; and (iv) conducting an extended theoretical analysis of differentially private GNNs, such as proving utility bounds and characterizing their expressiveness.

Acknowledgments

This work was supported by the European Commission's Horizon 2020 Program ICT-48-2020, under grant number 951911, AI4Media project. It was also supported by the French National Research Agency (ANR) through grant ANR-20-CE23-0015 (Project PRIDE). Ali Shahin Shamsabadi acknowledges support from The Alan Turing Institute.

Availability

Our open-source implementation is publicly available on GitHub at https://github.com/sisaman/GAP.

References

[1] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. Computing graph neural networks: A survey from algorithms to accelerators. *ACM Computing Surveys (CSUR)*, 54(9):1–38, 2021.

[2] 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*, pages 308–318, 2016.

[3] Wu Bang, Yang Xiangwen, Pan Shirui, and Yuan Xingliang. Adapting membership inference attacks to gnn for graph classification: Approaches and implications. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021.

[4] Mark Cheung and José M. F. Moura. Graph neural networks for covid-19 drug discovery. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5646–5648, 2020.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 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*, pages 257–266, 2019.

[6] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020.

[7] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. Node-level differentially private graph neural networks. *arXiv preprint arXiv:2111.15521*, 2021.

[8] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 890–897, 2019.

[9] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, page 76–85, New York, NY, USA, 2020. Association for Computing Machinery.

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

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

[12] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[13] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Benjamin Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

[14] Nikolaos Gkalelis, Andreas Goulas, Damianos Galanopoulos, and Vasileios Mezaris. 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*, pages 3375–3383, 2021.

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

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

[17] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.

[18] Michael Hay, Chao Li, Gerome Miklau, and David Jensen. Accurate estimation of the degree distribution of private networks. In *2009 Ninth IEEE International Conference on Data Mining*, pages 169–178. IEEE, 2009.

[19] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[20] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*, 2021.

[21] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Communication-Efficient triangle counting under local differential privacy. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[22] Matthew Jagielski, Jonathan R. Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? In *Proceedings of the Advances in Neural Information Processing (NeurIPS)*, Virtual Event, December 2020.

[23] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, Santa Clara, CA, August 2019. USENIX Association.

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

[25] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

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

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

[28] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

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

[30] Alireza Mohammadshahi and James Henderson. Graph-to-graph transformer for transition-based dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3278–3289, Online, November 2020. Association for Computational Linguistics.

[31] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. 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, May 2021.

[32] Iyiola E Olatunji, Thorben Funke, and Megha Khosla. Releasing graph neural networks with differential privacy guarantees. *arXiv preprint arXiv:2109.08907*, 2021.

[33] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 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)*, pages 11–20. IEEE, 2021.

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

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[37] Sofya Raskhodnikova and Adam Smith. Differentially private analysis of graphs. *Encyclopedia of Algorithms*, 2016.

[38] Sina Sajadmanesh and Daniel Gatica-Perez. Locally private graph neural networks. In *Proceedings of the*

2021 ACM SIGSAC Conference on Computer and Communications Security, pages 2130–2145, 2021.

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

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

[41] Petar Veli č kovi ć, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[42] Jianian Wang, Sheng Zhang, Yanghua Xiao, and Rui Song. A review on graph neural network methods in financial applications. *arXiv preprint arXiv:2111.15367*, 2021.

[43] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled renyi differential privacy and analytical moments accountant. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1226–1235. PMLR, 16–18 Apr 2019.

[44] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. Linkteller: Recovering private edges from graph neural networks via influence analysis. *arXiv preprint arXiv:2108.06504*, 2021.

[45] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[46] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[48] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5453–5462, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

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

[50] Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[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. Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*, 2021.

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

[53] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge & Data Engineering*, 34(01):249–270, jan 2022.

[54] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. Graphmi: Extracting private graph data from graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 3749–3755. International Joint Conferences on Artificial Intelligence Organization, 8 2021.

