# Preserving Node-level Privacy in Graph Neural Networks

Zihang Xiang
KAUST

Tianhao Wang
University of Virginia

Di Wang
KAUST

2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00270

***Abstract*—Differential privacy (DP) has seen immense applications in learning on tabular, image, and sequential data where instance-level privacy is concerned. In learning on graphs, contrastingly, works on *node-level* privacy are highly sparse. Challenges arise as existing DP protocols hardly apply to the message-passing mechanism in Graph Neural Networks (GNNs).**

**In this study, we propose a solution that specifically addresses the issue of *node-level* privacy. Our protocol consists of two main components: 1) a sampling routine called *Heter-Poisson*, which employs a specialized node sampling strategy and a series of tailored operations to generate a batch of sub-graphs with desired properties, and 2) a randomization routine that utilizes symmetric multivariate Laplace (SML) noise instead of the commonly used Gaussian noise. Our privacy accounting shows this particular combination provides a non-trivial privacy guarantee. In addition, our protocol enables GNN learning with good performance, as demonstrated by experiments on five real-world datasets; compared with existing baselines, our method shows significant advantages, especially in the high privacy regime. Experimentally, we also 1) perform membership inference attacks against our protocol and 2) apply privacy audit techniques to confirm our protocol's privacy integrity.**

**In the sequel, we present a study on a seemingly appealing approach [33] (USENIX'23) that protects *node-level* privacy via differentially private node/instance embeddings. Unfortunately, such work has fundamental privacy flaws, which are identified through a thorough case study. More importantly, we prove an impossibility result of achieving both (strong) privacy and (acceptable) utility through private instance embedding. The implication is that such an approach has intrinsic utility barriers when enforcing differential privacy.**

***Index Terms*—Node-level Privacy; Differential Privacy; Graph Neural Networks**

## 1. Introduction

Modern machine learning/deep learning systems exhibit privacy risks: over-fitting by neural networks results in memorization of the training data [48] and leads to unacceptable privacy risk [36]; it is known that only sharing model updates still leads to privacy infringements as the adversary can reconstruct the original training data [52]. Among all learning tasks, there has been an increasing focus on learning from non-Euclidean data, particularly from graph data utilizing the message-passing mechanism or Graph Neural Networks (GNNs). Unfortunately, similar to the privacy issues in learning tasks on tabular, sequential, or image data, privacy breaches are also observed when learning GNNs on graphs. Wu et al. [42] provided evidence that the presence of an edge could be inferred with access to the trained GNN model. There are also membership inference attacks against GNNs under various threat models [41].

These privacy issues underscore the importance of applying a formal approach to protect data privacy. Among all privacy-enhancing technologies, Differential Privacy (DP) [10] has emerged as a widely adopted approach. Simply speaking, DP introduces calibrated randomness into output intended to be accessible by the public, as this operation conceals the evidence of participation and encourages users' trust [39]. Previously, DP has been effectively and extensively applied to preserve instance-level privacy in learning tasks for image data in convolutional neural networks [1], sequential data in recurrent neural networks [23] and language models [47]. Among various private learning protocols, differentially private stochastic gradient descent (DP-SGD, *a.k.a.*, NoisySGD) [1], [37] is a prime example.

For graph data, the "instance" whose privacy needs to be protected can be an *edge* or a *node*. The former is called *edge-level* privacy, and the latter *node-level* privacy. In *edge-level* privacy, DP prevents the adversary from confidently inferring whether an *edge* between some nodes exists. In *node-level* privacy, DP prevents the adversary from confidently inferring whether a node has contributed to training a GNN, *i.e.*, membership inference attacks. For the *edge-level* privacy, there has been some work where the existences of some *edge* are hidden from being inferred [20], [42].

However, in contrast to all the above successful applications of DP, the protection of *node-level* privacy in GNNs remains largely unanswered. In fact, the success of existing popular applied privacy solutions does not transfer to *node-level* privacy, and existing work tackling this problem fails to give strong results. This suggests that preserving *node-level* privacy in GNN is non-trivial and more challenging. To have a first intuition on such difficulties, the message-passing mechanism requires a node to iteratively aggregate information from its neighbors, thus emphasizing interconnections (dependencies) between nodes; however, existing privacy protocols almost assume data instances are independent during the learning process, *i.e.*, settings are conflicting.

By referencing the DP-SGD protocol, let's dive deeper into some details. The sensitivity analysis, together with the algorithm to ensure bounded sensitivity, is the crucial element in the privacy analysis. Specifically, in the gradient



DOI 10.1109/SP54263.2024.00182

perturbation method, one can enforce bounded sensitivity even if the loss function has no Lipschitz property [1]. This is done by artificially clipping the $\ell_2$ norm of *per-example*'s gradient. However, those techniques for enforcing bounded sensitivity are not readily applicable to GNNs because of the particular behavior in message-passing. Specifically, it is not evident what should be the *per-example* counterpart in GNNs in the first place. Moreover, there are also issues with privacy accounting. It is well-known that sampling on instances leads to privacy amplification, which benefits privacy. However, the sampling on a graph often requires a node to sample its neighbors, and we have difficulties in calculating the sampling rate for a given node as nodes' edges can be arbitrary.

The above negative results suggest fundamentally new treatments are needed for preserving *node-level* privacy in GNNs. However, as techniques that can be leveraged are sparse, new challenges arise not only in algorithm design but also in privacy accounting. Recent work demonstrates real-world node privacy risks [13], indicating *node-level* privacy protection is of equal urgency. This is our primary motivation.

**This work**. Centered around preserving *node-level* privacy in learning GNNs on graphs, we begin by formulating the privacy problem. We also briefly discuss recent work on this problem and show their limitations. Subsequently, to better understand our design, we present a motivating experiment showing how we trace one node's impact. We find that the more out-edges (pointing to other nodes) a node has, the greater that node impacts the GNN model update. Intuitively, it is because a node has more "channels" to deliver impact. Based on such key observations, we form a neighborhood sampling strategy as a countermeasure to offset such impact, which is a part of our method, namely *HeterPoisson*.

In addition, *HeterPoisson* also enforces independent node sampling behavior to form sub-graphs, and independent trial enables clear and precise privacy analysis. In our algorithmic solution, we take the sub-graph as the *per-example* counterpart to compute the per-sub-graph gradient, followed by enforcing the per-sub-graph gradient with bounded $\ell_2$ norm. Such gradient clipping operation leverages the idea of DP-SGD, *i.e.*, separate-then-bound. However, current state-of-the-art privacy accounting methods [25] for DP-SGD (Gaussian mechanism with sub-sampling) can't be applied because the settings do not fit. Fortunately, due to the design of *HeterPoisson* and using symmetric multivariate Laplace (SML) random noise, we have a non-trivial privacy guarantee.

In the privacy accounting section, we prove a privacy guarantee due to *HeterPoisson* and SML instead of Gaussian. As will be analyzed in detail, such a random noise choice is tailored to *HeterPoisson* and leads to non-trivial privacy bound. In contrast, Gaussian noise does not. In principle, a whole family of random noise we introduce in the privacy accounting section can be leveraged to ensure DP as long as the noise suits the algorithmic operations. This is possibly useful for other related problems and is of independent interest.

In the evaluation section, our experiments on five real-world graph datasets also demonstrate that our solution provides non-trivial utility; compared with several existing baselines, our performance shows significant advantages, especially in high privacy regimes. Ablation studies are also provided to understand the effectiveness of our design choices. In addition to the performance evaluations, to show our solution's privacy integrity, we include 1) privacy attack experiments to show our solution's resistance to attacks by a strong privacy adversary and 2) privacy audit experiments to show that our privacy accounting has no bugs.

In the final part of this work, we present an in-depth study on a seemingly appealing approach [33] (USENIX'23) that protects *node-level* privacy via differentially private node/instance embeddings. However, we identify fundamental privacy flaws carried by such work through a case study. More importantly, we show an impossibility result of achieving both (strong) privacy and (acceptable) utility through private instance embedding. Such results have implications: differentially private instance embedding has intrinsic utility barriers. Accordingly, experiments to confirm our results are also provided. Our study highlights pitfalls that should be avoided in privacy applications. To give a concise summary of our contribution:

- We provide an algorithmic solution together with its privacy accounting for preserving *node-level* privacy in learning GNNs on graphs.
- We show our solution's effectiveness on real-world datasets, and we also experimentally show our method's privacy integrity and resilience by performing privacy audit and privacy attacks to show our protocol.
- We study and identify privacy flaws for private instance embedding used by some previous work. We also prove an impossibility result for such an approach and provide experimental evaluations.

## 2. Background

### 2.1. Graph Neural Networks

**Graph**. A graph is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the node set, $\mathcal{E}$ is the edge set, and $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the feature matrix whose $i$-th row is a $d$-dimensional vector of node $i$. In node classification tasks, we have additional information $\mathbf{Y}$, which stores the labels for each node with $C$ possible classes. We use $(i \to j) \in \mathcal{E}$ to denote the existence of an edge from node $i$ to $j$. For an undirected graph, $(i \to j) \in \mathcal{E}$ implies $(j \to i) \in \mathcal{E}$. We say $(i \to j) \in \mathcal{E}$ is an out-edge of $i$, and an in-edge of $j$. If $(i \to j) \in \mathcal{E}$, we use $i \in \mathcal{NB}(j)$ to denote $i$ is a neighbor of $j$. For node $i$, we use *in-degree* and *out-degree* to denote the size of its in-edges and out-edges, respectively.

**Definition 1** (GNN)**.** *A graph neural network (GNN) follows the message-passing mechanism: iteratively, it updates each node's embedding by aggregating information from neighbors. The $k$-th update is formulated as follows.*

$$\mathbf{m}_{\mathcal{NB}(u)}^{(k)} = \mathbf{AGG}^{(k)}\left(\left\{\mathbf{h}_v^{(k)} | v \in \mathcal{NB}(u)\right\}\right)$$
$$\mathbf{h}_u^{(k+1)} = \phi^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{NB}(u)}^{(k)}\right)$$

*where* $\mathbf{h}_u^{(k)}$ *is the obtained embedding for node* $u$ *and* $\mathbf{m}_{\mathcal{NB}(u)}^{(k)}$ *stands for the aggregated "message" from neighbors.* $\phi$ *is the update function with learnable parameters, often instantiated as a multilayer perceptron (MLP).* $\mathbf{AGG}$*, which aggregates information, is an arbitrary differentiable function.*

Usually, the final resultant node embedding is used as the representation vector for a node. In GNNs, there are many aggregation methods such as GCN [18], GIN [43], SAGE [11], etc. We present three instantiations of $\mathbf{AGG}$ used in our work in Appendix A.1. We also provide the general training routine for a GNN model in Appendix A.1 for reference.

**Scenarios**. There are mainly three kinds of supervised learning task: 1) *node classification* [19]; 2) *link prediction* [34]; 3) *graph classification* [46]. Depending on the graph, there are 1) *transductive* setting, where training nodes and testing nodes are on the same graph; 2) *inductive* setting, where training nodes and testing nodes are on different graphs, and the testing nodes are invisible during training.

## 2.2. Differential Privacy

**Definition 2.** *(Differential Privacy [10]) Given a data universe* $\mathcal{X}$*, two datasets* $X, X' \subseteq \mathcal{X}$ *are adjacent if they differ by only one data sample. A randomized algorithm* $\mathcal{M}$ *is* $(\varepsilon, \delta)$*-differentially private if for all adjacent datasets* $X$*,* $X'$ *and for all events* $S$ *in the output space of* $\mathcal{M}$*, we have* $\Pr(\mathcal{M}(X) \in S) \leq e^{\varepsilon} \Pr(\mathcal{M}(X') \in S) + \delta$.

The notion of the adjacent dataset $X$, $X'$ is context-dependent but is often ignored to be discussed. Technically, if $X'$ can be obtained by replacing a data instance of $X$, then it is called *bounded* DP [10]; if $X'$ can be obtained by addition/removal of a data sample of $X$, it is called *unbounded* DP [9]. Notably, these two notions are equivalent up to a factor of two [28]. Practically, to have a meaningful privacy guarantee, it is often to set $\varepsilon$ to be some small number, and the $\delta$ can be understood as the failure probability. DP has two notable properties: 1) immune to post-processing; 2) composition: running multiple DP algorithms sequentially also satisfies DP, and the overall privacy parameter can be bounded in terms of individual algorithm's parameters.

**Privacy accounting**. Privacy accounting gives the total privacy guarantee for the composition of several (adaptive) private algorithms. We introduce Rényi DP (RDP), a relaxation of DP based on Rényi divergence. RDP often serves as a basic analytical tool to analyze the composition of differentially private mechanisms while giving tight results.

**Definition 3** (Rényi DP [24])**.** *The Rényi divergence is defined as* $\mathcal{D}_\alpha(M||N) = \frac{1}{\alpha-1} \ln \mathbb{E}_{x\sim N}\left[\frac{M(x)}{N(x)}\right]^\alpha$ *with* $\alpha > 1$*. A randomized mechanism* $\mathcal{M} : \mathcal{X} \to R$ *is said to be* $(\alpha, \gamma)$*-RDP, if*

$$\mathcal{D}_\alpha(\mathcal{M}(X)||\mathcal{M}(X')) \leq \gamma$$

*holds for any adjacent dataset* $X, X'$.

**Theorem 1** (Composition by RDP, Theorem 20 [2])**.** *For* $\alpha > 1$ *and* $\delta > 0$*, and for a mechanism* $\mathcal{M}$ *which is* $(\alpha, \gamma)$*-RDP, the result for* $T$*-fold adaptive composition of* $\mathcal{M}$ *satisfies* $(\varepsilon, \delta)$*-differential privacy and*

$$\varepsilon = T\gamma + \log\frac{\alpha-1}{\alpha} - \frac{\log\delta + \log\alpha}{\alpha-1}.$$

Usually, one wants to convert an RDP guarantee to the $(\varepsilon, \delta)$-DP formulation. And Theorem 1 allows to make such conversion tightly. In practice, the final result is often obtained by optimizing $\varepsilon$ over $\alpha$.

## 2.3. Challenges & Existing Methods

**Challenges**. Compared to the popular privacy applications in learning on other types of data, solutions for *node-level* privacy in learning on graph data have hardly converged. In the following, we discuss some key challenges.

*1) New problem, few usable techniques.* The effectiveness of existing privacy protocols, including the well-known DP-SGD protocol, does not transfer to GNN training because of the behavior of the message-passing mechanism. At the algorithm level, to bound the sensitivity in DP learning on image, sequential, or tabular data, it suffices to quantify and limit the "impact" of *per-example* as data instances are independent. However, in a graph, nodes deliver impact to other nodes upon being aggregated by other nodes, and the counterpart as the *per-example* is ill-defined. In addition to the algorithmic challenge, we also face privacy accounting difficulties. Although current privacy accounting has reached tight results on the sub-sampled Gaussian mechanism [25], [27] in DP-SGD, unfortunately, it still cannot be applied because it is unknown how to analyze either the sampling rate for a node or the sensitivity. Settings do not fit. Challenges exist both at the algorithm level and privacy accounting level.

