When Differential Privacy Meets Graph Neural Networks

Sina Sajadmanesh
Idiap Research Institute
EPFL
sajadmanesh@idiap.ch

Daniel Gatica-Perez
Idiap Research Institute
EPFL
gatica@idiap.ch

Abstract

Graph Neural Networks have demonstrated superior performance in learning graph representations for several subsequent downstream inference tasks. However, learning over graph data types can raise privacy concerns when nodes represent people or human-related variables that involve personal information about individuals. Previous works have presented various techniques for privacy-preserving deep learning over non-relational data, such as image, audio, video, and text, but there is less work addressing the privacy issues involved in applying deep learning algorithms on graphs. As a result and for the first time, in this paper, we develop a privacy-preserving learning algorithm with formal privacy guarantees for *Graph Convolutional Networks (GCNs)* based on *Local Differential Privacy (LDP)* to tackle the problem of node-level privacy, where graph nodes have potentially sensitive features that need to be kept private, but they could be beneficial for learning rich node representations in a centralized learning setting. Specifically, we propose an LDP algorithm in which a central server can communicate with graph nodes to privately collect their data and estimate the graph convolution layer of a GCN. We then analyze the theoretical characteristics of the method and compare it with state-of-the-art mechanisms. Experimental results over real-world graph datasets demonstrate the effectiveness of the proposed method for both privacy-preserving node classification and link prediction tasks and verify our theoretical findings.

1 Introduction

Following the advances of deep neural networks in various machine learning domains, such as computer vision, natural language understanding, and speech processing, in the past few years, extending deep learning models for graph-structured data has attracted growing interest, leading to the advent of Graph Neural Networks (GNNs) [35]. These models have shown superior performance on a wide range of applications in social sciences [15], biology [31], molecular chemistry [7], and so on, achieving state-of-the-art results in various graph-based tasks, such as node classification [48], link prediction [47], and community detection [3]. However, most real-world graphs associated with people or human-related activities, such as social and economic networks, are often sensitive and contain personal information about individuals. For example in a social network, a user's friend list, profile information, likes and comments, etc., could potentially be private with different levels of privacy protection. To satisfy users' privacy expectations in accordance with the recent data protection policies, such as General Data Protection Regulations (GDPR), it is of great importance to develop privacy-preserving GNN models for applications that rely on graphs accessing users' personal data.

Problem and motivation. In light if these privacy considerations and (to our knowledge) for the first time, in this paper, we define the problem of node-level privacy, where graph nodes have potentially

Preprint. Under review.

sensitive attributes that are kept private, but the rest of the graph is observable from the viewpoint of a central server, who wishes to benefit from private node attributes to learn a GNN over the graph. The ability to learn graph representation from private node features can be very useful in many concrete applications in social network analysis and mobile computing. For example, social smartphone apps, such as social networks, messaging platforms, and dating apps, could potentially benefit from users' personal information, such as their phone's sensor data or list of installed apps, to learn better node representations which can eventually empower their recommendation algorithms and improve their user experience. However, without any means of user data protection, this implies that the application server needs to get access to and collect personal raw data, which can raise concerns with regards to privacy as the collected data can be used for other, unauthorized purposes.

Challenges. Training a GNN under node-level privacy setting is a challenging issue due to the relational characteristics of graphs. Unlike other deep learning models wherein the training data are independent, in the case of GNNs, the samples – nodes of the graph – are connected to each other via links and exchange information through a message-passing framework during the training of a GNN [16]. This fact makes employing conventional privacy-preserving machine learning paradigms, such as federated and split learning [27, 40], infeasible due to the excessive communication overhead that they impose because of the relationships between graph nodes.

Contributions. In this paper, we propose a novel privacy-preserving GNN learning framework that combines the power of Graph Convolutional Networks (GCNs) [24] and Local Differential Privacy (LDP) [21] to protect sensitive node attributes from being exposed to an honest-but-curious server [30], who requires these features for training its own GNN (Section 3). By extending LDP mechanisms, the server can efficiently communicate with the graph nodes to privately collect their features and estimate the first-layer graph convolution of the GNN (Section 4). Our solution is optimized for more communication efficiency, and we show that it works best when the distribution of features is skewed. We derive the theoretical properties of the proposed algorithm, such as its formal privacy guarantee and error bound (Section 4). Finally, we conduct comprehensive experiments over several real-world graph datasets to verify our theoretical findings and demonstrate the effectiveness of our framework in privacy-preserving node classification and link prediction tasks (Section 5). To the best of our knowledge, this paper defines the problem of GNN training with node-level privacy for the first time, studying the integration of differential privacy with graph neural networks.

2 Related Work

Graph Neural Networks. Recent years have seen a surge in applying GNNs for representation learning over graphs. A GNN consists of multiple layers, wherein each layer, the representation of a node is obtained by aggregating the previous-layer embeddings of the node and its neighbors (the neighborhood aggregation or message passing step [16]), followed by a neural network transformation. Based on this idea, numerous GNN models have been proposed, including Graph Convolutional Networks (GCN) [24], Graph Attention Networks (GAT) [39], GraphSAGE [15], Graph Isomorphism Networks (GIN) [44], Gated Graph Neural Networks [28], and so forth. We refer the reader to the available surveys on GNNs [43, 49] for other models and their internal architecture. While there is a growing number of GNN models designed to solve different tasks in various domains, we are not aware of prior work on training GNNs in the private setting.

Privacy-Preserving Machine Learning. Numerous techniques based on Homomorphic Encryption [34], Secure Multi-Party Computation (SMC) [12], Federated Learning [27], Split Learning [40], and Differential Privacy (DP) [8] have been proposed to address the privacy issue when the data is sensitive and cannot be released to untrusted third-parties for model training. Homomorphic encryption allows the training and inference of a model to be performed directly on encrypted data, and has been successfully applied to deep learning models [17, 18]. SMC enables two or more parties, who do not trust each other, to jointly train a model without exposing their input data to each other. It has been used for linear regression, logistic regression, and neural networks [29, 32, 1]. Federated and split learning algorithms allow users to keep their private data and communicate with the service provider in order to train a model, collaboratively [25, 2]. Finally, differential privacy provides a mathematical framework for computing statistical queries over private databases, and has been successfully adapted in deep learning systems to preserve training data privacy [37, 42].

However, these methodologies are mostly designed for non-relational data, such as images and text. We are not aware of any prior work extending any of these techniques for GNNs.

3 Preliminaries

Problem definition. We now formally define the problem of learning GNNs under the node-level privacy setting. Assume that a server has access to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose nodes and links are visible by the server, but the feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$, comprising m-dimensional feature vectors $\mathbf{x}_v$ for each $v \in \mathcal{V}$, is private to the nodes and thus not directly accessible by the server. The question is, how can the server collaborate with the nodes to use private features and learn a GNN over $\mathcal{G}$?

As our solution is based on graph convolutional networks and local differential privacy, in the following we present the required fundamental background about these two concepts.

Graph Convolutional Networks. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times m}$, a K-layer Graph Convolutional Network (GCN) [24] consists of K graph convolution layers , where the embedding of a node $v \in \mathcal{V}$ at layer k is generated by aggregating the embeddings of the node's neighbors from the previous layer using the GCN graph convolution function, defined by:

$$GC^k(v) = \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|\mathcal{N}(u)|\,|\mathcal{N}(v)|}} \tag{1}$$

where $\mathcal{N}(v)$ is the set of neighbors of node v including itself, and $\mathbf{h}_u^{k-1}$ is the embedding of node u at layer $k-1$. Also, $\mathbf{h}_v^0 = \mathbf{x_v}$, i.e. the initial representation of node v is equal to its original feature vector $\mathbf{x}_v$. The k-th layer embedding of v is then generated by $\mathbf{h}_v^k = \sigma(\mathbf{W}^k \cdot GC^k(v))$ where $\mathbf{W}^k$ is the trainable weight matrix for layer k.