[55] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. Inference attacks against graph neural networks. In *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, August 2022. USENIX Association.

[56] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

A Deferred Theoretical Arguments

A.1 Proof of Theorem 1

To prove Theorem 1, we first establish the following lemma.

Lemma 1. *Let* $\textsc{Agg}(\mathbf{X},\mathbf{A}) = \mathbf{A}^T \cdot \mathbf{X}$ *be the summation aggregation function. Assume that the input feature matrix* $\mathbf{X}$ *is row-normalized, such that* $\forall v \in \mathcal{V} : \|\mathbf{X}_v\|_2 = 1$. *Then, the edge-level sensitivity of the aggregation function is* $\Delta_{\textsc{Agg}} = 1$.

Proof. Let $\mathbf{A}$ and $\mathbf{A}'$ be the adjacency matrices of two arbitrary edge-level adjacent graphs. Therefore, there exist two nodes u and v such that:

$$\begin{cases} \mathbf{A}'_{i,j} \neq \mathbf{A}_{i,j}, & \text{if } i = u \text{ and } j = v, \\ \mathbf{A}'_{i,j} = \mathbf{A}_{i,j}, & \text{otherwise.} \end{cases} \tag{13}$$

Without loss of generality, we can assume that $\mathbf{A}_{v,u} = 1$ and $\mathbf{A}'_{v,u} = 0$. The goal is to bound the following quantity:

$$\|\textsc{Agg}(\mathbf{X},\mathbf{A}) - \textsc{Agg}(\mathbf{X},\mathbf{A}')\|_F.$$

Let $\mathbf{M} = \textsc{Agg}(\mathbf{X},\mathbf{A})$ be the aggregation function output on $\mathbf{A}$, and

$$\mathbf{M}_i = \sum_{j=1}^{N} \mathbf{A}_{j,i}\mathbf{X}_j,$$

be the i-th row of $\mathbf{M}$ corresponding to the aggregated vector for the i-th node. Analogously, let $\mathbf{M}' = \textsc{Agg}(\mathbf{X},\mathbf{A}')$. Then:

$$\begin{aligned} \|\textsc{Agg}(\mathbf{X},\mathbf{A}) - \textsc{Agg}(\mathbf{X},\mathbf{A}')\|_F &= \|\mathbf{M} - \mathbf{M}'\|_F \\ &= (\sum_{i=1}^{N} \|\mathbf{M}_i - \mathbf{M}'_i\|_2^2)^{1/2} \\ &= \left(\sum_{i=1}^{N} \|\sum_{j=1}^{N} (\mathbf{A}_{j,i}\mathbf{X}_j - \mathbf{A}'_{j,i}\mathbf{X}_j)\|_2^2\right)^{1/2} \\ &= \left(\|\mathbf{A}_{v,u}\mathbf{X}_v - \mathbf{A}'_{v,u}\mathbf{X}_v\|_2^2\right)^{1/2} \\ &= \|(\mathbf{A}_{v,u} - \mathbf{A}'_{v,u})\mathbf{X}_v\|_2 \\ &= \|\mathbf{X}_v\|_2 \\ &= 1, \end{aligned}$$

which concludes the proof. □

We can now prove Theorem 1.

Proof. The PMA mechanism applies the Gaussian mechanism on the output of the summation aggregation function $\textsc{Agg}(\mathbf{X},\mathbf{A}) = \mathbf{A}^T \cdot \mathbf{X}$. Based on Lemma 1, the edge-level sensitivity of $\textsc{Agg}(\cdot)$ is 1. Therefore, according to Corollary 3 of [29], each individual application of the Gaussian mechanism is $(\alpha, \alpha/2\sigma^2)$-RDP. As PMA can be seen as an adaptive composition of K such mechanisms, based on Proposition 1 of [29], the total privacy cost is $(\alpha, K\alpha/2\sigma^2)$-RDP. □