*2) How to ensure inference privacy in the transductive setting.* As mentioned before, in the transductive setting, test nodes' aggregation on neighbors can involve training (sensitive) nodes. Hence, the inference on testing nodes possibly leaks sensitive information [42]. A trivial countermeasure is to add calibrated noise during inference to ensure certain privacy guarantees. However, this is problematic: 1) adding noise affects the test accuracy; 2) for a fixed privacy budget, only a limited number of inferences/queries are allowed. Thus, this problem also remains unanswered.

**Existing methods**. There have been some notable attempts to tackle *node-level* privacy challenges in GNNs. We summarise them in the following.

*Not generalizable*. Olatunji et al. [29] proposed a solution based on the PATE approach [30]. However, such an approach only ensures privacy in the prediction process and needs many unlabeled public data for pre-training, making it impractical generally. Daigavane et al. [7] proposed a method (denoted as NDP) based on *gradient perturbation*. However, limited by the method, the noise added is overwhelming. Specifically, the reported results show that it

requires $\varepsilon \geq 15$ (or even $\varepsilon = 30$ in some cases) to make utility acceptable. Moreover, referring to the graph settings mentioned in Section 2.1, NDP fails to consider privacy issues during test/inference under transductive graph setting.

*Incorrect privacy analysis*. Due to the difficulties mentioned above, other works attempt a new type of approach, which is to privately derive the node/instance embeddings for each training node [33], [50]. The high-level idea is that once all embeddings for each node are derived privately, it is believed that nodes are decoupled from each other. Then, one can leverage existing privacy protocols (such as DP-SGD) to perform downstream private training. However, as will be proved in Section 6, there is a barrier in the utility-privacy trade-off. Specifically, it is impossible for such an approach to have both acceptable utility and strong privacy. We include a detailed case study in Section 6.2 on the analysis by [33], which has fundamental privacy analysis flaws.

## 3. Privacy Problem Formulation

In this study, we adopt the *unbounded* DP notion following previous work [1], [7], [33]. As pointed out in Section 2.3 (challenges), the interdependency between nodes due to message-passing is what differentiates *node-level* privacy problem from previous well-studied privacy problems. In the following, we discuss this issue.

**Data instances are independent in image datasets**. It is easy to find an interpretation/implication of DP for image datasets in a classification task: whether Alex chooses to contribute his photo or not, the final classifier will not be affected much. Note that the action taken by Alex never modifies other data instances in the dataset. This example represents the independent-data-instance assumption, which is widely used by previous work on privacy [6], [16], [22].

**Nodes are correlated in graph data**. However, it requires special care when dealing with graph datasets as nodes are connected. Depending on the scenarios, the action of "with" or "without" the differing node may modify data held by other nodes, and previous works have ignored this discussion.

To better understand our analysis, we reform the data into tabular form, *i.e.*, each row contains the information: $(a\ user's\ node\ vector, a\ user's\ out\text{-}edges)$. In a real-world application where Twitter (thus, out-edge information is who he/she follows) intends to learn a GNN model on the Twitter network to serve some recommendation purposes [45], in terms of what DP guarantees, there can be two typical scenarios shown in the following.

- **S1**: In this scenario, DP ensures that *whether a new user registers on Twitter or an existing user deletes/erases his/her account*, the final output model will not change much. Intuitively, this leads to that: *the private output makes it hard to infer whether a person is a Twitter user or not*. Consequently, registering or deleting modifies the data stored in other rows (altering the other rows' out-edges information), violating the independent-data-instance assumption. Technically, by Definition 2 of DP, this setting is equivalent to that the data universe $\mathcal{X}$ is all existing and incoming users.
- **S2**: Another guarantee is that *whether an existing user chooses to participate in the model training or not*, the final output model will not change much. Intuitively, this leads to that: *the private output makes it hard to infer whether a Twitter user ever participated in the study*. Consequently, even though Alex's row is used or never used, other users' data stays intact, as he has no right to force other users to follow or unfollow him. Technically, by Definition 2 of DP, this setting is equivalent to that the data universe $\mathcal{X}$ is all existing users.

Apparently, **S1** and **S2** represent different scenarios with different types of privacy guarantees. However, **S1** has some practical issues: 1) it has been argued by Kifer et al. [16] that it is not possible to have both privacy and utility if there is no assumption about the interdependency between data instances; moreover, a setting essentially analogous to **S1** is shown to provide poor utility-privacy trade-offs. 2) Based on the observation that adding/removing a row modifies data in multiple other rows in **S1**, we may also link this privacy guarantee to group privacy, *i.e.*, trying to maintain indistinguishability when a group of data instances changes. Note that a user can be followed by all other users, indicating the group size is $|\mathcal{G}|$ in the worst case, which is too ambitious to satisfy. 3) Although with **S1**'s privacy guarantee, knowing whether a person uses a prevalent social network like Twitter is probably not so informative (possibly not considered as leaking privacy) in some cases. If true, why bother to ensure DP in **S1**?

In contrast, **S2** does not violate the independent-data-instance assumption, and more importantly, **S2** models a more practical real-world privacy application in general. Our goal is to ensure DP in **S2**, and we define the *node-level* privacy for **S2** formally in the following.

**Definition 4** (Node-level Differential Privacy, formal statement for **S2**). *A private algorithm $\mathcal{L}$ is said to be node-level $(\varepsilon, \delta)$-DP with $\varepsilon > 0$ and $\delta \in (0, 1)$ if for any whole graph $\mathcal{G}$ and any pairs of adjacent graph $\mathcal{G}^* \subseteq \mathcal{G}, \mathcal{G}' \subseteq \mathcal{G}$ that differ by a node, and for all events $S \subseteq \mathbb{R}^d$, we have:*

$$\Pr(\mathcal{L}(\mathcal{G}^*) \in S) \leq e^{\varepsilon} \Pr(\mathcal{L}(\mathcal{G}') \in S) + \delta.$$

**Formal privacy model**. For any graph $\mathcal{G}$, we aim to ensure differential privacy as defined in Definition 4. In other words, we aim to protect the privacy of sensitive nodes (training nodes that have labels), including all of their node feature vector, edge information, and class information. W.o.l.g., suppose the differing node is $z$ and $\mathcal{G}^* \cup \{z\} = \mathcal{G}'$, information including 1) $z$'s feature vector, 2) label, and 3) in-edge $(i \to z) \in \mathcal{E}$, out-edge $(z \to i) \in \mathcal{E}, i \neq z$ will never be queried if $\mathcal{L}$ operates on $\mathcal{G}^*$. And a DP algorithm $\mathcal{L}$ must ensure the distributions of $\mathcal{L}(\mathcal{G}^*)$ and $\mathcal{L}(\mathcal{G}')$ are close.

Based on such, we can see that the privacy of the node's information we protect is just as strong as that in **S1**. Following how we model the targeted real-world privacy application, note that even though Alex's information is used

or never used, other users' following information will not be modified, as he has no right to force other users to follow or unfollow him when he participates or does not participate. Formally, this means that: in the whole graph $\mathcal{G}$, $\forall$ node $i$, the information of $i$'s in/out-degree stays unchanged upon Alex's choice as the whole graph $\mathcal{G}$ is fixed.

When social network users hesitate to participate in a study $\mathcal{L}$ that adopts GNN models, enforcing $\mathcal{L}$ is differentially private will significantly boost their trust and encourage participation. We focus on *node classification* GNN task, following previous work [7], [33]. We consider both inductive and transductive graph settings. We are targeting directed graphs, as it is more general if considering the connectivity (an undirected graph can be treated/processed as directed).

## 4. Our *Node-level* Privacy Solution

In this section, we first present the experiments that motivate the design of our solution. Then, we present our algorithmic solution with its privacy accounting.

### 4.1. Experiments Motivating Our Design

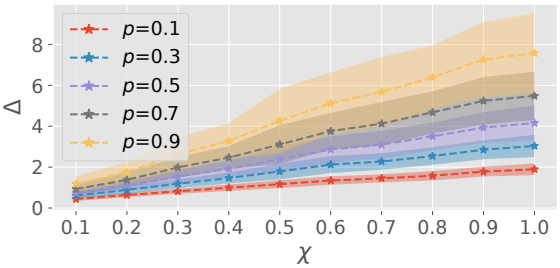



Figure 1: Empirical measurements on node $i$'s impact. Parameters are instantiated as $C = 10$, $n = 100$. The experiment runs 100 times with different seeds.

**Tracing the source: impact measurements**. Unlike other data forms, it is not straightforward to see how a node impacts the GNN model update. Nevertheless, we can still get some clues by analyzing the message-passing behavior: node $i$'s information is also propagated to node $j$ if $(i \to j) \in \mathcal{E}$. Each node $i$ influences other nodes by its out-edges, and this leads to our hypothesis that *the greater* $|\{(i \to j) \in \mathcal{E}\}|$ *is, the more impact node* $i$ *has on the GNN model update.* In what follows, we conduct experiments to measure node $i$'s impact and verify our hypothesis empirically:

1) Initialize a base graph $\mathcal{G}^*$ with $n$ nodes; each node's feature vector is $d$-dimension and is sampled from the standard normal distribution; each node's label is uniformly sampled from $\{1, 2, \cdots, C\}$; connecting edges using the Erdős-Rényi model [4] (simply speaking, it connects edges between each pair of nodes independently with some probability $p$).
2) Initialize a GNN model $w$ and compute the GNN model's gradient $g^* = \nabla f(\mathcal{G}^*; w)$.
3) Create a new node $i \notin \mathcal{G}^*$ by initializing its feature vector and class in the same way as before; forming $\chi \cdot n$ (where $\chi$ is a parameter we vary) out-edges from $i$ to different random nodes in $\mathcal{G}^*$; denote the resultant graph as $\mathcal{G}'$; compute the GNN model's gradient $g' = \nabla f(\mathcal{G}'; w)$.
4) Record $\Delta = \|g^* - g'\|_2$. Note that $g^*$ and $g'$ are the results corresponding to the pair of adjacent graphs. Finally, $\Delta$ serves as the proxy for node $i$'s impact on the model update.

**Experimental results**. The experimental results are provided in Figure 1. We can see that, as expected, when $\chi$ increases (more out-edge $i$ has), $\Delta$ becomes larger (node $i$ has more impact). This can be intuitively explained: the more out-edge $i$ has, the more "channels" through which the impact is delivered.

**Lessons learned**. Translating to technical terms, by definition, one node's impact on the output in maximum is exactly the ($\ell_2$-)sensitivity of the query function. As shown in [1], [24], the calibrated DP noise is proportional to $\ell_2$ sensitivity, and the final utility one can get is clearly inverse-related to how much DP noise is added. Hence, to improve the utility, we argue limiting one node's impact or preventing excessive impact on the output is a basic and natural countermeasure. And this leads to our sampling strategy, as will be shown.

**Our finding explains why previous works make a particular assumption**. Knowing how one instance affects the query function's output leads to clues to control the sensitivity. In our observations, one node's impact is roughly proportional to its out-edges. This coincides with previous work on *node-level* privacy in GNNs that they assume nodes' degree is bounded by some quantity D, which is assumed to be *much smaller* than the theoretical maximum degree of the graphs for utility reasons [7], [33]. Using our observation to explain such: bounded degree assumption leads to bounded node's impact. To ensure the bounded degree assumption, previous work [7], [33] adopts graph projection to erase some edges between nodes. However, we argue that trimming the graph then becomes part of their algorithms, and to avoid privacy leakage, this operation should be treated with extra caution because it is sensitive to one node's presence.

**Our design**. In our work, contrastingly, there is no assumption of bounded degree, which sidesteps the above issues. Based on the above experiments, we form a new idea to regulate one node's impact: if one node has larger out-degrees, we will lower the probability for that node to be sampled, offsetting the node's original impact.

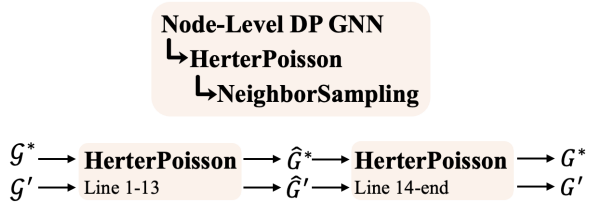



Figure 2: Our algorithm structure and the workflow of *HeterPoisson* applied to adjancent graph $\mathcal{G}^*$ and $\mathcal{G}'$. We also highlight the critical stages in *HeterPoisson*.