The advantage of the GCN model, from a privacy-preserving perspective, is that the only point where the node features are touched is inside the graph convolution function of the very first layer. Therefore, unlike other GNN models, such as GraphSAGE or GAT, where individual node features are also used along with the aggregation of the neighboring features, GCN only requires aggregated node features – the output of the graph convolution function (1) in the first layer. This allows to privately calculate this aggregation using local differential privacy.

Local Differential Privacy. Local differential privacy (LDP) [21] is an increasingly used approach for collecting private data and computing statistical queries, such as mean, count, and histogram, and is already deployed by Google [11], Apple [38], and Microsoft [5], for private data analytics at scale. The key idea behind LDP is that data holders do not need to share their private data with a data aggregator, but instead send a perturbed version of their data, which is meaningless individually, but can approximate the target query when aggregated. It basically includes two steps: (i) data collection, in which each data holder perturbs its data using a special randomized function $\mathcal{F}$ and sends the output to the aggregator; and (ii) estimation, in which, the aggregator combines all the received perturbed values and estimates the target query. To prevent the aggregator from inferring the original private value from the perturbed one, the function $\mathcal{F}$ must satisfy the following definition [21]:

Definition 1. *Given $\epsilon > 0$, a randomized function $\mathcal{F}$ satisfies ϵ-local differential privacy if for all possible pairs of user's private data x and x', and for all possible outputs $y \in Range(\mathcal{F})$, we have:*

$$\Pr[\mathcal{F}(x) = y] \leq e^{\epsilon} \Pr[\mathcal{F}(x') = y] \tag{2}$$

The parameter ϵ in the above definition is called the *"privacy budget"* and is used to tune the utility-privacy trade-off: a smaller ϵ leads to stronger privacy guarantees, but lower utility (and vice versa). Based on the above definition, the function $\mathcal{F}$ should assign similar probabilities (controlled by ϵ) to the outputs of different input values x and x', so that by looking at the outputs, an adversary could not infer the input value with high probability, regardless of any side knowledge he might have.

4 Proposed Method

Now, we describe our proposed framework for training a GNN using private node features. The key idea of our method is to approximate the first-layer graph convolution for any node $v \in \mathcal{V}$, as

an aggregated statistic over the private features of v itself and its neighbors, using local differential privacy (LDP). To this end, our approach requires the first layer of the GNN used by the server (as the point of contact with the features) to be the GCN convolution layer defined by (1), but there is no restriction on the choice of subsequent layers (they can be GAT, GraphSAGE, etc). As a result, our framework includes two steps: (i) collecting the node features by the server through a differentially private approach, and (ii) approximating the graph convolution in the first layer using the collected features and proceeding with the training of the GNN.

There are several LDP mechanisms in the literature that are used for calculating mean value over private numeric data that can be adapted for our problem. The most well-known algorithms include Laplace Mechanism [9], Duchi et al.'s Mechanism [6], and Piecewise Mechanism [41]. Here, we extend the method of Duchi for the aim of estimating the graph convolution function, which is a more complex statistic compared to the simple mean. Therefore, we call our LDP mechanism *Private Graph Convolution (PGC)*. Inspired by the work of Ding et al. [5], our mechanism is tuned for reduced communication cost between the server and the graph nodes. We derive the theoretical properties of our PGC method and discuss its differences with other mechanisms. Additional details about Laplace and Piecewise mechanisms are provided in Appendix A.

Collection of node features. Assume that each node v owns a private m-dimensional feature vector $\mathbf{x}_v = [\mathbf{x}_v^{(1)}, \mathbf{x}_v^{(1)}, \ldots, \mathbf{x}_v^{(m)}]^T$ where $\mathbf{x}_v^{(i)} \in [\alpha^{(i)}, \beta^{(i)}]$. When the feature vector of node v is needed, the server sends the parameter ϵ to v. After, the node sends to the server a m-dimensional perturbed vector $\mathbf{y}_v$ where each element $y_v^{(i)}$ is drawn independently from the following Bernoulli distribution:

$$\Pr[y_v^{(i)} \mid x_v^{(i)}] = \left(\frac{1}{e^\epsilon+1} + \frac{x_v^{(i)} - \alpha^{(i)}}{\Delta^{(i)}} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}\right)^{y_v^{(i)}} \left(\frac{e^\epsilon}{e^\epsilon+1} + \frac{x_v^{(i)} - \alpha^{(i)}}{\Delta^{(i)}} \cdot \frac{1 - e^\epsilon}{e^\epsilon + 1}\right)^{1-y_v^{(i)}} \tag{3}$$

where $\Delta^{(i)} = \beta^{(i)} - \alpha^{(i)}$, and ϵ is the privacy budget parameter provided by the server. The value of $y_v^{(i)}$ is recorded by the node to be returned in subsequent calls so that the server cannot guess the node's private feature by issuing repeated queries.

Approximation of graph convolution. Upon collecting the perturbed data, the server can estimate the first-layer graph convolution of any node v by:

$$\widehat{GC}(v) = \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{y}_v^*}{\sqrt{d_u d_v}} \tag{4}$$

where $d_v = |\mathcal{N}(v)|$ is the degree of node v and $\mathbf{y}_v^*$ is the unbiased form of $\mathbf{y}_v$, defined as:

$$\mathbf{y}_v^* = \frac{(e^\epsilon + 1)\mathbf{y}_v - 1}{e^\epsilon - 1} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \tag{5}$$

Here, $\boldsymbol{\alpha} = [\alpha_i]_{i=1:m}$, $\boldsymbol{\Delta} = [\Delta_i]_{i=1:m}$, and all vector multiplications and divisions are performed element-wise, thus the dot notation is abused for element-wise multiplication. Note that we can also calculate $\mathbf{y}_v^*$ at the node-side and send it to the server instead of $\mathbf{y}_v$. However, the advantage of performing this operation at the server-side is that $\mathbf{y}_v$ in this case will be binary-valued, which is much more efficient to send in terms of communication cost than the real-valued vector $\mathbf{y}_v^*$. This is also the advantage of PGC compared to the Laplace or Piecewise mechanisms, as the output of these two methods is continuous, which imposes more communication overhead compared to ours.

After this step, the server can proceed with the rest of the GNN layers to complete the training. The pseudo-code of the forward propagation of the proposed framework as an adaptation of the general GNN forward propagation [16] is presented in Algorithm 1.

4.1 Theoretical Analysis

In this section, we discuss theoretical properties of our method. The proof of all the lemmas presented here can be found in Appendix C.

Lemma 1. *Algorithm 1 satisfies ϵ-local differential privacy for individual node features.*

Lemma 2. *The graph convolution estimator $\widehat{GC}(v)$ defined by (4) is an unbiased estimator for the GCN graph convolution defined by (1).*

Algorithm 1: Forward propagation for a K-Layer GNN as an adaptation of [16]

Input : Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [2, K]\}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

Server sends $\epsilon, \boldsymbol{\alpha}, \boldsymbol{\Delta}$ to every node $v \in \mathcal{V}$.
Each node $v \in \mathcal{V}$ sends back $\mathbf{y}_v$ obtained using (3) to the server.