A.2 Proof of Proposition 2

Proof. Under edge-level DP, only the adjacency information is protected. In Algorithm 2, the only step where the graph's adjacency is used is the application of the PMA mechanism (step 4), which according to Theorem 1 is $(\alpha, K\alpha/2\sigma^2)$-RDP. Since EM does not use the graph's edges and the classification module only post-process the private aggregated features without accessing the edges again, the total privacy cost remains $(\alpha, K\alpha/2\sigma^2)$-RDP. Therefore, according to Proposition 1 it 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 $\epsilon = \frac{K}{2\sigma^2} + \sqrt{2K\log(1/\delta)}/\sigma$. □

A.3 Proof of Theorem 2

We first prove Lemma 2 and Lemma 3, and then prove Theorem 2.

Lemma 2. *Given any graph* $\mathcal{G} = (\mathcal{V},\mathcal{E},\mathbf{X})$, *let*

$$\mathit{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) = \sum_{u \in \mathfrak{N}_v} \mathbf{X}_u$$

be the summation aggregation function over the neighborhood $\mathfrak{N}_v$ *of any arbitrary node* $v \in \mathcal{V}$. *Assume that the input feature matrix* $\mathbf{X}$ *is row-normalized, such that* $\forall v \in \mathcal{V} : \|\mathbf{X}_v\|_2 = 1$. *Then, the node-level sensitivity of* $\mathit{agg}(.)$ *is* $\Delta_{\mathit{agg}} = 1$.

Proof. Consider a node-level adjacent graph $\mathcal{G}' = (\mathcal{V}',\mathcal{E}',\mathbf{X}')$ formed by adding a single node q to $\mathcal{G}$. Hence, we have $\mathcal{V}' = \mathcal{V} \cup \{q\}$, and $\mathbf{X}'_v = \mathbf{X}_v$ for every node $v \in \mathcal{V}$. Let $\mathbf{A}$ and $\mathbf{A}'$ be the adjacency matrices of $\mathcal{G}$ and $\mathcal{G}'$ respectively. The goal is to bound the following:

$$\|\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) - \mathsf{agg}\left(\{\mathbf{X}'_u : \forall u \in \mathfrak{N}'_v\}\right)\|_2 \leq 1. \tag{14}$$

where $\mathfrak{N}_v = \{u : \mathbf{A}_{u,v} = 1\}$ and $\mathfrak{N}'_v = \{u : \mathbf{A}'_{u,v} = 1\}$ are the adjacent nodes to v in $\mathcal{G}$ and $\mathcal{G}'$, respectively. Fixing any arbitrary node $v \in \mathcal{V}$, we have the following two cases:

1. If $q \in \mathfrak{N}'_v$, then we have $\mathfrak{N}_v = \mathfrak{N}'_v \setminus \{q\}$. Therefore:

$$\begin{aligned} &\|\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) - \mathsf{agg}\left(\{\mathbf{X}'_u : \forall u \in \mathfrak{N}'_v\}\right)\|_2 \\ &\qquad = \|\sum_{u \in \mathfrak{N}_v} \mathbf{X}_u - \sum_{u \in \mathfrak{N}'_v} \mathbf{X}'_u\|_2 \\ &\qquad = \|\mathbf{X}_q\|_2 = 1. \end{aligned}$$

2. If $q \notin \mathfrak{N}'_v$, then we have $\mathfrak{N}_v = \mathfrak{N}'_v$. Therefore:

$$\begin{aligned} &\|\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) - \mathsf{agg}\left(\{\mathbf{X}'_u : \forall u \in \mathfrak{N}'_v\}\right)\|_2 \\ &\qquad = \|\sum_{u \in \mathfrak{N}_v} \mathbf{X}_u - \sum_{u \in \mathfrak{N}'_v} \mathbf{X}'_u\|_2 = 0. \end{aligned}$$