**Algorithm 1** Node-Level DP GNN

**Input:** Whole graph $\mathcal{G}$, input graph $\hat{\mathcal{G}} \subseteq \mathcal{G}$, initial model $w^0$, number of iteration $T$, learning rate $\eta$, noise s.t.d. $\sigma$, base sampling rate $q_b$, multiplier $M$, loss function $f(;)$

**for** $t = 1, 2, \cdots, T$ **do**
  $G \leftarrow \underline{\textbf{HeterPoisson}}(\mathcal{G}, \hat{\mathcal{G}}, q_b, M)$ ▷ Algorithm 2
  **for** $G_i \in G$ **do in parallel**
    $g_i \leftarrow \nabla f(G_i; w^{t-1})$ ▷ Per-sub-graph gradient
    $\hat{g}_i \leftarrow g_i \cdot \textbf{min}\{1, \frac{1}{2\|g_i\|}\}$ ▷ Clip with Threshold 0.5
  **end for**
  $\bar{g}^t \leftarrow \sum \hat{g}_i$ ▷ Non-private output
  $g^t \leftarrow \bar{g}^t + \mathcal{LAP}(0, \sigma\mathbb{I}^d)$ ▷ Private output
  $w^t \leftarrow w^{t-1} - \eta g^t$ ▷ Model update, post-processing
**end for**

**Output:** learned model $w^T$

**Algorithm 2** HeterPoisson$(\mathcal{G}, \hat{\mathcal{G}}, q_b, M)$

**Input:** Whole graph $\mathcal{G}$, input graph $\hat{\mathcal{G}} \subseteq \mathcal{G}$, base sampling rate $q_b$, multiplier $M$

$G \leftarrow \emptyset$ ▷ Initialize sub-graph batch container
**for** each node $i$ in $\hat{\mathcal{G}}$ **do**
  ▷ Forming each sub-graph $G_i$ independently
  $p \leftarrow \text{Uniform}(0, 1)$
  **if** $p < q_b$ **then**
    ▷ Sampling neighbors of node $i$
    $N_i \leftarrow \underline{\textbf{NeighborSampling}}(\mathcal{G}, \mathcal{NB}(i), M)$ ▷ Alg. 3
    Mark $i$ as *central* node and $N_i$ as *peripheral* nodes
    Forming the induced sub-graph $G_i$ using $i$ and $N_i$
    $G.add(G_i)$ ▷ Add this sub-graph to the container
  **end if**
**end for**
▷ Ensuring no *peripheral* node can be a *central* node
$\Delta \leftarrow$ all *central* nodes sampled
**for** $G_i$ in $G$ **do**
  **for** $j$ in ***peripheral*** **nodes** of $G_i$ **do**
    **if** $j$ is in $\Delta$ **then**
      ▷ modify this node to be **NULL**
      modify the feature vector of $j$ to be zero in $G_i$
    **end if**
  **end for**
**end for**

**Output:** sub-graph batch $G$

**Algorithm 3** NeighborSampling$(\mathcal{G}, I, M)$

**Input:** Whole graph $\mathcal{G}$, node set $I$, multiplier $M$

$I_s \leftarrow \emptyset$
**for** $i$ in $I$ **do**
  $p \leftarrow \text{Uniform}(0, 1)$
  $D^i_{ot} \leftarrow i$'s out-degree in $\mathcal{G}$
  **if** $p < M/D^i_{ot}$ **then**
    $I_s.add(i)$
  **end if**
**end for**

**Output:** Sampled index set $I_s$

## 4.2. Algorithmic Solution

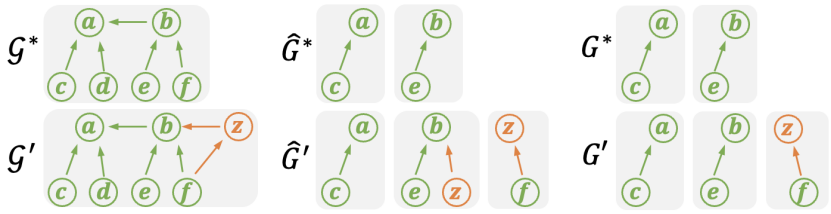
**Algorithm overview**. We present the illustration for the structure of our algorithm in Figure 2, showing how functions call the others. We also highlight the critical stages in *HeterPoisson* (Algorithm 2) in Figure 2. In Figure 3, we show toy examples of critical stages happening in *HeterPoisson* as highlighting those helps better understand our privacy analysis. For either of the adjacent input graphs $\mathcal{G}^*, \mathcal{G}'$, *HeterPoisson* first receives input graph $\hat{\mathcal{G}}$; then, after many sub-graphs are sampled and formed, we get sub-graph container $\hat{G}$; finally, after those sub-graphs are processed, we get the final sub-graph container $G$.

**Algorithm 1**. The high-level steps of our solution are presented in Algorithm 1. In each iteration, *Node-Level DP GNN* first calls function *HeterPoisson* to return a sub-graph container $\hat{G}$ which contains many sub-graphs. This allows us to leverage the ideas from DP-SGD, *i.e.*, separate-then-bound; we treat a single sub-graph as the *per-example*, and we clip the per-sub-graph gradient with bounded $\ell_2$-norm. Finally, we add symmetric multivariate Laplace (SML) noise $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ [21] to the sum of clipped gradient vectors to form the private gradient. We elaborate $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ in Section 4.3. Roughly speaking, in our algorithm framework, SML leads to both non-trivial privacy bound and non-trivial utility because its tail behaves as desired. In contrast, the well-known Gaussian cannot achieve such. This is explained by our discussion in Section 4.3,

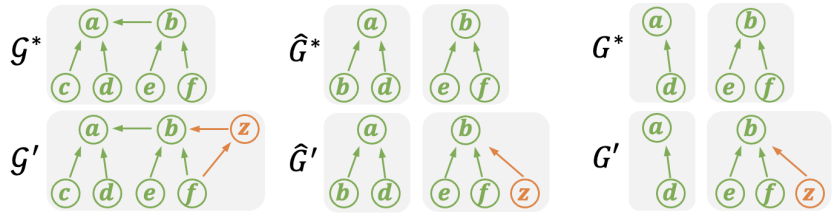
**Algorithm 2**. *HeterPoisson* is the core part of our solution. *HeterPoisson* returns a container that contains many sub-graphs. It first samples the *central* nodes, and based on a *central* node, it calls the function *NeighborSampling* to sample neighbors to form a sub-graph. Note that sampling on *central* nodes are independent trails. After forming many sub-graphs, it ensures no *central* node appears in the *peripheral* nodes in other sub-graphs. We enforce such an operation for utility purposes. Specifically, if we do not have such an operation, although our privacy accounting method can still apply, we need to add more noise as the sensitivity increases due to overlapping nodes among these sub-graphs, which hurts utility. Discussion and experimental results for this study are provided in Section 5.2.

**Algorithm 3**. Similar to the sampling behavior on *central* nodes, *NeighborSampling* also enforces independent trials when sampling on the neighbors of the *central* nodes. For each neighbor, the sampling ratio is adjusted to be inverse-proportional to this neighbor's out-degree in the whole graph $\mathcal{G}$, such that the neighbor's expected number of being sampled as some *peripheral* nodes is $\frac{M}{D^i_{ot}} \times D^i_{ot} = M$. This idea for controlling the impact of nodes comes from our previous motivating experiments in Section 4.1. On the other hand, we can control $M$ to balance the privacy and utility trade-off. Note that we enforce independent trials, just like when we sample the *central* nodes; this specific operation enables a tractable privacy analysis in Section 4.3.

**Complexity analysis**. We only elaborate on the additional computational complexities brought by *HeterPoisson*. Notably, in line 15 to 22 in Algorithm 2, the expected running time for the outer "for" loop is $\mathcal{O}(q_b|\mathcal{V}|)$, and the expected running time for the inner "for" loop is essentially the expected number of *peripheral* sampled, which



(a) Differing node $z$ is sampled as a *central* node



(b) Differing node $z$ is not sampled as a *central* node

Figure 3: Critical stages in *HeterPoisson* are highlighted out; we have $\mathcal{G}' = \mathcal{G}^* \cup \{z\}$ where $z$ is the differing node. (a) represents the case when the differing node $z$ is sampled as a *central* node; finally, sub-graph container $G'$ **has only 1 more sub-graph than** $G^*$. (b) represents the case when $z$ is not sampled, $k$ out-pointing nodes of $z$ is sampled (in the above figure, $k = 1$), and all of those $k$ nodes also sample $z$ as neighbors; finally, sub-graph container $G^*, G'$ **differ in** $k$ **sub-graphs**, *i.e.*, there are only $k$ sub-graphs in $G^*$ differing from another $k$ sub-graphs in $G'$.

is $\mathbb{E}_{i\in\mathcal{V}}(\sum_{(j\to i)\in\mathcal{E}} \frac{M}{D^j_{ot}}) = M$ where $D^j_{ot}$ is out-degree of node $j$. Hence the expected running time in line 15 to 22 in Algorithm 2 is $\mathcal{O}(q_b|\mathcal{V}|M)$, which is the additional expected computational complexity brought by Algorithm 2 in each iteration. Note that this is a conservative analysis; in practice, we can parallelize both the outer and inner "for" loops, which makes the additional running time almost negligible.

**Discussion on *HeterPoisson***. As suggested by our complexity analysis, our method scales linearly/mildly with the training graph's size ($\mathcal{V}$, number of nodes), making it practical for applications in large graphs. Our method does not assume the specific method of GNN aggregation but only follows the general abstracted aggregation in Definition 1. This makes it generalizable to other GNN instances, given alignments with Definition 1. Our method assumes the graph is static, i.e., the graph's nodes/edges are fixed during the training; for other types of graphs, such as dynamic or heterogeneous graphs, our method may not directly apply as the privacy model (e.g., the formulation of adjacent graph dataset pair) is quite different. Substantial additional work may be required to ensure node-level privacy in such graphs.

### 4.3. Privacy Accounting

In this part, we prove the privacy guarantee of our solution. As we mentioned, we use symmetric multivariate Laplace (SML) random noise instead of Gaussian because such noise better suits our solution and leads to a non-trivial privacy guarantee. From first principles, *any random noise can be leveraged to ensure differential privacy as long as it suits the algorithmic operations*. We begin by defining a special family of random noise distributions.

**Definition 5** (Spherical Distribution)**.** *A $d$-dimension distribution $\mathcal{B}(m, \sigma^2\mathbb{I}^d)$ with mean $m \in \mathbb{R}^d$ and diagonal covariance matrix $\sigma^2\mathbb{I}^d$ ( each coordinate has s.t.d. $\sigma$) is called spherical if its probability density function (PDF) $p(x)$ satisfies $\forall a, b \in \mathbb{R}^d$, $p(a) = p(b)$ if $\|a - m\|_2 = \|b - m\|_2$.*

**Examples**. Spherical distribution is a special case of elliptical distribution, and it emphasizes the phenomenon that *the density is rotational invariant*, *i.e.*, the contour of the density function is $\ell_2$-norm balls. Isotropic Gaussian $\mathcal{N}(0, \sigma^2\mathbb{I}^d)$ is spherical. The ***symmetric multivariate Laplace*** (SML) distribution $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ [21], whose density function given by

$$g(\mathbf{x}) = \frac{2}{(2\pi\sigma^2)^{d/2}} \left(\frac{\|\mathbf{x}\|_2^2}{2\sigma^2}\right)^{v/2} K_v\left(\sqrt{\frac{2\|\mathbf{x}\|_2^2}{\sigma^2}}\right)$$

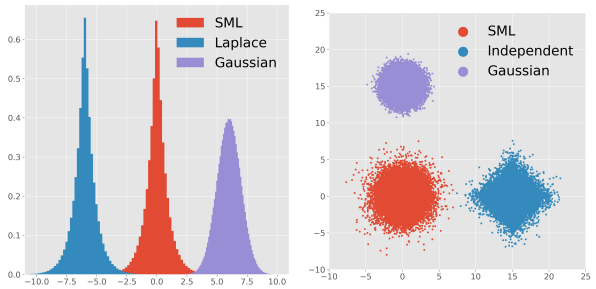
is also spherical ($v = \frac{2-d}{2}$ and $K_v$ is the modified Bessel function of the second kind). It is self-evident that the density is rotational invariant. We will leverage $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ as the random noise.

**Remark**. Technically, the marginal distribution ($d = 1$) of $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ is Laplace, however, $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ itself ($d > 1$) is not coordinate-wise independent although covariances are zeros. As a contrasting example, the vector of coordinate-wise independent univariate Laplace *is different* from $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$, and it is not spherical, as shown in Figure 4b.

**Algorithm 4** Sample $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$

**Input:** Noise s.t.d. $\sigma$, dimension $d$
1: $Z \sim \mathcal{N}(0, \sigma^2\mathbb{I}^d)$ ▷ Sample from multivariate Gaussian
2: $W \sim \mathbf{Exp}(1)$ ▷ Exponential distribution with mean 1
**Output:** $\sqrt{W}Z$

**Sampling SML**. We can sample a SML distribution $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ easily by simulation [21] as shown in Algorithm 4. We illustrate in Figure 4: 1) in Figure 4a,




(a) Histograms (b) Scattering plot

Figure 4: Distribution illustration. In (a), we show the histogram for 1-dimension case for 1) SML distribution, 2) directly sampled Laplace distribution ("Laplace"), and 3) Gaussian distribution. The vertical axis is the density. In (b), we show the 2-dimension case scattering plot. We can see that the contours of the SML and Gaussian are circles; in contrast, for coordinate-wise independent Laplace ("independent"), the contours show a square shape ($\ell_1$-norm ball). All distributions have the same coordinate-wise s.t.d. 1, and their means/centers are shifted from each other for better visualization; $10^6$ samples are drawn for each.

the simulated version under the 1-dimension case perfectly matches the directly sampled univariate Laplace (marginal distribution of $\mathcal{LAP}$); 2) in Figure 4b, although with the same covariance matrix $\sigma\mathbb{I}^d$, SML is more "spread out" (heavier-tail) than Gaussian.

Before introducing our main theorem, we first introduce Lemma 1 and 2 that serve our purpose.

**Lemma 1.** *Consider a function $f : \mathcal{X} \to \mathbb{R}^d$ that has $k$ $\ell_2$-sensitivity on adjacent datasets $X, X'$, i.e., $\max_{X,X'} \|f(X) - f(X')\|_2 = k$ and a private mechanism $\mathcal{M} : \mathcal{X} \to \mathbb{R}^d$ that adds spherical random noise $\mathcal{B}(0, \sigma^2\mathbb{I}^d)$ to $f(x)$, i.e., $\mathcal{M}(X) = f(X) + \mathcal{B}(0, \sigma^2\mathbb{I}^d)$, the Rényi divergence between $\mathcal{M}(X)$ and $\mathcal{M}(X')$ satisfies:*

$$\mathcal{D}_\alpha(\mathcal{M}(X)||\mathcal{M}(X')) \leq \mathcal{D}_\alpha(\mathcal{B}(k, \sigma^2)||\mathcal{B}(0, \sigma^2)),$$

*i.e., it suffices to consider the one-dimension case.*

Proof is provided in Appendix B.1. The above lemma tells us if the random noise we add is spherical and the $\ell_2$-sensitivity is bounded, it suffices to compute the Rényi divergence for the one-dimension case because the density of noise distribution is rotation-invariant.

**Lemma 2** (RDP upper bound for a pair of mixture distributions)**.** *Let $\mu_\rho = \mathbb{E}_{k\sim\rho}[\mu_k] = \sum_k \Pr_\rho(k)\mu_k$ be a mixture distribution constructed from individual distributions $\mu_k$ where the index random variable $k$ follows distribution $\rho$. Similarly, define $\xi_\rho = \mathbb{E}_{k\sim\rho}[\xi_k] = \sum_k \Pr_\rho(k)\xi_k$, then*

$$e^{(\alpha-1)\mathcal{D}_\alpha(\mu_\rho||\xi_\rho)} \leq \mathbb{E}_{k\sim\rho} e^{(\alpha-1)\mathcal{D}_\alpha(\mu_k||\xi_k)}. \tag{1}$$

Proof is provided in Appendix B.2. Lemma 2 allows us to obtain an RDP upper bound for a pair of mixture distributions if each pair's RDP bounds are known. We then have the following privacy accounting result.