Server-side computation:
for $k = 1$ *to* K **do**
  for $v \in \mathcal{V}$ **do**
    if $k == 1$ **then**
      $\mathbf{GC}_v = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \left[\frac{(e^\epsilon + 1)\mathbf{y}_u - 1}{e^\epsilon - 1} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \right]$
      $\mathbf{h}_v^1 = \sigma\left(\mathbf{W}^1 \cdot \mathbf{GC}_v\right)$
    else
      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$
      $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
  end
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K$

Lemma 3. *With probability at least $1-\delta$, for every node v and any arbitrary feature $x_v \in [\alpha, \alpha+\Delta]$, we have:*

$$\left|\widehat{GC}(v) - GC(v)\right| \leq \Delta \cdot \frac{e^\epsilon + 1}{e^\epsilon - 1} \sqrt{\frac{1}{2} \log(\frac{2}{\delta})} \tag{6}$$

Lemma 4. *For any node v and any arbitrary input feature $x_v \in [\alpha, \beta]$ with $\Delta = \beta - \alpha$, the variance of the unbiased PGC mechanism, y_v^*, is calculated by:*

$$Var[y_v^*] = \Delta^2 \frac{e^\epsilon}{(e^\epsilon - 1)^2} + (x_v - \alpha)(\beta - x_v) \tag{7}$$

The variance of an LDP mechanism is the key factor affecting the accuracy of the estimator, as a lower variance will usually lead to a more accurate estimation. The variance of y_v^* as shown by (7) depends on the original feature x_v. It can be easily shown that the variance will reaches its minimum as $x_v \to \alpha$ or $x_v \to \beta$, and gets maximized when $x_v \to \frac{\alpha+\beta}{2}$. Therefore, the *expected variance* of y_v^* is minimum when $\mathbb{E}[x_v]$ is close to either α or β, and this happens when the distribution of the features is skewed toward either the beginning or the end of their domain $[\alpha, \beta]$ (e.g. Exponential distribution).

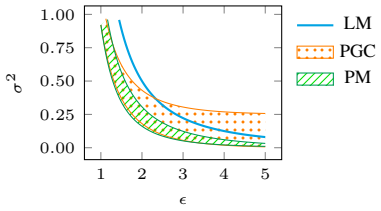


Figure 1: Variance of different LDP mechanisms with $\alpha = 0, \beta = 1$.

Figure 1 compares the minimum and maximum variance of PGC with those of Laplace mechanism (LM) and Piecewise mechanism (PM) for an arbitrary feature x with $\alpha = 0, \beta = 1$, and $\epsilon \in [1, 5]$. The dotted orange area represents the region where the variance of PGC changes between its minimum and maximum, and the green hatched area corresponds to PM. LM is shown by a single blue line, as its variance does not depend on x. According to the figure, while the worst-case variance of PM is always lower than PGC and LM, this is not necessarily true in the expected case (as we will see in our experiments), especially when the feature distribution is skewed. This is because as opposed to PGC, the variance of PM is maximized at the two sides of the feature range and minimized in the middle [41], thus it works better if the the feature distribution is symmetric on $[\alpha, \beta]$ (e.g. Gaussian). A more detailed comparison (with LM and PM variance formulas) is available in Appendix B.

Table 1: Descriptive statistics of the real-world datasets

DATASET	#CLASSES	#NODES	#EDGES	# NODE FEATURES	AVG. DEGREE	TRAIN/VAL/TEST*
CITESEER	6	3,327	4,552	3,703	2.74	4/15/30%†
CORA	7	2,708	5,278	1,433	3.90	5/17/37%†
ELLIPTIC	2	203,769	234,355	166	2.30	11.4/5.7/5.7%†
FLICKER	7	89,250	449,878	500	10.08	50/25/25%
TWITCH	2	7,126	35,324	2,545	9.91	50/25/25%

* Corresponds to node classification. For link prediction, all datasets use 85/5/10% splits.
† The splits do not sum to 100% as not all the nodes are labeled.

5 Experiments

We conduct extensive experiments to assess the privacy-utility trade-off of the proposed framework and evaluate its performance in two classical tasks: node classification and link prediction.

Datasets and baselines. In order to test the performance of the proposed method and other baselines, we use five real-world datasets with different characteristics, whose description is summarized in Table 1. Additional details about the datasets are provided in Appendix D. Alongside our PGC method, we also use the Laplace mechanism (LM) and the Piecewise mechanism (PM) in the proposed framework for error analysis and predictive performance comparison. Additionally, to see how the performance of different LDP methods compares to the optimal case, we use standard GCN with original features without any privacy protection, denoted as RAW, whose result can be considered as an upper-bound on the privacy-preserving methods.

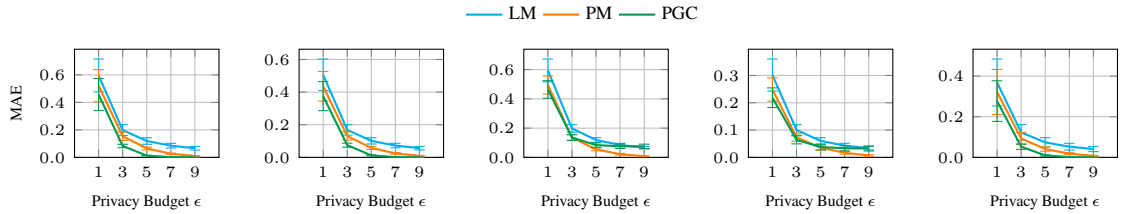
Experiment setup. For the node classification task, we use standard training/validation/test splits for the citation networks as well as Flickr [45, 46]. In other datasets, we randomly split labeled nodes with 50/25/25% ratios for training, validation, and test sets, respectively. For link prediction, all the datasets are randomly split with 85/5/10% ratios. We normalize the node features of all the datasets between zero and one, so in all cases, we have $\alpha = 0$ and $\Delta = 1$. We use the two-layer GCN as in [24] for the node classification task. For link prediction, we use Variational Graph Auto-Encoder [23] with a two-layer GCN as the encoder, with batch-normalization applied after the first convolutional layer. In all cases, we use ReLU activation function for all layers and fix the hidden and output dimensions of the model to 32 and 16, respectively. We optimize hyper-parameters for RAW method on the validation set over initial learning rate, weight decay, and dropout rate, and use the same values for all the other methods. All models are trained using Adam optimizer [22] over a maximum of 500 epochs. For node classification, we train the models for a minimum of 10 epochs and stop the training if the validation loss does not decrease over 20 consecutive epochs. In the case of link prediction, we set the minimum number of epochs to 100, and check the validation loss every 10 epochs. We stop the training if the validation loss does not decrease over 10 consecutive checks. Finally, we measure the performance on the test set over 10 runs and report the average and standard deviation of the results. Implementation is done using PyTorch Geometric [14] and PyTorch Lightning [13].

5.1 Results

Analysis of estimation error. In the first set of experiments, we set to empirically measure the mean absolute error (MAE) in the estimation of the graph convolution for our PGC mechanism and compare it with the one generated by LM and PM. We calculate the MAE for any node v as:

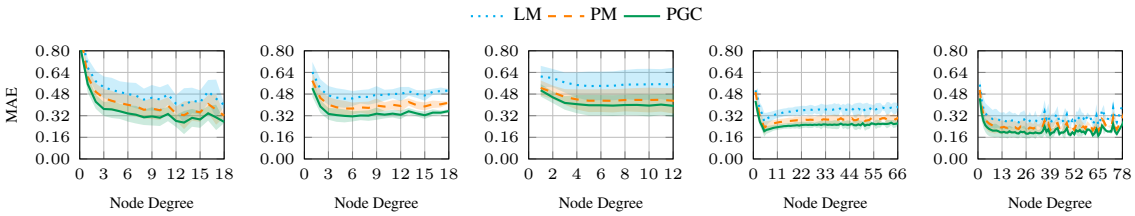
$$MAE(v) = \frac{1}{m}\sum_{i=1}^{m}\left|GC(v)^{(i)} - \widehat{GC}(v)^{(i)}\right| \tag{8}$$

First, we look at how different LDP mechanisms perform in terms of the estimation error with respect to different values of the privacy budget ϵ. We vary the value of ϵ within $\{1, 3, \ldots, 9\}$, and average the MAE over all the nodes of the graph. The results for different datasets are illustrated in Figure 2. As expected, by increasing the value of ϵ, the MAE of all methods decreases exponentially and converges toward zero. We can see that the error of the PGC mechanism is consistently lower than other mechanisms across three out of five datasets, namely Cora, CiteSeer, and Twitch. But



(a) CiteSeer (b) Cora (c) Elliptic (d) Flickr (e) Twitch

Figure 2: Effect of the privacy budget ϵ on the mean absolute error of the graph convolution estimation.