Eq. 14 follows from the above two cases. □

Lemma 3. *Given any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with adjacency matrix $\mathbf{A}$ and maximum degree bounded above by some constant $D > 0$, assume that the feature matrix $\mathbf{X}$ is row-normalized, such that $\forall v \in \mathcal{V} : \|\mathbf{X}_v\|_2 = 1$. Let $\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) = \sum_{u \in \mathfrak{N}_v} \mathbf{X}_u$ be the summation aggregation function over the neighborhood $\mathfrak{N}_v$ of any arbitrary node $v \in \mathcal{V}$, and $\widetilde{A_{GG}}(\mathbf{X}, \mathbf{A})$ be a noisy aggregation mechanism which applies the Gaussian mechanism independently on the aggregated vector of every individual node as:*

$$\widetilde{A_{GG}}(\mathbf{X}, \mathbf{A}) = \left[\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\}) + \mathcal{N}(\sigma^2 \mathbb{I}) : \forall v \in \mathcal{V}\right]^T .$$

Then $\widetilde{A_{GG}}(.)$ is $(\alpha, {}^{D\alpha}/_{2\sigma^2})$-RDP.

Proof. According to Lemma 2, the node-level sensitivity of $\mathsf{agg}(\{\mathbf{X}_u : \forall u \in \mathfrak{N}_v\})$ is 1, and thus each individual noisy aggregation query is $(\alpha, {}^{\alpha}/_{2\sigma^2})$-RDP. Although $\widetilde{\text{AGG}}$ is composed of $N = |\mathcal{V}|$ such queries in total (one noisy aggregation per node), as $\mathcal{G}$'s maximum degree is bounded above by D, the embedding $\mathbf{X}_u$ of each node u only contributes to maximum D out of N queries. As these N queries are chosen non-adaptively and the noise of the Gaussian mechanism is independently drawn for each query, the maximum privacy cost of $\widetilde{\text{AGG}}(.)$ is equivalent to D compositions of $(\alpha, {}^{\alpha}/_{2\sigma^2})$-RDP mechanisms, which based on Proposition 1 of [29] is $(D\alpha, {}^{\alpha}/_{2\sigma^2})$-RDP. □

Now, we prove Theorem 2.

Proof. At each step of the PMA mechanism, the Gaussian mechanism is applied on every output row of the summation aggregation function $\text{AGG}(\mathbf{X}, \mathbf{A}) = \mathbf{A}^T \cdot \mathbf{X}$. Based on Lemma 3, this mechanism is $(\alpha, {}^{\alpha D}/_{2\sigma^2})$-RDP. As PMA can be seen as an adaptive composition of K such mechanisms, based on Proposition 1 of [29], the total privacy cost is $(\alpha, {}^{\alpha DK}/_{2\sigma^2})$-RDP. □

A.4 Proof of Proposition 3

Proof. Under node-level DP, all the information pertaining to an individual node, including its features, label, and edges, are private. The first step of Algorithm 2 privately processes the node features and labels so as to satisfy $(\alpha, \epsilon_1(\alpha))$-RDP. Steps 2 and 3 of the algorithm, however, expose the private node features, but then they are processed by steps 4 and 5, which are $(\alpha, {}^{DK\alpha}/_{2\sigma^2})$-RDP (according to Theorem 2) and $(\alpha, \epsilon_5(\alpha))$-RDP, respectively. As a result, Algorithm 2 can be seen as an adaptive composition of an $(\alpha, \epsilon_1(\alpha))$-RDP mechanism, an $(\alpha, {}^{DK\alpha}/_{2\sigma^2})$-RDP mechanism, and an $(\alpha, \epsilon_5(\alpha))$-RDP mechanism. Therefore, based on Proposition 1 of [29], the total node-level privacy cost of Algorithm 2 is $(\alpha, \epsilon_1(\alpha) + {}^{DK\alpha}/_{2\sigma^2} + \epsilon_5(\alpha))$-RDP, which ensures $(\epsilon_1(\alpha) + \epsilon_5(\alpha) + \frac{DK\alpha}{2\sigma^2} + \frac{\log(1/\delta)}{\alpha - 1}, \delta)$-DP based on Proposition 1. □