**Theorem 2** (Main Privacy Result)**.** *Algorithm 1 satisfies $(\alpha, \gamma)$-RDP, where $\gamma$ satisfies*

$$\gamma \leq \max_{D_{ot}\in[|\mathcal{G}|-1]} \frac{T}{\alpha-1} \ln \mathbb{E}_{k\sim\rho} \left[ \underbrace{\frac{\alpha e^{\sqrt{2}(\alpha-1)k/\sigma}}{2\alpha-1} + \frac{1}{2}}_{B_k} \right]. \tag{2}$$

*where $[n] = \{0, 1, 2, \cdots, n\}$, and $k$ can be treated as a quantity measuring sensitivity. The PMF of random variable $k$ from distribution $\rho$ is:*

$$\Pr_\rho(k; D_{ot}) = \begin{cases} (1-q_b)\mathbf{Bi}(k; D_{ot}, \frac{q_b M}{D_{ot}}), & \text{for } k \in [D_{ot}] \\ q_b, & \text{for } k = 0.5 \end{cases}$$

*where $\mathbf{Bi}$ is the PMF of a binomial distribution, i.e., $\mathbf{Bi}(k; n, p) = \binom{n}{k} p^k(1-p)^{n-k}$. For the case $D_{ot} = 0$, define $\Pr_\rho(0) = 1 - q_b$ and $\Pr_\rho(0.5) = q_b$.*

Proof is provided in Appendix B.3. The main idea of proving Theorem 2 is first to consider several cases as shown in Figure 3 where we can equivalently transform the RDP bound computation in one-dimension case as allowed by Lemma 1, then we derive an RDP guarantee by Lemma 2. Finally, if needed, applying Theorem 1 to the above result leads to the $(\varepsilon, \delta)$-DP guarantee.

**Why SML**? In principle, many randomness may be potentially leveraged to ensure some level of DP. Although the Gaussian mechanism (ensuring DP by adding Gaussian noise) is probably the most well-known privacy-preserving approach, we argue that it does not fit our method *Heter-Poisson* as Gaussian is not able to achieve non-trivial privacy and non-trivial utility simultaneously. Technically, this is because of its tail behaviors. We explained in the following.

To have a non-trivial privacy bound, we need to ensure the $\gamma$ upper-bound in Equation (2) is reasonably small for some fixed $\alpha$. Note that we have an expectation calculation over many $B_k$ terms in Equation (2). This means that each term

$$A_k = \Pr_\rho(k)B_k = \Pr_\rho(k)\left(\frac{\alpha e^{\sqrt{2}(\alpha-1)k/\sigma}}{2\alpha-1} + \frac{1}{2}\right) \tag{3}$$

must be small enough for each $k \in [D_{ot}]$ and $D_{ot}$ can be as large as $|\mathcal{G}|-1$ (once $D_{ot}$ is fixed, we omitted the notation of $D_{ot}$). A first intuition is that when $B_k$ is exponentially large, $\Pr_\rho(k)$ must be at least exponentially small. Moving $\Pr_\rho(k)$ to the exponent, as $\alpha$ is fixed and other terms are small, we only need to investigate $e^{C_k} = e^{\sqrt{2}(\alpha-1)k/\sigma + \ln \Pr_\rho(k)}$. For $D_{ot} > 0, k > 1$, we have

$$\begin{aligned} \ln \Pr_\rho(k) &\leq \ln \frac{D_{ot}!}{k!(D_{ot}-k)!} - k \ln \frac{D_{ot}}{q_b M} \\ &= -\ln k! - k \ln \frac{1}{q_b M} + \ln \frac{D_{ot}!}{(D_{ot}-k)!(D_{ot})^k} \\ &\leq -\ln k! - k \ln \frac{1}{q_b M} \end{aligned} \tag{4}$$

hence:

$$C_k \leq \frac{\sqrt{2}(\alpha-1)}{\sigma} k - \ln k! - k \ln \frac{1}{q_b M} \tag{5}$$

As $\ln(k!) = k\ln k - k + \mathcal{O}(\ln k)$, the right-hand side of Equation (5) always decreases asymptotically. It is also easy to see that the right-hand side of Equation (5) strictly decreases w.r.t. $k$ if $(\alpha-1)/\sigma$ is small enough. In practice, we often optimize the final $(\varepsilon, \delta)$-DP result over $\alpha \in (1, 10]$, and the noise scale $\sigma$ can also be small enough without making $A_k$ large, for example, $\sigma \leq 10$ will suffice. Therefore, we can indeed have non-trivial privacy (small enough $\gamma$) and non-trivial utility (small enough $\sigma$) simultaneously.

**Why not Gaussian**? Things are very different when the noise is Gaussian. For fixed $\sigma$, $A_k$ overflows when $k$ is a large integer (*e.g.*, $k = |\mathcal{G}| - 1$) if the noise is Gaussian. In such case,

$$A_k = \Pr_\rho(k) B_k = e^{\alpha(\alpha-1)k^2/(2\sigma^2) + \ln \Pr_\rho(k)}.$$

As we can see the $k$ term is quadratic instead of linear in the first term in the exponent. This will lead to trivial results, as we must set the noise $\sigma$ exceedingly large to make each $A_k$ small. Large noise $\sigma$ means little utility. To provide some numerical examples, suppose $|\mathcal{G}| = 10^4+1, q_b = 0.01, M = 1, \alpha = 2, \sigma = 1$ for SML, then, to reach the same $A_{10^4}$ value if using Gaussian, we need $\sigma \approx 84$. The learning may not even converge with noise having such large s.t.d.. Roughly, if using Gaussian, the noise s.t.d. $\sigma$ needs to scale up by $\Theta(\sqrt{|\mathcal{G}|})$ factor to reach the same $A_{|\mathcal{G}|}$ as that of SML. This means that if using Gaussian, $\sigma$ becomes even worse when the graph has more nodes. This phenomenon is confirmed by our experiment in Section 5.2.

Thus, in our algorithmic framework, we claim that the algorithm (determining $\Pr_\rho(k)$) and the random noise (determining $B_k$) should "cooperate" well to have both non-trivial privacy (small enough $\gamma$) and non-trivial utility (small enough $\sigma$). Due to using SML random noise $\mathcal{LAP}(0, \sigma^2\mathbb{I}^d)$ and the special sampling strategy in *HeterPoisson*, our solution achieves this goal.

**Discussion**. We have seen a seemingly *"amplification"* effect in privacy, *i.e.*, with the same coordinate-wise noise s.t.d. $\sigma$, in the exponent of $A_k$, the divergence due to noise contributes $k$ linear if the random noise is SML, instead of quadratic in $k$ if using Gaussian noise. This is because the divergence computation includes tails, and SML's tail is heavier, hence leading to a privacy boost in the upper bound. Meanwhile, we must emphasize that *this privacy boost does have a cost theoretically*: although the coordinate-wise s.t.d. is the same for SML and Gaussian, SML decay slower than Gaussian, *i.e.*, SML are more "spread out" as shown in Figure 4b, thus potentially having more negative impact on the utility. However, this is not a significant concern as the probability of sampling some extreme noise value (tail probability) is still exponentially small. In this sense, SML amplifies privacy. As will also be evaluated in real-world datasets, Gaussian only leads to trivial results, and our choice works in contrast.

**Extension**. In principle, we can also claim that as long as the noise is spherical and decays no faster than Laplace, it can also be leveraged in our method to have a non-trivial privacy guarantee because $A_k$ will be small. This is of independent interest and can be useful to other privacy applications. However, this introduces new complexities in the divergence computation and requires additional evaluation of its performance; hence, we choose SML in this study.

### 4.4. Preserve Privacy at Inference

Another important functionality of *HeterPoisson* is to ensure inference privacy at test time under the transductive setting as discussed in Section 3. We have shown that adding noise during inference is problematic, and to address this issue, we devise a simple yet effective approach. Given all of our previous work, it only requires modifying one line in our algorithm: during test/inference, in line 7 in Algorithm 2, instead of passing the whole neighboring nodes of the current test nodes as the function argument, we only pass the neighboring nodes which are not training nodes, *i.e.*, test nodes can only sample nodes within the test nodes set, not from the training nodes set. That is, sensitive information is insulated from being accessed at test time, hence no more privacy concerns.

| | Facebook | Twitch | Amazon | PubMed | Reddit |
|---|---|---|---|---|---|
| # nodes | 22K | 9K | 13K | 19K | 232K |
| # edges | 342K | 315K | 491K | 88K | 114M |
| Avg. # edges | 15 | 33 | 35 | 5 | 492 |
| Features | 128 | 128 | 767 | 500 | 602 |
| Classes | 4 | 2 | 10 | 3 | 41 |

TABLE 1: Dataset statistics

## 5. Method Evaluation

**Organization**. We first show the performance of our method by comparing it with various approaches; we also provide ablations studies on our design choices. Finally, we 1) apply current privacy audit approaches to test the privacy integrity of our method and 2) perform membership inference attacks by a strong adversary to test our protocol's resilience. Experimental setups are presented in Appendix A.2. Our implementation is in an anonymous link[1].

**Datasets**. We test our method on five datasets with various properties: Facebook [32], Twitch [32], Reddit [11], Amazon [35], and PubMed [44]. The first three are related to social network applications where node/user privacy is concerned; the other two datasets with different properties are included for completeness. The summary for these datasets is presented in Table 1. We test our approach under both transductive and inductive settings.

### 5.1. Performance Comparison

In Table 2, we first observe that the *naive DP-SGD* fails to achieve satisfying utility, although such an approach is straightforward. To re-describe the *naive DP-SGD* approach, it only takes the node feature vector as the input, and it treats one feature vector just as it treats an image in the well-studied DP-SGD applications on image data [1]. These experimental results show that the effectiveness of DP-SGD

1. https://github.com/zihangxiang/PNPiGNNs.git

| $\varepsilon$ | GCN | | | | | GIN | | | | | SAGE | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Facebook | Twitch | Amazon | PubMed | Reddit | Facebook | Twitch | Amazon | PubMed | Reddit | Facebook | Twitch | Amazon | PubMed | Reddit |
| 2 | $48.6_{\pm 3.7}$ | $59.0_{\pm 2.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $41.9_{\pm 1.6}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.3_{\pm 0.9}$ | $55.9_{\pm 0.3}$ | $34.9_{\pm 0.9}$ | $41.0_{\pm 1.2}$ | $25.4_{\pm 0.3}$ | – | – | – | – | – | – | – | – | – | – |
| | $19.1_{\pm 0.1}$ | $13.7_{\pm 0.5}$ | $24.7_{\pm 0.4}$ | $38.8_{\pm 0.3}$ | $39.2_{\pm 0.3}$ | $32.7_{\pm 0.9}$ | $4.9_{\pm 0.9}$ | $4.6_{\pm 0.9}$ | $32.0_{\pm 0.5}$ | $35.3_{\pm 0.3}$ | $19.1_{\pm 0.6}$ | $2.3_{\pm 0.4}$ | $22.1_{\pm 0.9}$ | $39.9_{\pm 0.9}$ | $35.4_{\pm 0.4}$ |
| | **73.5**$_{\pm 0.9}$ | **65.4**$_{\pm 0.7}$ | **74.8**$_{\pm 2.1}$ | **77.1**$_{\pm 0.4}$ | **72.3**$_{\pm 0.9}$ | **73.5**$_{\pm 0.3}$ | **65.3**$_{\pm 0.8}$ | **73.8**$_{\pm 1.3}$ | **81.4**$_{\pm 0.4}$ | **65.4**$_{\pm 0.9}$ | **74.2**$_{\pm 0.8}$ | **65.5**$_{\pm 1.1}$ | **74.3**$_{\pm 1.0}$ | **79.8**$_{\pm 0.2}$ | **69.2**$_{\pm 1.1}$ |
| 4 | $51.0_{\pm 3.6}$ | $60.2_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $42.7_{\pm 1.5}$ | – | – | – | – | – | – | – | – | – | – |
| | $35.2_{\pm 0.4}$ | $59.0_{\pm 0.4}$ | $30.9_{\pm 0.6}$ | $34.8_{\pm 0.4}$ | $25.5_{\pm 0.9}$ | – | – | – | – | – | – | – | – | – | – |
| | $48.5_{\pm 0.2}$ | $32.7_{\pm 0.4}$ | $30.5_{\pm 0.9}$ | $37.4_{\pm 0.5}$ | $50.9_{\pm 0.9}$ | $47.2_{\pm 0.4}$ | $32.2_{\pm 0.9}$ | $13.4_{\pm 0.9}$ | $39.2_{\pm 0.4}$ | $45.1_{\pm 0.9}$ | $38.4_{\pm 0.3}$ | $15.1_{\pm 0.5}$ | $22.9_{\pm 0.2}$ | $40.5_{\pm 0.9}$ | $49.2_{\pm 0.2}$ |
| | **74.5**$_{\pm 1.0}$ | **66.0**$_{\pm 0.6}$ | **79.6**$_{\pm 0.2}$ | **78.4**$_{\pm 0.3}$ | **79.1**$_{\pm 0.7}$ | **74.8**$_{\pm 0.3}$ | **65.9**$_{\pm 0.8}$ | **79.1**$_{\pm 0.7}$ | **82.9**$_{\pm 0.3}$ | **74.6**$_{\pm 0.8}$ | **74.7**$_{\pm 0.6}$ | **66.4**$_{\pm 0.6}$ | **79.2**$_{\pm 0.5}$ | **80.8**$_{\pm 0.3}$ | **76.0**$_{\pm 0.9}$ |
| 8 | $52.1_{\pm 3.0}$ | $60.4_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $43.4_{\pm 1.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.1_{\pm 1.2}$ | $56.7_{\pm 0.9}$ | $36.1_{\pm 0.9}$ | $33.3_{\pm 0.3}$ | $29.1_{\pm 0.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $56.9_{\pm 0.4}$ | $61.6_{\pm 0.2}$ | $27.9_{\pm 0.6}$ | $38.3_{\pm 0.3}$ | $62.2_{\pm 0.6}$ | $64.9_{\pm 0.4}$ | $41.8_{\pm 0.9}$ | $26.1_{\pm 0.9}$ | $47.7_{\pm 0.4}$ | $54.9_{\pm 0.4}$ | $52.7_{\pm 0.3}$ | $56.8_{\pm 0.3}$ | $32.0_{\pm 0.9}$ | $39.3_{\pm 0.4}$ | $62.4_{\pm 0.5}$ |
| | **74.6**$_{\pm 1.1}$ | **66.4**$_{\pm 0.4}$ | **82.1**$_{\pm 0.6}$ | **79.1**$_{\pm 0.2}$ | **83.2**$_{\pm 0.3}$ | **75.8**$_{\pm 0.7}$ | **66.1**$_{\pm 0.8}$ | **82.8**$_{\pm 0.3}$ | **84.4**$_{\pm 0.5}$ | **80.6**$_{\pm 0.3}$ | **75.2**$_{\pm 0.6}$ | **66.9**$_{\pm 0.3}$ | **82.0**$_{\pm 0.8}$ | **81.6**$_{\pm 0.5}$ | **81.1**$_{\pm 0.8}$ |
| 16 | $52.9_{\pm 2.7}$ | $60.4_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $44.0_{\pm 1.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $32.6_{\pm 0.1}$ | $56.9_{\pm 0.9}$ | $34.6_{\pm 1.2}$ | $31.4_{\pm 0.4}$ | $34.8_{\pm 1.2}$ | – | – | – | – | – | – | – | – | – | – |
| | $70.3_{\pm 0.1}$ | $63.7_{\pm 0.2}$ | $41.2_{\pm 0.9}$ | $40.3_{\pm 0.9}$ | $67.9_{\pm 0.3}$ | $75.2_{\pm 0.9}$ | $50.5_{\pm 0.9}$ | $38.9_{\pm 0.9}$ | $52.0_{\pm 0.9}$ | $59.9_{\pm 0.9}$ | $67.3_{\pm 0.1}$ | $64.4_{\pm 0.4}$ | $32.8_{\pm 0.3}$ | $39.6_{\pm 0.9}$ | $69.9_{\pm 0.9}$ |
| | **74.9**$_{\pm 1.3}$ | **66.6**$_{\pm 0.5}$ | **83.1**$_{\pm 0.5}$ | **79.2**$_{\pm 0.5}$ | **85.8**$_{\pm 0.2}$ | **76.5**$_{\pm 0.7}$ | **66.4**$_{\pm 0.5}$ | **84.2**$_{\pm 0.6}$ | **84.9**$_{\pm 0.5}$ | **84.1**$_{\pm 0.3}$ | **75.5**$_{\pm 0.4}$ | **66.9**$_{\pm 0.3}$ | **83.8**$_{\pm 1.1}$ | **82.1**$_{\pm 0.6}$ | **84.1**$_{\pm 0.4}$ |