(a) CiteSeer (b) Cora (c) Elliptic (d) Flickr (e) Twitch

Figure 3: Mean absolute error of the graph convolution estimation w.r.t node degree with $\epsilon = 1$.

in the other two, Elliptic and Flickr, it falls behind PM for higher values of ϵ. This is because the features in the first three mentioned datasets are rather sparse, so the feature distribution is skewed toward zero. But in the other two datasets, the average feature value is closer to 0.5, resulting in lower error for PM. However, even in this case, when ϵ is small, PGC still results in a more accurate estimation than PM.

Next, we assess how the estimation error varies on average as a function of node degree. To this end, we set $\epsilon = 1$, and average the MAE over all the nodes v with the same degree d_v and plot the result for $d_v \in [1, d_{max}]$, where d_{max} is the 99% quantile of the degree distribution that we choose as a cut-off. The result is depicted in Figure 3. According to the figure, the estimation error drops first as the node degree increases but quickly converges to a constant value. This result is consistent with Lemma 3, verifying that the error is not asymptotically dependent on the node degree.

Analysis of predictive performance. Finally, we evaluate how our proposed framework performs in learning rich node representations for node classification and link prediction tasks under the privacy requirements. For this experiment, we selected the value of ϵ from $\{1, 5, 9\}$. The performance of different methods is measured in terms of micro averaged F1 score for node classification and the area under ROC curve (AUC) for link prediction. Table 2 presents the experimental results of different methods in both node classification and link prediction tasks. For illustrative purposes, for each value of ϵ, The closest number to the result of the RAW method (the optimal result) is bold-faced. Similar to Figure 2, we see that the performance of different methods (except RAW obviously) rise as we increase ϵ. The table shows that our proposed PGC method is superior to PM and LM most of the times in both node classification and link prediction across all datasets and for different values of ϵ. The only exception here is the Elliptic dataset, for which the PM method outperforms PGC. This is analogous to the error estimation result in Figure 2. As mentioned previously for that figure, the average feature value in the Elliptic dataset is higher than the other datasets, which eventually results in lower variance, lower estimation error, and finally higher accuracy for PM.

5.2 Discussion on choosing parameters α, Δ, and ϵ

In addition to the GNN model, our framework has also its own hyper-parameters $\boldsymbol{\alpha}$, $\boldsymbol{\Delta}$, and ϵ, which are sent to the graph nodes by the server. The proposed framework requires that the server knows the range of private features, indicated by $\boldsymbol{\alpha}$ and $\boldsymbol{\Delta}$, which is a common assumption in the literature [6].

Table 2: Performance of different methods in link prediction and node classification tasks.

		Link Prediction (AUC %)			Node Classification (Micro-F1 %)		
Dataset	Method	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 9$	$\epsilon = 1$	$\epsilon = 5$	$\epsilon = 9$
Citeseer	LM	$\mathbf{77.5 \pm 0.8}$	81.1 ± 1.0	85.9 ± 0.6	36.5 ± 1.7	57.2 ± 2.0	68.8 ± 1.1
	PM	77.3 ± 1.3	84.8 ± 1.6	90.9 ± 0.7	37.0 ± 1.6	67.0 ± 1.5	$\mathbf{71.0 \pm 0.7}$
	PGC	$\mathbf{77.5 \pm 1.2}$	$\mathbf{89.3 \pm 0.9}$	$\mathbf{91.5 \pm 0.4}$	$\mathbf{38.1 \pm 1.5}$	$\mathbf{71.1 \pm 1.0}$	70.1 ± 1.1
	Raw	91.8 ± 0.7	91.8 ± 0.7	91.8 ± 0.7	70.1 ± 0.6	70.1 ± 0.6	70.1 ± 0.6
Cora	LM	82.1 ± 1.4	87.4 ± 1.0	89.1 ± 0.7	53.9 ± 1.6	73.0 ± 1.5	78.2 ± 0.5
	PM	82.0 ± 1.1	89.1 ± 1.0	92.2 ± 0.6	$\mathbf{55.1 \pm 2.4}$	77.5 ± 0.8	80.9 ± 0.7
	PGC	$\mathbf{83.0 \pm 0.8}$	$\mathbf{91.8 \pm 0.5}$	$\mathbf{93.0 \pm 0.7}$	$\mathbf{55.1 \pm 2.3}$	$\mathbf{80.6 \pm 0.6}$	$\mathbf{81.4 \pm 0.5}$
	Raw	93.3 ± 0.5	93.3 ± 0.5	93.3 ± 0.5	81.4 ± 0.5	81.4 ± 0.5	81.4 ± 0.5
Elliptic	LM	54.7 ± 0.3	70.2 ± 1.1	76.1 ± 1.3	90.6 ± 0.1	91.3 ± 0.2	92.6 ± 0.3
	PM	$\mathbf{57.0 \pm 0.6}$	$\mathbf{75.3 \pm 1.3}$	$\mathbf{79.7 \pm 0.9}$	$\mathbf{90.6 \pm 0.0}$	$\mathbf{92.8 \pm 0.3}$	$\mathbf{94.3 \pm 0.3}$
	PGC	$\mathbf{57.0 \pm 0.3}$	71.8 ± 0.8	72.2 ± 0.8	$\mathbf{90.6 \pm 0.0}$	91.9 ± 0.3	92.3 ± 0.2
	Raw	80.1 ± 1.2	80.1 ± 1.2	80.1 ± 1.2	95.6 ± 0.4	95.6 ± 0.4	95.6 ± 0.4
Flickr	LM	72.3 ± 1.3	74.2 ± 1.7	75.1 ± 2.5	$\mathbf{44.2 \pm 0.4}$	47.4 ± 0.4	48.6 ± 0.6
	PM	71.9 ± 0.6	75.3 ± 1.6	$\mathbf{75.9 \pm 2.3}$	44.0 ± 0.6	$\mathbf{48.5 \pm 0.4}$	$\mathbf{49.7 \pm 0.1}$
	PGC	$\mathbf{73.5 \pm 0.8}$	$\mathbf{75.7 \pm 2.2}$	75.2 ± 1.6	$\mathbf{44.2 \pm 0.4}$	$\mathbf{48.5 \pm 0.2}$	48.5 ± 0.3
	Raw	75.5 ± 3.7	75.5 ± 3.7	75.5 ± 3.7	50.0 ± 0.1	50.0 ± 0.1	50.0 ± 0.1
Twitch	LM	80.1 ± 0.3	82.3 ± 0.4	83.0 ± 0.3	57.8 ± 0.5	59.0 ± 0.9	60.1 ± 0.8
	PM	80.6 ± 0.3	83.2 ± 0.4	84.5 ± 0.3	$\mathbf{58.1 \pm 0.6}$	59.1 ± 0.9	61.0 ± 0.6
	PGC	$\mathbf{80.7 \pm 0.4}$	$\mathbf{84.2 \pm 0.6}$	$\mathbf{84.7 \pm 0.4}$	57.9 ± 0.5	$\mathbf{61.0 \pm 0.7}$	$\mathbf{61.3 \pm 0.3}$
	Raw	85.1 ± 0.4	85.1 ± 0.4	85.1 ± 0.4	61.3 ± 0.2	61.3 ± 0.2	61.3 ± 0.2

Technically, the server could either guess the range of the features (e.g. between 00:00 and 23:59 if the feature represents time) or could use other privacy-preserving methods, such as secure multi-party computation [36], as a pre-processing step to determine range parameters. It is important to note that if, for any node of the graph, its requested feature lies outside the bounds provided by the server (e.g. due to inaccurate guessing), the node can simply clip its feature within the given bound.

Probably the most important parameter of the method is the privacy budget ϵ, which controls the trade-off between the accuracy of the GNN and the strength of the privacy guarantee. While there are some efforts trying to systematically determining ϵ in differentially private applications [26, 20], still there is no general guideline for choosing the best value for ϵ, since the privacy guarantees enforced by differential privacy are very different based on the data domain and the target query. Therefore, previous works [11, 5, 38] usually evaluate their systems by trying different values of ϵ, and then pick the smallest one resulting in an acceptable performance for the deployment. For instance, Microsoft uses $\epsilon = 1$ to collect telemetry data from Windows users [5], while Apple's choice of ϵ in iOS and macOS ranges between 2 and 8 for different tasks [4]. Regarding our experiments, and according to the Table 2, while an ϵ of 1 results in an acceptable performance in some settings (e.g. link prediction on Flickr), it seems that setting ϵ to 5 leads to near-optimal results in almost all cases.

6 Conclusion

In this paper, we presented a privacy-preserving framework for graph representation learning based on graph convolutional networks and local differential privacy to address the node-level privacy, when graph nodes have sensitive attributes that are kept private, but they could be advantageous to a GNN for learning rich node representations by a central server. To this end, we proposed *Private Graph Convolution*, which is a ϵ-locally differentially private mechanism, for graph nodes to perturb their private features before sending them to the server, and presented an unbiased estimator for the server by which it can estimate the first-layer graph convolution of the GNN using the perturbed features. Experimental experiments over real-work network datasets on node classification and link prediction tasks demonstrated that the proposed framework can manage the privacy-utility trade-off in learning representation for graph nodes, performs better than the classic Laplace mechanism consistently, and outperforms the Piecewise mechanism when ϵ is small or when the feature distribution is skewed.

Several potential future directions and improvements are imaginable for this work. One of the limitations of the current method is that for any node in the graph, a separate privacy budget is allocated for each individual feature. Therefore, the total privacy budget of a single node scales linearly with the number of features. While this is a common practice to assign separate privacy budgets to different features (for instance, Apple uses different dedicated privacy budgets for emoji suggestions, lookup hints, QuickType suggestions, etc [38]), it can be problematic when the server request high-dimensional features which are highly correlated. Therefore, as future work, we plan to work on a multi-dimensional LDP mechanism to address this issue. As another future step, we would like to work on a hybrid mechanism that is optimal in terms of expected noise-variance. Finally, it might be interesting to address the problem of node-level privacy using other privacy-preserving machine learning paradigms, such as secure multi-party computation. Overall, the concept of privacy-preserving graph representation learning is a novel and unexplored field of research with many potential future directions that can go beyond node-level privacy.

Acknowledgments and Disclosure of Funding

We would like to thank Hamed Haddadi for his helpful advice and comments on the paper. This work was supported as part of the Dusk2Dawn project by the Swiss National Science Foundation (SNSF) through the Sinergia interdisciplinary program (grant number 173696).

References

[1] Nitin Agrawal, Ali Shahin Shamsabadi, Matt J Kusner, and Adrià Gascón. Quotient: two-party secure neural network training and prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1231–1247, 2019.

[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[3] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[4] Apple Differential Privacy Team. Learning with privacy at scale. `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pd`

[5] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580, 2017.

[6] John C Duchi, Michael I Jordan, and Martin J Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[7] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

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

[9] 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.

[10] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[11] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[12] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. A pragmatic introduction to secure multi-party computation. *Foundations and Trends® in Privacy and Security*, 2(2-3):70–246, 2018.

[13] WA Falcon. Pytorch lightning. *GitHub. Note: https://github. com/williamFalcon/pytorch-lightning Cited by*, 3, 2019.

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

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[17] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. Cryptodl: Deep neural networks over encrypted data. *arXiv preprint arXiv:1711.05189*, 2017.

[18] Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018(3):123–142, 2018.

[19] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[20] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. Differential privacy: An economic method for choosing epsilon. In *2014 IEEE 27th Computer Security Foundations Symposium*, pages 398–410. IEEE, 2014.

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

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

[23] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

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

[25] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[26] Jaewoo Lee and Chris Clifton. How much is enough? choosing ϵ for differential privacy. In Xuejia Lai, Jianying Zhou, and Hui Li, editors, *Information Security*, pages 325–340, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[27] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[28] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[29] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. *IACR Cryptology ePrint Archive*, 2017:396, 2017.

[30] AJ Paverd, Andrew Martin, and Ian Brown. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep.*, 2014.

[31] Sungmin Rhee, Seokjun Seo, and Sun Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

[32] Bita Darvish Rouhani, M Sadegh Riazi, and Farinaz Koushanfar. Deepsecure: Scalable provably-secure deep learning. *arXiv preprint arXiv:1705.08963*, 2017.

[33] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *arXiv preprint arXiv:1909.13021*, 2019.

[34] Sai Sri Sathya, Praneeth Vepakomma, Ramesh Raskar, Ranjan Ramachandra, and Santanu Bhattacharya. A review of homomorphic encryption libraries for secure computation. *arXiv preprint arXiv:1812.02428*, 2018.

[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[36] R. Sheikh and D. K. Mishra. Protocols for getting maximum value for multi-party computations. In *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, pages 597–600, 2010.

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

[38] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and Xiaofeng Wang. Privacy loss in apple's implementation of differential privacy on macos 10.12. *arXiv preprint arXiv:1709.02753*, 2017.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[40] Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning. *arXiv preprint arXiv:1812.03288*, 2018.

[41] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE, 2019.

[42] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322, 2017.

[43] 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.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[45] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

[46] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

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

[48] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5829–5836, 2019.

[49] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

A Related LDP Mechanisms

A.1 Laplace Mechanism

Laplace mechanism (LM) [9] is a classic differentially private algorithm that can be used in both the global and the local differential privacy setting. In the local approach, each data subject basically adds a random noise to its data, which is drawn from $Lap(y \mid \frac{\Delta}{\epsilon})$, where:

$$Lap(y \mid \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|y|}{\lambda}\right)$$

is the zero-mean Laplace distribution with scale parameter λ. In our experiments, we independently add Laplace noise to each individual feature x_v for every node v. Formally, we set:

$$y_v^* = x_v + y_v \tag{9}$$

where y_v is a random noise drawn from $Lap(y_v \mid \frac{\Delta}{\epsilon})$. As the mean of this distribution is zero, the Laplace mechanism is unbiased, i.e. $\mathbb{E}[y_v^*] = x_v$. Furthermore, the variance of the Laplace mechanism is by definition equal to:

$$Var[y_v^*] = 2(\frac{\Delta}{\epsilon})^2 \tag{10}$$

A.2 Duchi's Mechanism

Another LDP mechanism for numeric data collection and mean estimation, from which our PGC method is derived, is proposed by Duchi et al. [6] (denoted as DM). This mechanism accepts an input value $t \in [-1, 1]$ and outputs a perturbed value $t^* \in \{\frac{e^\epsilon+1}{e^\epsilon-1}, -\frac{e^\epsilon+1}{e^\epsilon-1}\}$ according to the following probability distribution:

$$\Pr[t^*|t] = \begin{cases} \frac{1}{2} + \frac{e^\epsilon-1}{2e^\epsilon+2} \cdot t, & \text{if } t^* = \frac{e^\epsilon+1}{e^\epsilon-1} \\ \frac{1}{2} - \frac{e^\epsilon-1}{2e^\epsilon+2} \cdot t, & \text{if } t^* = -\frac{e^\epsilon+1}{e^\epsilon-1} \end{cases} \tag{11}$$