TABLE 2: Classification accuracy. In each cell, from top to bottom, the result is *naive DP-SGD*, GAP [33], NDP [7], and **our method**. Best results are shaded in gray. As *naive DP-SGD* and GAP are not related to any GNN model, their results are only presented in the "GCN" column.

does not transfer to GNN, and ignoring edge information is sub-optimal. In the following comparison, we aim to show that our solution also has a significant advantage over existing baselines. Results are also presented in Table 2.

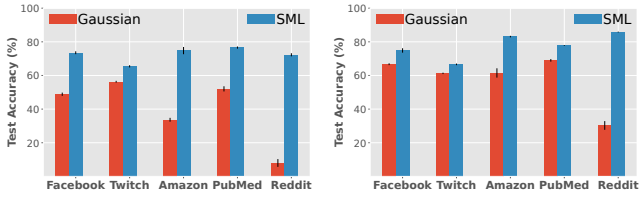
One notable work included for comparison is GAP [33]. As discussed in Section 6 and Section 6.2, GAP significantly underestimated the DP noise needed in its aggregation stage. Therefore, we correct the amount of DP noise in this experiment. We can observe that our solution outperforms GAP across each dataset and each privacy level tested. Another notable work is NDP [7]. Recall that NDP fails to provide inference privacy mentioned in Section 4.4, nevertheless, we ignore such weakness of NDP in the experiment. Also recall that in Section 2.3, we mentioned that NDP ensures DP by adding excessive noise, and it is reported in NDP that 1) it needs $\varepsilon \geq 15$ (or $\varepsilon \approx 30$ in some cases) to have acceptable utility; 2) performance in high privacy regime (*e.g.*, $\varepsilon = 2$) is trivial (close to random gussing). We can also observe this fact from Table 2 that, when $\varepsilon = 2$, the test accuracy of NDP is very low, which is consistent with the conclusions given by [7].

Note that Table 2 shows the results for transductive settings, and the results under inductive setting are provided in Appendix C (Table 4). All results show that our method's performance leads by a large margin compared with existing baselines, especially in the high privacy regime. We also report the results of classification precision in Table 3, which is relevant to the discussion in Section 6.

## 5.2. Ablation Studies

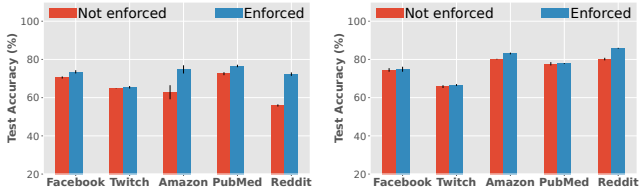
In this part, we present the ablation studies on our design choices and some other aspects that provide some guidance in the parameter selections in *node-level* privacy-preserving GNN applications.

**What if we use Gaussian noise**? To confirm the effectiveness of choosing SML noise instead of Gaussian, we show the performance comparison in this part. In Figure 5, we can see a wide performance gap between SML and Gaussian across all tasks. Notably, when $\varepsilon = 2$, using Gaussian on dataset Reddit leads to a performance close to random guessing. Although the gaps become narrower when we operate at the lowest privacy level ($\varepsilon = 16$), performance on Reddit is still not acceptable. We can also observe that the gap is widest on Reddit; this is because Reddit has the largest number of nodes, a situation where Gaussian noise is predicted to perform even worse. This phenomenon confirms our analysis in Section 4.3.




(a) $\varepsilon = 2$ (b) $\varepsilon = 16$

Figure 5: Performance comparison between different randomness.



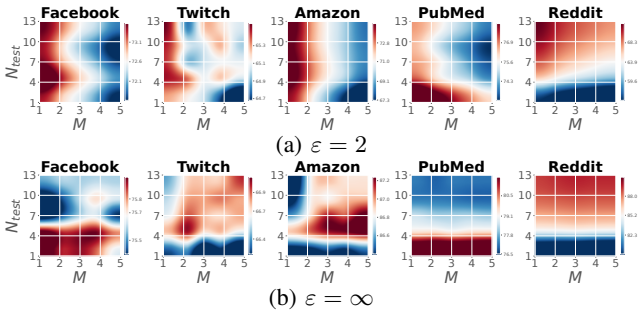

(a) $\varepsilon = 2$ (b) $\varepsilon = 16$

Figure 6: Performance comparison between when line 15 to 22 in Algorithm 2 is "Enforced" and "Not enforced".

**Necessity of enforcing no *peripheral* node overlaps with *central* nodes**. We study the effect of line 15 to 22 in Algorithm 2. In principle, without such operations, our privacy accounting can still apply. This only requires modifying $A_k$ in Equation (3), *i.e.*, the distribution of $\rho$ (also with its support) changes. It is easy to find the new distribution, as elaborated in Appendix B.3. However, in total, this increases the upper bound in Equation (2), resulting in larger noise s.t.d. to ensure the same $(\varepsilon, \delta)$-DP. Hence, it potentially hurts utility.

We present the results for cases when Line 15 to 22 is *enforce* compared to *not enforced* in Figure 6. The results show that the *enforced* version performs better due to smaller noise s.t.d., especially under the most private case. This suggests that although the enforcing line 15 to

22 will alter the sampled sub-graphs, it still benefits utility, suggesting that in real-world applications, line 15 to 22 in Algorithm 2 is necessary.




(a) $\varepsilon = 2$

(b) $\varepsilon = \infty$

Figure 7: Trade-offs in neighborhood sampling in transductive graph setting. We interpolate the accuracy value to make it easier to see the trend. Closer to dark red means better performance.

**Trade-offs in neighborhood sampling**. Increasing $M$ leads to aggregating more information from neighbors, which is good for utility; however, increasing $M$ will also increase the sampling rate, hence less private. In other words, it needs more DP noise to guarantee the same privacy level. Therefore, a trade-off exists in selecting $M$. In addition to $M$, we use $N_{test}$ to denote the number of neighbors to sample during testing, and it is another quantity worthy of investigation.

Results corresponding to GCN model are presented in Figure 7. We give results on privacy levels ($\varepsilon = 2, \infty$). As shown in Figure 7, in the most private case, there is an obvious *noise-averse* phenomenon, *i.e.*, although larger $M$ means *central* nodes aggregating more information from neighborhoods, to guarantee the same privacy level, the additional noise needed outweighs such merit. In contrast, when no noise is added ($\varepsilon = \infty$), we can see that a larger $M$ is favored, which indeed shows that aggregating more neighbors benefits utility. The above phenomenon indicates that the DP noise has a significant impact on the utility. This suggests that, in practice, some small $M$ value is favored during training. From these results, we can also conclude that larger $N_{test}$ tends to benefit utility. However, the experimental results on dataset PubMed show that the best $N_{test}$ lies between 1-4 instead of some larger value. Possibly, this is because the dataset Pubmed itself favors sampling a smaller number of neighbors as its average edges (only 5) are considerably less than that of the rest of the datasets.

### 5.3. Privacy Audit & Resistance to Privacy Attacks

Privacy audit focusing on the validation of theoretical privacy guarantees. In this part, we apply the privacy audit method in [26], [27] to demonstrate that our protocol indeed provides claimed DP guarantees. To test our method in a real-world scenario with privacy threats, we also apply the strong adversary model assumed by privacy auditing to perform a membership inference attack, and this is to test the limit of our method and showcase our protocol's resistance to (even strong) privacy attacks. The algorithm for auditing adapted from [26], is provided in Algorithm 5 of Appendix A.3.

We perform such evaluation on both our method (Figure 8a and 8b) and NDP [7] (Figure 8c and 8d). In Figure 8a, we can see that under the most private scenario ($\varepsilon = 2$), the attack accuracy is close to random guessing. As we have lower privacy requirements, naturally, we observe an increase in attack accuracy. Similar attack performance is observed when evaluated on NDP in Figure 8c, and results suggest that our method's defense against privacy attack is roughly the same as that of NDP.

In Figure 8b, we show the audit results on our protocol. We observe that the audited/empirical $\varepsilon$ is close to the theoretical value when $\varepsilon = 2$; however, as $\varepsilon$ increases, a wider gap is observed. The fact that no audited/empirical $\varepsilon$ is higher than the theoretical value suggests that our protocol is private as claimed. We may also conclude that when we are at a higher privacy regime, our protocol's DP guarantee becomes tighter. We can also see that the privacy accounting of NDP has no bugs, as shown in Figure 8d.

## 6. Study on Private Node Embedding

The purpose of presenting this section is not only to show that [33] fails to ensure claimed privacy protection but also to show that the private node embedding approach fundamentally cannot achieve good utility in high privacy regimes. This implication is important as it is not limited to only graph data. We first present the definition of private node embedding as follows.

**Definition 6** (Private Node Embedding [33])**.** *For a graph $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y})$, the private node embeddings (with dimension $k$) for each node are derived by a sequence of private algorithms $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_{|\mathcal{V}|})$ such that $\mathcal{M}_i(\mathcal{G}) \in \mathbb{R}^k$ outputs the embedding for node $i$ and $\mathcal{M}_i$ is $(\varepsilon_i, \delta_i)$-DP for $i \in \mathcal{V}$, i.e.,*

$$\Pr(\mathcal{M}_i(\mathcal{G}^*) \in S) \leq e^{\varepsilon_i} \Pr(\mathcal{M}_i(\mathcal{G}') \in S) + \delta_i \qquad (6)$$

*holds for any $S \subseteq \mathbb{R}^k$ and any adjacent graph pair $(\mathcal{G}^*, \mathcal{G}')$.*

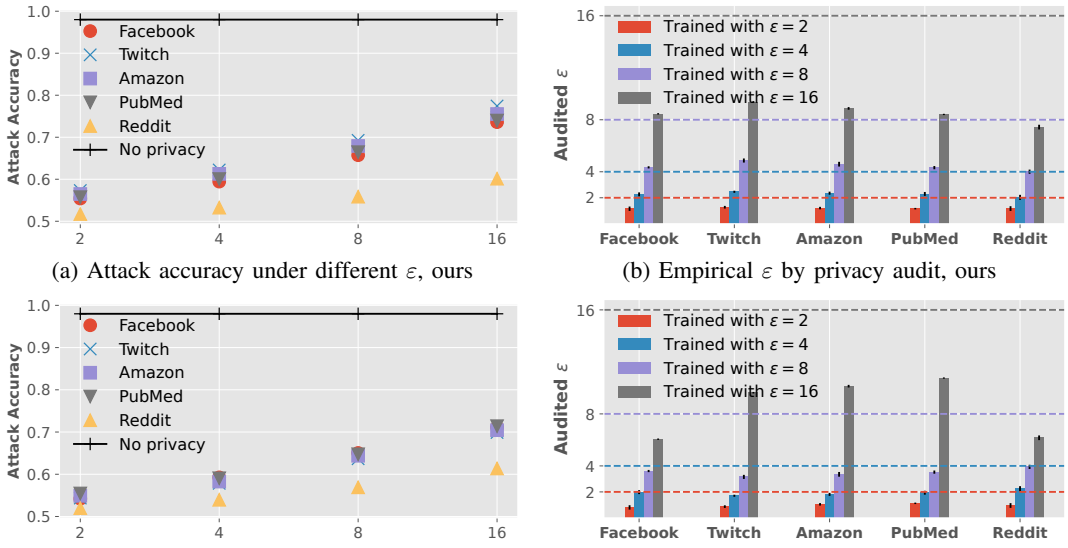
### 6.1. Impossibility Result

**Definition 7** (Class-$\zeta$-aligned Classifier)**.** *Given any graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{Y})$, and private node embeddings algorithm $\mathcal{M}$. Let node $i$ be uniformly randomly chosen from the node set $\mathcal{V}$, and then we derive its private embedding $u_i = \mathcal{M}_i(\mathcal{G})$, i.e., $u_i$ is a random vector, and randomness comes from both sampling $i$ and the private node embedding algorithm $\mathcal{M}_i$. A classifier $h$ is said to be Class-$\zeta$-Aligned if*

$$\Pr(\mathbf{Y}[i] = \zeta) = \Pr(h(u_i) = \zeta). \qquad (7)$$

Intuitively, Equation (7) says that the portion of nodes whose class is $\zeta$ is equal to

$$\frac{\#\text{ nodes predicted to be class } \zeta}{\#\text{ all predictions}}.$$




(a) Attack accuracy under different $\varepsilon$, ours

(b) Empirical $\varepsilon$ by privacy audit, ours

(c) Attack accuracy under different $\varepsilon$, NDP

(d) Empirical $\varepsilon$ by privacy audit, NDP

Figure 8: Results of private attack/audit on our protocol. In (a), we present the membership inference attacks on our protocol. In (b), we present the privacy audit results with 95% confidence. Experiments are performed under a strong adversary model, *i.e.*, the adversary has access to all intermediate privatized gradient vectors and can insert gradient canaries.