Duchi et al. show that the above mechanism satisfies ϵ-LDP, and that t^* is an unbiased estimator of the input t. Also, the variance of the mechanism is calculated as:

$$Var[t^*] = \left(\frac{e^\epsilon+1}{e^\epsilon-1}\right)^2 - t^2 \tag{12}$$

A.2.1 Derivation of PGC from Duchi's mechanism

Here we describe how the PGC method is derived from DM. First, we modify DM to accept inputs withing the range $[\alpha, \beta]$ for any individual feature x_v of any node v:

$$\begin{aligned}
\alpha \leq x_v \leq \beta &\implies 0 \leq x_v - \alpha \leq \Delta \\
&\implies 0 \leq \frac{x_v - \alpha}{\Delta} \leq 1 \\
&\implies 0 \leq \frac{2(x_v - \alpha)}{\Delta} \leq 2 \\
&\implies -1 \leq \frac{2(x_v - \alpha)}{\Delta} - 1 \leq 1
\end{aligned} \tag{13}$$

Based on the above, let t_v be defined as:

$$t_v = \frac{2(x_v - \alpha)}{\Delta} - 1 \tag{14}$$

which is in the range $[-1, 1]$ and thus can be used as the input of DM. From (11), we have:

$$\Pr\left[t_v^* = \frac{e^\epsilon + 1}{e^\epsilon - 1} \mid t_v\right] = \frac{1}{2} + \frac{e^\epsilon - 1}{2e^\epsilon + 2} \cdot t_v$$

Substituting (14) in the above, we get:

$$\begin{aligned}
\Pr\left[t_v^* = \frac{e^\epsilon + 1}{e^\epsilon - 1} \mid x_v\right] &= \frac{1}{2} + \frac{e^\epsilon - 1}{2e^\epsilon + 2} \cdot \left(\frac{2(x_v - \alpha)}{\Delta} - 1\right) \\
&= \frac{1}{2} + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} - \frac{1}{2} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \\
&= \frac{1}{2}\left(1 - \frac{e^\epsilon - 1}{e^\epsilon + 1}\right) + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \\
&= \frac{1}{2} \cdot \frac{2}{e^\epsilon + 1} + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \\
&= \frac{1}{e^\epsilon + 1} + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}
\end{aligned} \tag{15}$$

Similar to the above, we will have:

$$\Pr\left[t_v^* = -\frac{e^\epsilon + 1}{e^\epsilon - 1} \mid x_v\right] = \frac{e^\epsilon}{e^\epsilon + 1} - \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \tag{16}$$

Analogous to [5] and to achieve communication efficiency, for each node v, we define $y_v \in \{0, 1\}$ to be the output of the data collection step, such that:

$$y_v = \begin{cases} 1, & \text{if } t_v^* = \frac{e^\epsilon + 1}{e^\epsilon - 1} \\ 0, & \text{if } t_v^* = -\frac{e^\epsilon + 1}{e^\epsilon - 1} \end{cases} \tag{17}$$

Combining (15), (16), and (17), we get:

$$\Pr[y_v | x_v] = \begin{cases} \frac{1}{e^\epsilon + 1} + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}, & \text{if } y_v = 1 \\ \frac{e^\epsilon}{e^\epsilon + 1} - \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}, & \text{if } y_v = 0 \end{cases} \tag{18}$$

which is equal to PGC data collection mechanism defined in (3). From the above, we have:

$$\mathbb{E}[y_v] = \frac{1}{e^\epsilon + 1} + \frac{x_v - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \tag{19}$$

which is clearly biased as the expectation is not equal to x_v. To get an unbiased estimator, let y_v^* be a random variable, such that $\mathbb{E}[y_v^*] = x_v$. Therefore, Combining with the above equation, we have:

$$\begin{aligned}
\frac{1}{e^\epsilon + 1} + \frac{\mathbb{E}[y_v^*] - \alpha}{\Delta} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} &= \mathbb{E}[y_v] \\
1 + \frac{\mathbb{E}[y_v^*] - \alpha}{\Delta} \cdot (e^\epsilon - 1) &= (e^\epsilon + 1)\,\mathbb{E}[y_v] \\
\frac{\mathbb{E}[y_v^*] - \alpha}{\Delta} &= \frac{(e^\epsilon + 1)\,\mathbb{E}[y_v] - 1}{e^\epsilon - 1} \\
\mathbb{E}[y_v^*] &= \frac{(e^\epsilon + 1)\,\mathbb{E}[y_v] - 1}{e^\epsilon - 1}\Delta + \alpha
\end{aligned}$$

Finally, removing the expectations from both sides, we end up with:

$$y_v^* = \frac{(e^\epsilon + 1)y_v - 1}{e^\epsilon - 1}\Delta + \alpha \tag{20}$$

which is the scalar form of the unbiased estimator defined by (5).

A.3 Piecewise Mechanism

A recently proposed LDP algorithm for numeric mean estimation is the Piecewise mechanism (PM) by Wang et al. [41]. This mechanism assumes that the input t is again in the range $[-1, 1]$ and returns a perturbed value $t^* \in [-C, C]$, where:

$$C = \frac{\exp(\epsilon/2) + 1}{\exp(\epsilon/2) - 1} \tag{21}$$

Then, the output t^* is drawn randomly from the following piecewise constant distribution:

$$\Pr(t^* \mid t) = \begin{cases} p, & \text{if } t^* \in \big[\ell(t),\, r(t)\big], \\ \frac{p}{\exp(\epsilon)}, & \text{if } t^* \in \big[-C,\, \ell(t)\big) \cup \big(r(t),\, C\big]. \end{cases} \tag{22}$$

where

$$\begin{aligned}
p &= \frac{\exp(\epsilon) - \exp(\epsilon/2)}{2\exp(\epsilon/2) + 2}, \\
\ell(t) &= \frac{C+1}{2} \cdot t - \frac{C-1}{2}, \text{ and} \\
r(t) &= \ell(t) + C - 1.
\end{aligned}$$

Wang et al. prove that their mechanism is unbiased ($\mathbb{E}[t^*] = t$), and the variance is calculated as:

$$Var[t^*] = \frac{t^2}{e^{\epsilon/2} - 1} + \frac{e^{\epsilon/2} + 3}{3(e^{\epsilon/2} - 1)^2} \tag{23}$$

A.3.1 Integration of PM into the proposed framework

In order to make the variance of PM comparable to PGC's defined in (7), we extend the mechanism to accept inputs withing the range $[\alpha, \beta]$ for any individual feature x_v of any node v. Based on (13), we set:

$$\begin{aligned}
t_v &= \frac{2(x_v - \alpha)}{\Delta} - 1 \\
&= \frac{2(x_v - \alpha) - (\beta - \alpha)}{\Delta} \\
&= \frac{2}{\Delta}(x_v - \frac{\alpha + \beta}{2})
\end{aligned} \tag{24}$$

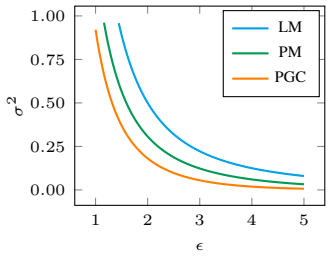


Figure 4: Variance of different LDP mechanisms for $x_v = \alpha$ (or $x_v = \beta$). The values of α and β was set to 0 and 1, respectively.

Combining (23) and (24), the variance becomes:

$$\begin{aligned} Var[t_v^*] &= \frac{{t_v}^2}{e^{\epsilon/2}-1} + \frac{e^{\epsilon/2}+3}{3(e^{\epsilon/2}-1)^2} \\ &= \frac{\left(\frac{2}{\Delta}(x_v - \frac{\alpha+\beta}{2})\right)^2}{e^{\epsilon/2}-1} + \frac{e^{\epsilon/2}+3}{3(e^{\epsilon/2}-1)^2} \\ &= \frac{4}{\Delta^2}\frac{\left(x_v - \frac{\alpha+\beta}{2}\right)^2}{e^{\epsilon/2}-1} + \frac{e^{\epsilon/2}+3}{3(e^{\epsilon/2}-1)^2} \end{aligned} \tag{25}$$

Let y_v^* be a random variable such that $\mathbb{E}[y_v^*] = x_v$. Since $\mathbb{E}[t_v^*] = t_v$, using (24) we have:

$$\begin{aligned} \mathbb{E}[t_v^*] &= \frac{2}{\Delta}(\mathbb{E}[y_v^*] - \frac{\alpha+\beta}{2}) \\ &= \mathbb{E}\left[\frac{2}{\Delta}(y_v^* - \frac{\alpha+\beta}{2})\right] \end{aligned}$$

removing the expectations from both sides and rearranging, we get:

$$y_v^* = \frac{\Delta}{2}(t_v^* + 1) + \alpha \tag{26}$$

The variance of y_v^* is then derived as:

$$Var[y_v^*] = Var\left[\frac{\Delta}{2}(t_v^* + 1) + \alpha\right] = \frac{\Delta^2}{4}Var[t_v^*]$$

Finally, combining with (25), we have:

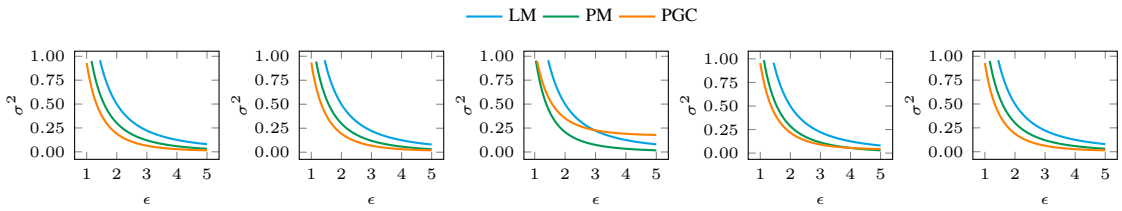
$$Var[y_v^*] = \frac{\left(x_v - \frac{\alpha+\beta}{2}\right)^2}{e^{\epsilon/2}-1} + \frac{e^{\epsilon/2}+3}{3(e^{\epsilon/2}-1)^2} \cdot \frac{\Delta^2}{4} \tag{27}$$

B Variance Comparison of LDP Mechanisms

Equation (27) clearly shows that the minimum variance of PM is achieved when $x_v = \frac{\alpha+\beta}{2}$, while its variance is maximized when $x_v = \alpha$ or $x_v = \beta$. In other words, as opposed to the PGC mechanism, whose variance decreases to its minimum as x_v approaches α or β, the variance of PM gets elevated toward its maximum. This is illustrated further in Figure 4, where the variance of the three LDP mechanisms, LM, PM, and PGC, is depicted for the case when $x_v = \alpha$ or $x_v = \beta$. It can be observed that in this case, PGC generates less variance than both LM and PM, regardless of the value of ϵ. Therefore, considering that the feature x_v is generated from some prior distribution $p(x)$ defined on $[\alpha, \beta]$, PGC is expected to be a better choice if $p(x)$ is skewed toward either α or β, e.g. when $p(x)$ is a truncated exponential distribution defined over the range $[\alpha, \beta]$, so that $\mathbb{E}[x]$ is close to α or β. If α is zero, this implies that PGC works best if most of the features are sparse. On the

Table 3: Average feature value $\bar{x}$ in different datasets

DATASET	CITESEER	CORA	ELLIPTIC	FLICKR	TWITCH
$\bar{x}$	0.009	0.013	0.219	0.037	0.008



(a) CiteSeer (b) Cora (c) Elliptic (d) Flickr (e) Twitch

Figure 5: Variance of LDP mechanisms based on different values of the privacy budget ϵ obtained using empirical average of features across different datasets.

contrary, if $\mathbb{E}[x]$ is close to the middle of the range $[\alpha, \beta]$, e.g. when $p(x)$ is a Gaussian or uniform distribution over $[\alpha, \beta]$, then PM is preferred over PGC (as an extension of DM) when ϵ is greater than around 0.61 [41].

Figure 5 exhibits the empirical variance of the three LDP methods for $\epsilon \in [1, 5]$ across different datasets. For each dataset, the value of x_v in the PM and PGC's variance formulas (equations (27) and (7)) is replaced by the average of all the features over all the nodes, denoted as $\bar{x}$:

$$\bar{x} = \frac{1}{m \cdot |\mathcal{V}|} \sum_{v \in \mathcal{V}} \sum_{i=1}^{m} x_v^{(i)} \tag{28}$$

Here, we normalized the features between 0 and 1, so that they have the same scale. The value of $\bar{x}$ for different datasets is shown in Table 3. According to Figure 5 and Table 3, it can be observed that for all the datasets except Elliptic, the empirical variance of PGC is lower than PM and LM, as in these datasets the average feature value tends to be very close to 0. However, in case of Elliptic, for which $\bar{x}$ is far greater than the rest of the datasets, the variance of PGC exceeds PM, and for higher values of ϵ it also passes LM. This result is completely in line with error estimation, node classification, and link prediction results presented in Section 5.

C Proof of Lemmas

C.1 Proof of Lemma 1

Proof. According to (3), for a single output y_v corresponding to an arbitrary private input feature x_v, the probability that $y_v = 1$ ranges from $\frac{1}{e^\epsilon+1}$ to $\frac{e^\epsilon}{e^\epsilon+1}$ depending on x_v. Analogously, the probability that $y_v = 0$ also varies from $\frac{1}{e^\epsilon+1}$ to $\frac{e^\epsilon}{e^\epsilon+1}$. Therefore, for every node pairs v and u and every output {0,1}, the ratio of the respected probabilities can be at most e^ϵ. As each nodes contributes with its feature to the server only once, due to the robustness of differentially private algorithms to post-processing [10], Algorithm 1 will be ϵ-LDP for individual node features. □

C.2 Proof of Lemma 2

Proof. We need to show that $\mathbb{E}[\widehat{GC}(v)] = GC(v)$.

$$\begin{aligned}
\mathbb{E}[\widehat{GC}(v)] &= \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{y}_v^*}{\sqrt{d_u d_v}} \\
&= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \left[\frac{(e^\epsilon + 1)\, \mathbb{E}[\mathbf{y}_u] - 1}{e^\epsilon - 1} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \right]
\end{aligned} \tag{29}$$

As $\mathbf{y}_u$ is drawn from a Bernoulli distribution, for any node u we have:

$$\mathbb{E}[\mathbf{y}_u] = \frac{1}{e^\epsilon + 1} + \frac{\mathbf{x}_u - \boldsymbol{\alpha}}{\boldsymbol{\Delta}} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1} \tag{30}$$

Combining (29) and (30) we get

$$\begin{aligned}
\mathbb{E}[\widehat{GC}(v)] &= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \left[\frac{(e^\epsilon + 1)(\frac{1}{e^\epsilon + 1} + \frac{\mathbf{x}_u - \boldsymbol{\alpha}}{\boldsymbol{\Delta}} \cdot \frac{e^\epsilon - 1}{e^\epsilon + 1}) - 1}{e^\epsilon - 1} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \right] \\
&= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \left[\frac{1 + \frac{\mathbf{x}_u - \boldsymbol{\alpha}}{\boldsymbol{\Delta}}(e^\epsilon - 1) - 1}{e^\epsilon - 1} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \right] \\
&= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_u d_v}} \left[\frac{\mathbf{x}_u - \boldsymbol{\alpha}}{\boldsymbol{\Delta}} \cdot \boldsymbol{\Delta} + \boldsymbol{\alpha} \right] \\
&= \sum_{u \in \mathcal{N}(v)} \frac{\mathbf{x}_u}{\sqrt{d_u d_v}} \\
&= GC(v)
\end{aligned}$$

□

C.3 Proof of Lemma 3

Proof. Let $z_v = \frac{y_v}{\sqrt{d_v}} \in [0, 1]$. Then using Hoeffding's inequality [19], for all $t > 0$ we have:

$$\Pr\left[\left| \sum_{u \in \mathcal{N}(v)} z_u - \mathbb{E}\left(\sum_{u \in \mathcal{N}(v)} z_u \right) \right| \geq t \right] \leq 2 \exp\{-\frac{2t^2}{d_v}\}$$