It is easy to see that, for a good classifier predicting the class of node $i$, Equation (7) should hold, or at least $h$ should satisfy $\Pr(\mathbf{Y}[i] = \zeta) \approx \Pr(h(u_i) = \zeta)$. We assume such a good classifier to show that *even if we have such a good one, a utility barrier still exists*.

For simplicity and ease of discussion, we take $\Pr(\mathbf{Y}[i] = \zeta) = \Pr(h(u_i) = \zeta)$. Based on Definition 7, the following theorem shows that there exists a utility barrier for private node embedding.

**Theorem 3** (Impossibility Result)**.** *Consider a graph $\mathcal{G}$ whose nodes have $C$ classes. We have a sequence of private node embeddings $\mathcal{M} = (\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_{|\mathcal{V}|})$ where each $\mathcal{M}_k$ is $(\varepsilon, \delta)$-DP for $k \in \{1, 2, \cdots, |\mathcal{V}|\}$. Then, for an arbitrary classifier $h$ which is Class-$\zeta$-Aligned for some $\zeta \in \{1, 2, \cdots, C\}$, we have the following bound on the precision of $h$ if $h$ operates on $u_i$, which is defined in Definition 7*

$$\Pr(\mathbf{Y}[i] = \zeta | h(u_i) = \zeta) \leq \frac{e^{\varepsilon} + \delta(C-1)}{C - 1 + e^{\varepsilon}}. \tag{8}$$

Proof is provided in Appendix B.4. From Equation (8), we can easily see that when in the high privacy regime, *i.e.*, $\varepsilon \to 0$, the upper bound is close to $\frac{1}{C}$. This is close to random guessing as $\mathcal{M}_i$ outputs a purely random vector in the extreme case. Note that there is no assumption on the classifier $h$ other than it is Class-$\zeta$-Aligned, so Equation (8) holds for any classifier, including potentially well-trained ones that perform optimally on non-private test data. The impossibility result shows that it is impossible to have private $\mathcal{M}_i$ and high utility from the output of $\mathcal{M}_i$ simultaneously. For example, when we have $C = 10$ classes in total and each $\mathcal{M}_i$ is $(2, 10^{-5})$-DP, the precision upper bound is almost $\frac{e^2}{9+e^2} \approx 0.45$.

**Implications**. The impossibility result reveals that releasing private node embeddings for each node in a graph has poor trade-offs between privacy and utility. This phenomenon coincides with one of the limitations of DP: *differential privacy is not a meaningful privacy notion if the analysis action is taken on a specific individual* [40]. Our impossibility results are linked to such a claim. Unfortunately, private node embedding falls into this case. It can be more intuitive by considering the following: Naturally, we want the embeddings of two totally different nodes/instances to be *distinguishable* to each other; however, in privacy node/instance embedding, under DP's definition, it enforces the distributions of such two embeddings to be *close/indistinguishable* to each other, conflicting with our expectation, hence not possible to have both strong privacy and good utility.

We provide a case study in Section 6.2 that evaluates a method using private node embedding [33]. The analysis of such a method on privacy has flaws, resulting in a significantly underestimated amount of DP noise, which leads to incorrect privacy guarantees.

## 6.2. Case study

**A fundamental flaw: *Unbounded* or *bounded* DP**? Note that the definition of private node embedding is only reasonable under the *bounded* DP setting. Suppose there is a pair of two adjacent graphs $\mathcal{G}^*$ and $\mathcal{G}' = \mathcal{G}^* \cup \{\text{node } z\}$, if the *unbounded* DP is adopted as its privacy definition, there is a problem when writing down

$$\Pr(\mathcal{M}_z(\mathcal{G}') \in S) \leq e^{\varepsilon} \Pr(\mathcal{M}_z(\mathcal{G}^*) \in S) + \delta,$$

*i.e.*, as there is no node $z$ in $\mathcal{G}^*$. Therefore, this type of information publishing in privacy node/instance embedding is incompatible with *Unbounded* DP formulation.

| $\varepsilon$ | GCN Facebook | GCN Twitch | GCN Amazon | GCN PubMed | GCN Reddit | GIN Facebook | GIN Twitch | GIN Amazon | GIN PubMed | GIN Reddit | SAGE Facebook | SAGE Twitch | SAGE Amazon | SAGE PubMed | SAGE Reddit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | (71.1) | (88.1) | (45.1) | (78.7) | (15.6) | – | – | – | – | – | – | – | – | – | – |
| | $49.2_{\pm 2.0}$ | $42.5_{\pm 8.2}$ | $14.7_{\pm 0.4}$ | $15.4_{\pm 0.4}$ | $32.4_{\pm 1.6}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.3_{\pm 0.9}$ | $55.9_{\pm 0.3}$ | $34.9_{\pm 0.9}$ | $41.0_{\pm 1.2}$ | $15.0_{\pm 0.3}$ | – | – | – | – | – | – | – | – | – | – |
| | $21.8_{\pm 0.1}$ | $17.9_{\pm 0.5}$ | $27.4_{\pm 0.4}$ | $42.6_{\pm 0.3}$ | $43.5_{\pm 0.3}$ | $36.9_{\pm 0.9}$ | $7.3_{\pm 0.9}$ | $9.1_{\pm 0.9}$ | $34.7_{\pm 0.5}$ | $35.8_{\pm 0.3}$ | $22.5_{\pm 0.6}$ | $4.8_{\pm 0.4}$ | $25.2_{\pm 0.9}$ | $44.3_{\pm 0.9}$ | $39.1_{\pm 0.4}$ |
| | **73.4**$_{\pm 0.9}$ | **64.5**$_{\pm 0.6}$ | **71.7**$_{\pm 1.5}$ | **77.2**$_{\pm 0.3}$ | **70.1**$_{\pm 1.2}$ | **73.4**$_{\pm 0.3}$ | **64.6**$_{\pm 0.8}$ | **70.6**$_{\pm 1.3}$ | **81.7**$_{\pm 0.5}$ | **62.4**$_{\pm 0.8}$ | **74.0**$_{\pm 0.8}$ | **64.7**$_{\pm 1.3}$ | **70.6**$_{\pm 2.0}$ | **79.9**$_{\pm 0.2}$ | **66.0**$_{\pm 1.1}$ |
| 4 | $49.4_{\pm 2.2}$ | $40.9_{\pm 8.8}$ | $14.4_{\pm 0.9}$ | $15.4_{\pm 0.4}$ | $33.3_{\pm 1.5}$ | – | – | – | – | – | – | – | – | – | – |
| | $35.2_{\pm 0.4}$ | $59.0_{\pm 0.4}$ | $30.9_{\pm 0.6}$ | $34.8_{\pm 0.4}$ | $16.4_{\pm 0.9}$ | – | – | – | – | – | – | – | – | – | – |
| | $51.4_{\pm 0.2}$ | $35.9_{\pm 0.4}$ | $31.1_{\pm 0.9}$ | $40.7_{\pm 0.5}$ | $54.3_{\pm 0.9}$ | $49.0_{\pm 0.4}$ | $35.1_{\pm 0.9}$ | $16.8_{\pm 0.9}$ | $41.0_{\pm 0.4}$ | $47.7_{\pm 0.9}$ | $39.9_{\pm 0.3}$ | $19.0_{\pm 0.5}$ | $24.5_{\pm 0.2}$ | $41.2_{\pm 0.9}$ | $50.0_{\pm 0.2}$ |
| | **74.3**$_{\pm 1.0}$ | **65.1**$_{\pm 0.7}$ | **77.7**$_{\pm 0.5}$ | **78.4**$_{\pm 0.3}$ | **78.2**$_{\pm 0.8}$ | **74.9**$_{\pm 0.4}$ | **65.1**$_{\pm 0.8}$ | **77.0**$_{\pm 0.3}$ | **83.2**$_{\pm 0.4}$ | **73.1**$_{\pm 0.8}$ | **74.6**$_{\pm 0.6}$ | **65.5**$_{\pm 0.8}$ | **76.7**$_{\pm 1.7}$ | **80.8**$_{\pm 0.3}$ | **74.5**$_{\pm 0.8}$ |
| 8 | $49.5_{\pm 1.8}$ | $40.9_{\pm 8.8}$ | $14.7_{\pm 1.5}$ | $15.4_{\pm 0.4}$ | $35.0_{\pm 1.9}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.1_{\pm 1.2}$ | $56.7_{\pm 0.9}$ | $36.1_{\pm 0.9}$ | $33.3_{\pm 0.3}$ | $19.7_{\pm 0.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $60.8_{\pm 0.4}$ | $65.7_{\pm 0.2}$ | $30.9_{\pm 0.6}$ | $41.7_{\pm 0.3}$ | $67.1_{\pm 0.6}$ | $69.2_{\pm 0.4}$ | $43.5_{\pm 0.9}$ | $29.9_{\pm 0.9}$ | $49.9_{\pm 0.4}$ | $58.4_{\pm 0.4}$ | $52.8_{\pm 0.3}$ | $58.3_{\pm 0.3}$ | $34.9_{\pm 0.9}$ | $41.7_{\pm 0.4}$ | $64.2_{\pm 0.5}$ |
| | **74.4**$_{\pm 1.2}$ | **65.5**$_{\pm 0.5}$ | **81.0**$_{\pm 1.8}$ | **79.2**$_{\pm 0.1}$ | **82.9**$_{\pm 0.3}$ | **75.9**$_{\pm 0.6}$ | **65.2**$_{\pm 0.8}$ | **81.4**$_{\pm 0.9}$ | **84.6**$_{\pm 0.7}$ | **80.0**$_{\pm 0.4}$ | **75.2**$_{\pm 0.6}$ | **66.0**$_{\pm 0.4}$ | **81.0**$_{\pm 1.0}$ | **81.6**$_{\pm 0.5}$ | **80.3**$_{\pm 0.8}$ |
| 16 | $49.8_{\pm 1.3}$ | $40.8_{\pm 8.8}$ | $15.1_{\pm 2.1}$ | $15.4_{\pm 0.4}$ | $35.8_{\pm 2.0}$ | – | – | – | – | – | – | – | – | – | – |
| | $32.6_{\pm 0.1}$ | $56.9_{\pm 0.9}$ | $34.6_{\pm 1.2}$ | $31.4_{\pm 0.4}$ | $24.5_{\pm 1.2}$ | – | – | – | – | – | – | – | – | – | – |
| | $73.8_{\pm 0.1}$ | $68.1_{\pm 0.2}$ | $45.5_{\pm 0.9}$ | $43.2_{\pm 0.9}$ | $71.7_{\pm 0.3}$ | $77.4_{\pm 0.9}$ | $52.4_{\pm 0.9}$ | $40.0_{\pm 0.9}$ | $54.8_{\pm 0.9}$ | $61.8_{\pm 0.9}$ | $69.0_{\pm 0.1}$ | $65.6_{\pm 0.4}$ | $37.5_{\pm 0.3}$ | $43.2_{\pm 0.9}$ | $73.8_{\pm 0.9}$ |
| | **74.7**$_{\pm 1.4}$ | **65.8**$_{\pm 0.5}$ | **81.4**$_{\pm 1.1}$ | **79.3**$_{\pm 0.5}$ | **85.6**$_{\pm 0.3}$ | **76.6**$_{\pm 0.7}$ | **65.6**$_{\pm 0.5}$ | **83.2**$_{\pm 0.6}$ | **85.1**$_{\pm 0.5}$ | **83.8**$_{\pm 0.2}$ | **75.4**$_{\pm 0.5}$ | **66.1**$_{\pm 0.4}$ | **83.6**$_{\pm 2.5}$ | **82.2**$_{\pm 0.6}$ | **83.8**$_{\pm 0.4}$ |

TABLE 3: Classification precision (averaged across classes). The organization of the presentation is identical to Table 2. We add the precision upper bound, *i.e.*, Equation (8), in the first row (in parenthesis) for $\varepsilon = 2$. For larger $\varepsilon$, we omit the upper bound as it becomes trivial.

Besides the above technical explanation, considering the following from an adversary point of view can be much more intuitive. Since the embedding is already claimed private, the privacy adversary can access those private outputs. Suppose we have $|\mathcal{V}|$ nodes in $\mathcal{G}^*$, the adversary can just count the number ($|\mathcal{V}|$ V.S. $|\mathcal{V}|+1$) of embeddings to perfectly distinguish the graph from which those embeddings are derived, thus perfectly infers whether $z$ has participated or not. Note that the adversary will always succeed no matter how much noise is added to the embeddings. Unfortunately, we notice the private node embedding use case with *unbounded* DP formulation [33].

**Incorrect sensitivity analysis**. We know that private node embedding is only compatible with *bounded* DP formulation, and the following discussion is carried out based on this. In [33], a private node embedding for node $i$ needs to be released. Simply speaking, DP noise is added to the result

$$\hat{x}_i = \frac{x_i}{\|x_i\|} + \sum_{j \in \mathcal{NB}(i)} \frac{x_j}{\|x_j\|}$$

for each node $i$ ($x_j$ is the feature vector, and the normalization operation is to enforce bounded sensitivity).

[33] analyzes that the $\ell_2$ sensitivity of $x_i'$ is 1 by Lemma 2 of [33], and it corresponds to the case when one of $i$'s neighbors is the differing node. However, this analysis is problematic because the differing node can be $i$ itself, and the differing node can differ in all of its neighbors, hence the sensitivity is substantially underestimated, specifically when the nodes have D degrees in maximum, the DP noise is underestimated by a $2(\mathrm{D}+1)$ factor (by definition, the $\ell_2$ sensitivity is the $\ell_2$ norm of the difference between the sum of arbitrary $(\mathrm{D}+1)$ normalized vectors and the sum of another $(\mathrm{D}+1)$ normalized vectors, such norm is $2(\mathrm{D}+1)$ in maximum by triangle inequality). Therefore, to correct the amount of DP noise, the noise s.t.d. in [33] should have been multiplied by $2(\mathrm{D}+1)$, the flaw in Lemma 2 in [33] propagates to Lemma 3 and the main theorem in [33]. By this, one can see that there is little utility remaining as the noise intensity always scales with the size of neighbors of one node. Our impossibility results shown previously capture this phenomenon.

### 6.3. Experiments on Precision

With the same experiment setup as in Section 5, we add the precision results to confirm our precision upper bound in Equation (8). Results under the transductive setting are presented in Table 3. We can see that the performance of GAP [33] is below the upper bound after the random noise is corrected, confirming the utility barrier. In contrast, for other methods, it can be seen that they perform better than the upper bound naturally because other methods do not use private instance embedding, hence not limited to the utility barrier. Note that as we have less privacy ($\varepsilon$ becomes greater), the precision upper bound becomes trivial (approaching 1), which is the reason why we do not list the bound for settings with lower privacy in Table 3.

## 7. Related Work

**Differential privacy in machine learning**. Depending on where the randomness is injected, there are roughly three categories: 1) *output pertubation*, this method perturb the trained model [5], [15] ; 2) *object modification*, this method adds randomness to the loss function [17], [49] ; 3) *gradient perturbation*, this method adds noise to the gradient during training [1], [3] . Notably, the third one does not assume any convexity of the loss function, which makes it more suitable for general machine learning/deep learning applications. Among successfully applied protocols, DP-SGD [1], [37] is a prime example. Our work falls under the third category.

**Differential privacy in graph statistic analysis**. This type of work, differing from learning tasks, focuses on privacy issues in analyzing graph data statistics. For instance, a line of work tackling the problems in releasing sub-graph counting [14], [38] , some other works focus on degree distributions [8], [12]. Interestingly, *node-level* privacy is also considered significantly harder to obtain than *edge-level* without incurring notable utility loss even in the central DP model [31]. We also focus on *node-level* privacy, although the tasks are different.

**Differentially privacy in GNNs**. Preserving edges/nodes' privacy is also a natural request in GNNs. *Node-level* privacy is much stronger than *edge-level* privacy, as the former

implies the latter. There is notable work tackling *edge-level* privacy protection. Wu et al. propose an edge-level privacy protection method using Laplace Mechanism [42]. Kolluri et al. provide another *edge-level* privacy solution [20], improving above [42]. In contrast, there is no paralleled previous work tackling *node-level* privacy protection with strong results. Like graph statistics analysis tasks, compared with *edge-level*, ensuring *node-level* privacy is also more challenging.

**Privacy attacks to learning GNNs on graphs**. This type of work is on the *adversary* side while our work is on the *defender* side. Depending on the applications, multiple privacy attacks exist under various threat models. Zhang et al. [51] provide an empirical study on the privacy issues when releasing graph embeddings, demonstrating risks at releasing. Wu et al. [42] provide evidence that the existence of an edge can be inferred by querying on GNN models, showing the privacy risks at test/inference. Recent work by He et al. also studies the membership inference attack on nodes [13], highlighting equal urgency of preserving *node-level* privacy in learning GNN on graphs.

## 8. Conclusion

In this study, we present a novel and non-trivial method to protect *node-level* privacy in learning GNNs on graphs. To provide a better understanding of our design, we first introduce a motivating experiment that leads to our designed sampling routine, namely *HeterPoisson*. Additionally, we utilize SML noise instead of the commonly employed Gaussian noise to obfuscate the gradient. Our privacy accounting demonstrates that this choice, rather than Gaussian noise, guarantees non-trivial privacy. Our experimental results also illustrate the non-trivial utility afforded by this choice. Furthermore, we theoretically establish that the family of spherical noise can be adopted to obfuscate the output of a function with bounded $\ell_2$-sensitivity, which can be of independent interest. To further evaluate the effectiveness of our privacy solution, we subject it to membership inference attacks and privacy audit techniques. Through experimental analysis, we demonstrate that our protocol is resistant to privacy attacks and it has no bugs.

In addition to our privacy solution, we investigate a method that adopts private node/instance embedding to address the issue of *node-level* privacy. Specifically, we conduct a case study to reveal the fundamental privacy flaws in a recent study that adopts this approach. More importantly, we highlight the inherent utility barriers associated with the private instance embedding approach, providing insights into pitfalls that should be avoided in privacy applications.

## Acknowledgement

We thank all anonymous reviewer construction feedback. Di Wang and Zihang Xiang were supported by BAS/1/1689-01-01, URF/1/4663-01-01, FCC/1/1976-49-01, RGC/3/4816-01-01, and REI/1/4811-10-01 of King Abdullah University of Science and Technology (KAUST) and KAUST-SDAIA Center of Excellence in Data Science and Artificial Intelligence. Tianhao Wang is supported by CNS-2220433.

## References

[1] 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.

[2] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 2496–2506. PMLR, 2020.

[3] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 464–473. IEEE, 2014.

[4] Béla Bollobás. *Random Graphs, Second Edition*, volume 73 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 2011.

[5] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. *Advances in neural information processing systems*, 21, 2008.

[6] Rui Chen, Benjamin CM Fung, Philip S Yu, and Bipin C Desai. Correlated network data publication via differential privacy. *The VLDB Journal*, 23:653–676, 2014.

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

[8] Wei-Yen Day, Ninghui Li, and Min Lyu. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*, pages 123–138, 2016.

[9] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[12] 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.

[13] 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.

[14] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. Locally differentially private analysis of graph statistics. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 983–1000. USENIX Association, 2021.

[15] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 1895–1912. USENIX Association, 2019.

[16] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[17] Daniel Kifer, Adam D. Smith, and Abhradeep Thakurta. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pages 25.1–25.40. JMLR.org, 2012.

[18] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[20] Aashish Kolluri, Teodora Baluta, Bryan Hooi, and Prateek Saxena. Lpgnet: Link private graph networks for node classification. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1813–1827, 2022.

[21] Samuel Kotz, Tomasz Kozubowski, and Krzysztof Podgórski. *The Laplace distribution and generalizations: a revisit with applications to communications, economics, engineering, and finance*. Number 183. Springer Science & Business Media, 2001.

[22] Ninghui Li, Wahbeh Qardaji, Dong Su, Yi Wu, and Weining Yang. Membership privacy: A unifying framework for privacy definitions. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 889–900, 2013.

[23] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[24] Ilya Mironov. Rényi differential privacy. In *30th IEEE Computer Security Foundations Symposium, CSF 2017, Santa Barbara, CA, USA, August 21-25, 2017*, pages 263–275. IEEE Computer Society, 2017.

[25] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *CoRR*, abs/1908.10530, 2019.

[26] Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. USENIX Association, 2023.

[27] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 866–882. IEEE, 2021.

[28] Joseph P. Near and Xi He. Differential privacy for databases. *Found. Trends Databases*, 11(2):109–225, 2021.

[29] Iyiola E. Olatunji, Thorben Funke, and Megha Khosla. Releasing graph neural networks with differential privacy guarantees. *CoRR*, abs/2109.08907, 2021.

[30] 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.

[31] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. Generating synthetic decentralized social graphs with local differential privacy. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 425–438, 2017.

[32] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

[33] Sina Sajadmanesh, Ali Shahin Shamsabadi, Aurélien Bellet, and Daniel Gatica-Perez. Gap: Differentially private graph neural networks with aggregation perturbation. In *32nd USENIX Security Symposium*, 2023.

[34] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[35] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[36] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 3–18. IEEE Computer Society, 2017.

[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] Shuang Song, Susan Little, Sanjay Mehta, Staal A. Vinterbo, and Kamalika Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, abs/1809.02575, 2018.

[39] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: Differential privacy as a causal property. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 354–371. IEEE, 2020.

[40] Salil P. Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer International Publishing, 2017.

[41] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. Adapting membership inference attacks to gnn for graph classification: Approaches and implications. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 1421–1426. IEEE, 2021.

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

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[44] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[45] 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.

[46] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[47] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. Differentially private fine-tuning of language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[48] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[49] Jun Zhang, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, and Marianne Winslett. Functional mechanism: Regression analysis under differential privacy. *Proc. VLDB Endow.*, 5(11):1364–1375, 2012.

[50] Qiuchen Zhang, Jing Ma, Jian Lou, Carl Yang, and Li Xiong. Towards training graph neural networks with node-level differential privacy. *arXiv preprint arXiv:2210.04442*, 2022.

[51] 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)*, pages 4543–4560, 2022.

[52] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

# Appendix A. Content for reference

## A.1. Used GNN model

The following three GNN models are used in our evaluations.

- GCN: the aggregation is a weighted summation over representations of the last layer,

$$\mathbf{h}_u^{(k+1)} = \phi^{(k)}\Big(\sum_{j\in\mathcal{NB}(u)\cup\{u\}} \frac{1}{\sqrt{d_j d_u}}\mathbf{h}_j^{(k)}\Big)$$

  where $d_u$ is the degree of node $u$.
- GIN: the update can be described as,

$$\mathbf{h}_u^{(k+1)} = \phi^{(k)}\Big(\sum_{j\in\mathcal{NB}(u)} \mathbf{h}_j^{(k)} + (1+\lambda)\mathbf{h}_u^{(k)}\Big)$$

  where $\lambda$ is a learnable parameter.
- SAGE: take the mean aggregator as an example, the update can be described as,

$$\mathbf{h}_u^{(k+1)} = \phi^{(k)}(\text{mean}\{\mathbf{h}_j^{(k)} | j \in \mathcal{NB}(u)\cup\{u\}\})$$

**GNN training routine**. For example, for the node-classification tasks, the training of a GNN model $w$ can be described as the following.

1) *Forward propagation*: *i.e.*, update $\mathbf{h}_i^k, i \in \mathcal{V}$ until we get the results $\mathbf{h}_i^K, i \in \mathcal{V}$ after $K$-th iteration; feed $\mathbf{h}_i^K, i \in \mathcal{V}$ to a classification module (often a fully connected layer followed by a *softmax* layer) to get the predictions $\hat{y}_i, i \in \mathcal{V}$; compute the loss by some loss metric on $\hat{y}_i$ and the ground-truth label $\mathbf{Y}[\mathbf{i}]$ for $i \in \mathcal{V}$. The total loss for a graph $\mathcal{G}$ is often the averaged loss of all nodes, and we denote it as $f(\mathcal{G}; w)$ and $f$ is the loss function.
2) *Backward propagation*: *i.e.*, compute the gradient with respect to the learnable parameter $g = \nabla f(\mathcal{G}; w)$.
3) *Model update*: update $w$ using $g$ by gradient decent.

## A.2. Experiment Setups

**Parameters**. To set up the graph for experiments, the training and testing nodes are randomly split by $(80\%, 20\%)$ from the original graph. We set the neighborhood sampling multiplier $M$ to be the one lead to best performance among $\{1, 2, 3, 4\}$. For privacy parameters, we set $\delta = \frac{1}{|\mathcal{V}|^{1.1}}$ and vary $\varepsilon$ to be $\{2, 4, 8, 16\}$ for all datasets. We set $q_b = \frac{4096}{|\mathcal{V}|}$ for all datasets and $T = \lceil \frac{9}{q_b} \rceil$ for Facebook, Twitch, Amazon and PubMed, $T = \lceil \frac{4}{q_b} \rceil$ for Reddit. We use $N_{test}$ to denote the number of neighbors to samples during testing, and $N_{test} = 13$ unless otherwise re-clarified.
**Network setup**. For the neural network setup, we instantiate the update function $\phi$ to be a multilayer perceptron (MLP) with $elu$ non-linear activation; the input dimension is determined by the dimension of the node's feature vectors, and we use the same hidden dimension setup $hid\_dim = 128$ for all GNN models.
**Software implementation:** Our implementation builds on top of PyTorch Geometric 2.2.0 and Pytorch 1.13. To efficiently parallelize per-sub-graph gradient computation, we leverage *functorch*'s *vmap* primitive.

## A.3. Privacy Audit

The algorithm for privacy audit is presented in Algorithm 5, which is adapted from [26]. Auditing is done on observations $\mathbf{O}^*, \mathbf{O}'$ following section 5.2 of [26]; simply speaking, the goal is trying to distinguish whether the canary gradient is included or not. For the threshold setup, we use various threshold values and choose the one with the strongest audit/attack result. The membership inference attack is also carried out based on $\mathbf{O}^*, \mathbf{O}'$.

**Algorithm 5** White-box Privacy Audit with Gradient Canaries

**Input:** Identical to input of Algorithm 1
Initialize Observations $\mathbf{O}^* \leftarrow \emptyset$, $\mathbf{O}' \leftarrow \emptyset$
**for** $t = 1, 2, \cdots, T$ **do**
  Run line 2 to 6 in Algorithm 1
  ▷ Insert the **Dirac canary gradient** as defined in [26]
  ▷ Set canary gradient $g^c = [k, 0, 0, \cdots, 0], k \sim \rho$ where distribution $\rho$ is defined in Theorem 2
  W.p. $q_b$, choose some gradient $\hat{g}_i$ and let $\hat{g}_i \leftarrow \hat{g}_i + g^c$
  $\bar{g}^t \leftarrow \sum \hat{g}_i$
  $g^t \leftarrow \bar{g}^t + \mathcal{LAP}(0, \sigma\mathbb{I}^d)$
  $\mathbf{O}'[t] \leftarrow \langle g^t, g^c \rangle$, $\mathbf{O}^*[t] \leftarrow \langle g^t - g^c, g^c \rangle$
  $w^t \leftarrow w^{t-1} - \eta g^t$
**end for**
**Output:** Observations $\mathbf{O}^*, \mathbf{O}'$

# Appendix B. Proofs

## B.1. Proof of Lemma 1

*Proof.* $\mathcal{M}(X) = \mathcal{B}(f(X), \sigma^2\mathbb{I}^d)$, and $\mathcal{M}(X') = \mathcal{B}(f(X'), \sigma^2\mathbb{I}^d)$. Since Rényi divergence is translation invariant, it is equivalent to compute $\mathcal{D}_\alpha(\mathcal{B}(f(X) -$

$f(X'), \sigma^2\mathbb{I}^d)||\mathcal{B}(0, \sigma^2\mathbb{I}^d))$. Note that the noise is spherical, and for each value that $f(X) - f(X')$ may take, we can always apply a rotation by left multiplying a unitary matrix $U$ such that $U(f(X) - f(X')) = [c, 0, \cdots, 0]$ for some $c \leq k$, and then equivalently consider the one-dimension case:

$$\mathcal{D}_\alpha(\mathcal{B}(c, \sigma^2)||\mathcal{B}(0, \sigma^2)) = \mathcal{D}_\alpha(\mathcal{B}(1, \frac{\sigma^2}{c^2})||\mathcal{B}(0, \frac{\sigma^2}{c^2})).$$

This is because left multiplying a unitary matrix $U$ is $\ell_2$-length-preserving, *i.e.*, it does not change the density value, hence does not change the divergence. Then, we can obtain $\mathcal{B}(1, \frac{\sigma^2}{c^2})$ from $\mathcal{B}(1, \frac{\sigma^2}{k^2})$ by adding noise $\mathcal{B}(0, \frac{\sigma^2}{c^2} - \frac{\sigma^2}{k^2})$, similarly, we can obtain $\mathcal{B}(0, \frac{\sigma^2}{c^2})$ from $\mathcal{B}(0, \frac{\sigma^2}{k^2})$ by adding noise $\mathcal{B}(0, \frac{\sigma^2}{c^2} - \frac{\sigma^2}{k^2})$. Then, we have

$$\begin{aligned}\mathcal{D}_\alpha(\mathcal{M}(X)||\mathcal{M}(X')) &= \mathcal{D}_\alpha(\mathcal{B}(1, \frac{\sigma^2}{c^2})||\mathcal{B}(0, \frac{\sigma^2}{c^2})) \\ &\leq \mathcal{D}_\alpha(\mathcal{B}(1, \frac{\sigma^2}{k^2})||\mathcal{B}(0, \frac{\sigma^2}{k^2})) \\ &= \mathcal{D}_\alpha(\mathcal{B}(k, \sigma^2)||\mathcal{B}(0, \sigma^2))\end{aligned}$$

where the inequality is due to data processing inequality of Rényi divergence.