Substituting z_u with $\frac{y_u}{\sqrt{d_u}}$ and combining with (30) we get

$$
\begin{aligned}
&\Pr\left[\left|\sum_{u\in\mathcal{N}(v)} z_u - \mathbb{E}\left(\sum_{u\in\mathcal{N}(v)} z_u\right)\right| \geq t\right] \\
&= \Pr\left[\left|\sum_{u\in\mathcal{N}(v)} \frac{y_u}{\sqrt{d_u}} - \sum_{u\in\mathcal{N}(v)} \frac{\mathbb{E}\left(y_u\right)}{\sqrt{d_u}}\right| \geq t\right] \\
&= \Pr\left[\left|\sum_{u\in\mathcal{N}(v)} \frac{y_u}{\sqrt{d_u}} - \sum_{u\in\mathcal{N}(v)} \frac{\frac{1}{e^\epsilon+1} + \frac{x_u-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}}{\sqrt{d_u}}\right| \geq t\right] \\
&= \Pr\left[\left|\sum_{u\in\mathcal{N}(v)} \frac{1}{\sqrt{d_u}}\left(y_u - \frac{1}{e^\epsilon+1} - \frac{x_u-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)\right| \geq t\right] \\
&= \Pr\left[\left|\sum_{u\in\mathcal{N}(v)} \left[\frac{1}{\sqrt{d_u}}\left(\frac{(e^\epsilon+1)y_u-1}{e^\epsilon-1}\cdot\Delta+\alpha\right) - \frac{x_u}{\sqrt{d_u}}\right]\right| \geq \frac{e^\epsilon+1}{e^\epsilon-1}\Delta\cdot t\right] \\
&= \Pr\left[\left|\sqrt{d_v}\sum_{u\in\mathcal{N}(v)} \left[\frac{1}{\sqrt{d_u d_v}}\left(\frac{(e^\epsilon+1)y_u-1}{e^\epsilon-1}\cdot\Delta+\alpha\right) - \frac{x_u}{\sqrt{d_u d_v}}\right]\right| \geq \frac{e^\epsilon+1}{e^\epsilon-1}\Delta\cdot t\right] \\
&= \Pr\left[\left|\sqrt{d_v}\left(\widehat{GC}(v) - GC(v)\right)\right| \geq \frac{e^\epsilon+1}{e^\epsilon-1}\Delta\cdot t\right] \\
&= \Pr\left[\left|\widehat{GC}(v) - GC(v)\right| \geq \frac{e^\epsilon+1}{e^\epsilon-1}\frac{t}{\sqrt{d_v}}\cdot\Delta\right] \leq 2\exp\{-\frac{2t^2}{d_v}\}
\end{aligned} \tag{31}
$$

Setting $2\exp\{-\frac{2t^2}{d_v}\} = \delta$, we get $t = \sqrt{\frac{d_v}{2}\log\frac{2}{\delta}}$. Then by combining with (31), we have

$$
\Pr\left[\left|\widehat{GC}(v) - GC(v)\right| \geq \Delta\cdot\frac{e^\epsilon+1}{e^\epsilon-1}\sqrt{\frac{1}{2}\log(\frac{2}{\delta})}\right] \leq \delta
$$

and therefore

$$
\Pr\left[\left|\widehat{GC}(v) - GC(v)\right| \leq \Delta\cdot\frac{e^\epsilon+1}{e^\epsilon-1}\sqrt{\frac{1}{2}\log(\frac{2}{\delta})}\right] \geq 1-\delta \tag{32}
$$

which concludes the proof. □

C.4 Proof of Lemma 4

Proof. We first calculate the variance of y_v as in (3). Since y_v is Bernoulli, its variance can be obtained as:

$$
Var[y_v] = \mathbb{E}[y_v](1-\mathbb{E}[y_v]) \tag{33}
$$

Combining with (30), we get:

$$
\begin{aligned}
Var[y_v] &= \left(\frac{1}{e^\epsilon+1} + \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)\left(1-\left[\frac{1}{e^\epsilon+1} + \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right]\right) \\
&= \left(\frac{1}{e^\epsilon+1} + \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)\left(\frac{e^\epsilon}{e^\epsilon+1} - \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right) \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} - \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{(e^\epsilon+1)^2} + \frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon(e^\epsilon-1)}{(e^\epsilon+1)^2} - \left(\frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2 \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} + \left(\frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{(e^\epsilon+1)^2}\right)(e^\epsilon-1) - \left(\frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2 \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} + \frac{x_v-\alpha}{\Delta}\cdot\left(\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2 - \left(\frac{x_v-\alpha}{\Delta}\cdot\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2 \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} + \left(\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2\left[\frac{x_v-\alpha}{\Delta} - \left(\frac{x_v-\alpha}{\Delta}\right)^2\right] \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} + \left(\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2\left(\frac{x_v-\alpha}{\Delta}\right)\left(1-\frac{x_v-\alpha}{\Delta}\right) \\
&= \frac{e^\epsilon}{(e^\epsilon+1)^2} + \left(\frac{e^\epsilon-1}{e^\epsilon+1}\right)^2\left(\frac{x_v-\alpha}{\Delta}\right)\left(\frac{\beta-x_v}{\Delta}\right) \qquad (34)
\end{aligned}
$$

Finally, using the above equation and according to (5), we have:

$$
\begin{aligned}
Var[y_v^*] &= Var\left[\frac{(e^\epsilon+1)y_v-1}{e^\epsilon-1}\cdot\Delta+\alpha\right] \\
&= \Delta^2\cdot\left(\frac{e^\epsilon+1}{e^\epsilon-1}\right)^2 Var[y_v] \\
&= \Delta^2\frac{e^\epsilon}{(e^\epsilon-1)^2} + (x_v-\alpha)(\beta-x_v) \qquad (35)
\end{aligned}
$$

□

D Additional Dataset Details

Citation networks. We use three well-known benchmark datasets, namely Cora and CiteSeer [45]. These datasets are citation networks, where documents are represented as nodes, and edges represent citation links. Additionally, each node has a bag-of-words feature vector and a label indicating its category.

Elliptic. This dataset is the network of Bitcoin transactions released by Elliptic[1]. Each node in this dataset represents a transaction, and edges represent the flow of Bitcoin between two transactions. Around 23% of the nodes in the dataset have been labeled as being created by a "licit" or "illicit" entity. Vertex features comprise local and aggregated information about the transactions.

Flickr. We also use the Flickr dataset from [46], in which nodes represent images uploaded to Flickr, and edges indicate that images share some common attributes, such as location, gallery, or comments by the same user. The dataset also contains bag-of-words feature vectors for each node, and the nodes are labeled according to their tags.

Twitch. The last dataset is collected from Twitch, which is a social networking platform for gamers. Nodes in the dataset correspond to Twitch users, and edges represent mutual friendships between them. Node features are extracted based on location, games played and liked, and streaming habits. The label of each node determines if the user uses explicit language. Here, we use the UK version of this dataset from [33].

[1] `https://www.kaggle.com/ellipticco/elliptic-data-set`

Table 4: Best hyper-parameter configurations based on validation set performance.

	GCN for Node Classification			VGAE for Link Prediction		
Dataset	Learning Rate	Weight Decay	Dropout	Learning Rate	Weight Decay	Dropout
CiteSeer	0.01	0.1	0.5	0.01	0.01	0
Cora	0.01	0.01	0.5	0.01	0.01	0
Elliptic	0.01	0	0	0.01	0.0001	0
Flickr	0.001	0.0001	0	0.001	0.001	0
Twitch	0.001	0.0001	0	0.01	0.001	0

E Hyper-Parameter Configuration

We performed grid search in order to find the best choices for initial learning rate, weight decay, and dropout rate based on the performance of each model, GCN for node classification and VGAE for link prediction, across different datasets. Table 4 displays the best performing hyper-parameters used for our experiments.