□

## B.2. Proof of Lemma 2

*Proof.* First, for the function $r : (0, \infty)^2 \to (0, \infty)$ give by $r(u, v) = u^\alpha v^{1-\alpha}$, we can see that $r$ is convex for $\alpha > 1$. Hence, by Jensen's inequality, we have

$$\mathbb{E}[U]^\alpha \mathbb{E}[V]^{1-\alpha} = r(\mathbb{E}[(U, V)]) \leq \mathbb{E}[r(U, V)] = \mathbb{E}[U^\alpha V^{1-\alpha}]$$

holds for a pair of random variables $(U, V)$. Then we have

$$\begin{aligned}&\int \mathbb{E}_{k\sim\rho}[\mu_k(x)]^\alpha \mathbb{E}_{k\sim\rho}[\xi_k(x)]^{1-\alpha} \mathrm{d}x \\ \leq &\int \mathbb{E}_{k\sim\rho}[\mu_k(x)^\alpha \xi_k(x)^{1-\alpha}] \mathrm{d}x = \mathbb{E}_{k\sim\rho}\left[\int \mu_k(x)^\alpha \xi_k(x)^{1-\alpha} \mathrm{d}x\right]\end{aligned} \tag{9}$$

the inequality holds because

$$\mathbb{E}_{k\sim\rho}[\mu_k(x)]^\alpha \mathbb{E}_{k\sim\rho}[\xi_k(x)]^{1-\alpha} \leq \mathbb{E}_{k\sim\rho}[\mu_k(x)^\alpha \xi_k(x)^{1-\alpha}]$$

holds point-wise for $x$. Then, based on Equation (9) and by the definition of Rényi divergence, we have

$$\begin{aligned}e^{(\alpha-1)\mathcal{D}_\alpha(\mu_\rho||\xi_\rho)} &= \int \mathbb{E}_{k\sim\rho}[\mu_k(x)]^\alpha \mathbb{E}_{k\sim\rho}[\xi_k(x)]^{1-\alpha} \mathrm{d}x \\ \leq &\mathbb{E}_{k\sim\rho}\left[\int \mu_k(x)^\alpha \xi_k(x)^{1-\alpha} \mathrm{d}x\right] = \mathbb{E}_{k\sim\rho} e^{(\alpha-1)\mathcal{D}_\alpha(\mu_k||\xi_k)}\end{aligned}$$

□

## B.3. Proof of Theorem 2

*Proof.* The $T$ multiplication exists because $T$-fold adaptive composition results in linear add-up in Rényi divergence. It is sufficient to only upper-bound the divergence for one iteration. We denote $\nu_k^\sigma(x)$ as the density function of univariate Laplace distribution with mean $k$ and s.t.d. $\sigma$, *i.e.*, $\nu_k^\sigma(x) = \frac{1}{\sqrt{2}\sigma} \exp\left(\frac{-\sqrt{2}|x-k|}{\sigma}\right)$.

1) Suppose we are in this case: differing node $z$ is sampled, such as the case in Figure 3a, the final sub-graph container $G'$ **has only 1 more sub-graph than** $G^*$, which centers around $z$. Denote $\bar{g}^t|_{G^*}$ as the output shown at line 7 of Algorithm 1 if the graph container is $G^*$ due to input graph $\mathcal{G}^*$, and similar, $\bar{g}^t|_{G'}$ due to input graph $\mathcal{G}'$, we have

$$\max \|\bar{g}^t|_{G^*} - \bar{g}^t|_{G'}\|_2 = \max \|\hat{g}_z\|_2 = 0.5$$

where $\hat{g}_z$ is the clipped gradient computed on sub-graph centering around $z$. Conditioned on this case, denote the distribution for $\mathcal{G}^*$ and $\mathcal{G}'$ at line 8 as $g^t|_{G^*,0.5}$ and $g^t|_{G',0.5}$, respectively. We have $\mathcal{D}_\alpha(g^t|_{G^*,0.5}||g^t|_{G',0.5}) \leq \mathcal{D}_\alpha(\nu_{0.5}^\sigma(x)||\nu_0^\sigma(x))$ by Lemma 1. This case happens with probability $q_b$ as $z$ is sampled with probability $q_b$.

2) Suppose we are in another case: differing node $z$ is **not** sampled and $k$ out-pointing nodes of $z$ are sampled, and all of those $k$ nodes also sample $z$ as neighbors, as depicted in Figure 3b where we draw the scenario when $k = 1$. Note that container $G^*, G'$ **differ in** $k$ **sub-graphs**, then, we have

$$\max \|\bar{g}^t|_{G^*} - \bar{g}^t|_{G'}\|_2 = 2 \times k \times 0.5 = k.$$

Conditioned on this case and in a similar way, denote the distribution for $\mathcal{G}^*$ and $\mathcal{G}'$ at line 8 as $g^t|_{G^*,k}$ and $g^t|_{G',k}$, respectively. We have $\mathcal{D}_\alpha(g^t|_{G^*,k}||g^t|_{G',k}) \leq \mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x))$ by Lemma 1. This case happen with probability $(1 - q_b)\mathbf{Bi}(k; D_{ot}, \frac{q_b M}{D_{ot}})$, where $D_{ot}$ is out-degree of $z$.

Now express the distribution due to $\mathcal{G}^*$ and $\mathcal{G}'$ at line 8 as (mixture distribution) $g^t|_{G^*,\rho} = \sum_{k\sim\rho} \Pr_\rho(k) g^t|_{G^*,k}$ and $g^t|_{G',\rho} = \sum_{k\sim\rho} \Pr_\rho(k) g^t|_{G',k}$, respectively. We have

$$\begin{aligned}&e^{(\alpha-1)\mathcal{D}_\alpha(g^t|_{G^*,\rho}||g^t|_{G',\rho})} \\ \leq &\mathbb{E}_{k\sim\rho} e^{(\alpha-1)\mathcal{D}_\alpha(g^t|_{G^*,k}||g^t|_{G',k})} \\ \leq &\mathbb{E}_{k\sim\rho} e^{(\alpha-1)\mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x))}\end{aligned}$$

The first inequality is due to Lemma 2; the second is due to our analysis for each case. When $D_{ot} = 0$, the above reasoning also applies. $\mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x))$ has a closed-form solution, with a simple calculation, we have

$$\begin{aligned}&\mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x)) \\ = &\frac{1}{\alpha-1} \ln\left(\frac{\alpha}{2\alpha-1} e^{\sqrt{2}(\alpha-1)k/\sigma} + \frac{\alpha-1}{2\alpha-1} e^{-\sqrt{2}\alpha k/\sigma}\right)\end{aligned}$$

With a little simplification, we have

$$e^{(\alpha-1)\mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x))} \leq \frac{\alpha}{2\alpha-1} e^{\sqrt{2}(\alpha-1)k/\sigma} + \frac{1}{2}$$

which explain Equation (2).

Note that we ensure a guarantee for any node with any $D_{ot}$, hence taking maximum over all possible $D_{ot}$. A final remark is: to have valid privacy bound for a randomized algorithm $\mathcal{M}$ (denote $\mathcal{M}^* = \mathcal{M}(\mathcal{G}^*)$ and $\mathcal{M}' = \mathcal{M}(\mathcal{G}')$ in our case), due to symmetric of adjacency, we must ensure

$$\gamma \leq \max(\mathcal{D}_\alpha(\mathcal{M}^*||\mathcal{M}'), \mathcal{D}_\alpha(\mathcal{M}'||\mathcal{M}^*)).$$

It already satisfies such because $\mathcal{D}_\alpha(\nu_k^\sigma(x)||\nu_0^\sigma(x)) = \mathcal{D}_\alpha(\nu_0^\sigma(x)||\nu_k^\sigma(x))$. □

| $\varepsilon$ | GCN Facebook | GCN Twitch | GCN Amazon | GCN PubMed | GCN Reddit | GIN Facebook | GIN Twitch | GIN Amazon | GIN PubMed | GIN Reddit | SAGE Facebook | SAGE Twitch | SAGE Amazon | SAGE PubMed | SAGE Reddit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | $48.6_{\pm 3.7}$ | $59.0_{\pm 2.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $41.9_{\pm 1.6}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.3_{\pm 0.9}$ | $55.9_{\pm 0.3}$ | $34.9_{\pm 0.9}$ | $41.0_{\pm 1.2}$ | $25.4_{\pm 0.3}$ | – | – | – | – | – | – | – | – | – | – |
| | $19.1_{\pm 0.1}$ | $13.7_{\pm 0.5}$ | $24.7_{\pm 0.4}$ | $38.8_{\pm 0.3}$ | $39.2_{\pm 0.3}$ | $32.7_{\pm 0.9}$ | $4.9_{\pm 0.9}$ | $4.6_{\pm 0.9}$ | $32.0_{\pm 0.5}$ | $35.3_{\pm 0.3}$ | $19.1_{\pm 0.6}$ | $2.3_{\pm 0.4}$ | $22.1_{\pm 0.9}$ | $39.9_{\pm 0.9}$ | $35.4_{\pm 0.4}$ |
| | **73.5**$_{\pm 0.9}$ | **65.4**$_{\pm 0.7}$ | **74.8**$_{\pm 2.1}$ | **77.1**$_{\pm 0.4}$ | **72.3**$_{\pm 0.9}$ | **73.5**$_{\pm 0.3}$ | **65.3**$_{\pm 0.8}$ | **73.8**$_{\pm 1.3}$ | **81.4**$_{\pm 0.4}$ | **65.4**$_{\pm 0.9}$ | **74.2**$_{\pm 0.8}$ | **65.5**$_{\pm 1.1}$ | **74.3**$_{\pm 1.0}$ | **79.8**$_{\pm 0.2}$ | **69.2**$_{\pm 1.1}$ |
| 4 | $51.0_{\pm 3.6}$ | $60.2_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $42.7_{\pm 1.5}$ | – | – | – | – | – | – | – | – | – | – |
| | $35.2_{\pm 0.4}$ | $59.0_{\pm 0.4}$ | $30.9_{\pm 0.6}$ | $34.8_{\pm 0.4}$ | $25.5_{\pm 0.9}$ | – | – | – | – | – | – | – | – | – | – |
| | $48.5_{\pm 0.2}$ | $32.7_{\pm 0.4}$ | $30.5_{\pm 0.9}$ | $37.4_{\pm 0.5}$ | $50.9_{\pm 0.9}$ | $47.2_{\pm 0.4}$ | $32.2_{\pm 0.9}$ | $13.4_{\pm 0.9}$ | $39.2_{\pm 0.4}$ | $45.1_{\pm 0.9}$ | $38.4_{\pm 0.3}$ | $15.1_{\pm 0.5}$ | $22.9_{\pm 0.2}$ | $40.5_{\pm 0.9}$ | $49.2_{\pm 0.2}$ |
| | **74.5**$_{\pm 1.0}$ | **66.0**$_{\pm 0.6}$ | **79.6**$_{\pm 0.2}$ | **78.4**$_{\pm 0.3}$ | **79.1**$_{\pm 0.7}$ | **74.8**$_{\pm 0.3}$ | **65.9**$_{\pm 0.8}$ | **79.1**$_{\pm 0.7}$ | **82.9**$_{\pm 0.3}$ | **74.6**$_{\pm 0.8}$ | **74.7**$_{\pm 0.6}$ | **66.4**$_{\pm 0.6}$ | **79.2**$_{\pm 0.5}$ | **80.8**$_{\pm 0.3}$ | **76.0**$_{\pm 0.9}$ |
| 8 | $52.1_{\pm 3.0}$ | $60.4_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $43.4_{\pm 1.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $34.1_{\pm 1.2}$ | $56.7_{\pm 0.9}$ | $36.1_{\pm 0.9}$ | $33.3_{\pm 0.3}$ | $29.1_{\pm 0.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $56.9_{\pm 0.4}$ | $61.6_{\pm 0.2}$ | $27.9_{\pm 0.6}$ | $38.3_{\pm 0.3}$ | $62.2_{\pm 0.6}$ | $64.9_{\pm 0.4}$ | $41.8_{\pm 0.9}$ | $26.1_{\pm 0.9}$ | $47.7_{\pm 0.4}$ | $54.9_{\pm 0.4}$ | $52.7_{\pm 0.3}$ | $56.8_{\pm 0.3}$ | $32.0_{\pm 0.9}$ | $39.3_{\pm 0.4}$ | $62.4_{\pm 0.5}$ |
| | **74.6**$_{\pm 1.1}$ | **66.4**$_{\pm 0.4}$ | **82.1**$_{\pm 0.6}$ | **79.1**$_{\pm 0.2}$ | **83.2**$_{\pm 0.3}$ | **75.8**$_{\pm 0.7}$ | **66.1**$_{\pm 0.8}$ | **82.8**$_{\pm 0.3}$ | **84.4**$_{\pm 0.5}$ | **80.6**$_{\pm 0.3}$ | **75.2**$_{\pm 0.6}$ | **66.9**$_{\pm 0.3}$ | **82.0**$_{\pm 0.8}$ | **81.6**$_{\pm 0.5}$ | **81.1**$_{\pm 0.8}$ |
| 16 | $52.9_{\pm 2.7}$ | $60.4_{\pm 0.8}$ | $37.4_{\pm 0.9}$ | $39.3_{\pm 0.5}$ | $44.0_{\pm 1.4}$ | – | – | – | – | – | – | – | – | – | – |
| | $32.6_{\pm 0.1}$ | $56.9_{\pm 0.9}$ | $34.6_{\pm 1.2}$ | $31.4_{\pm 0.4}$ | $34.8_{\pm 1.2}$ | – | – | – | – | – | – | – | – | – | – |
| | $70.3_{\pm 0.1}$ | $63.7_{\pm 0.2}$ | $41.2_{\pm 0.9}$ | $40.3_{\pm 0.9}$ | $67.9_{\pm 0.3}$ | $75.2_{\pm 0.9}$ | $50.5_{\pm 0.9}$ | $38.9_{\pm 0.9}$ | $52.0_{\pm 0.9}$ | $59.9_{\pm 0.9}$ | $67.3_{\pm 0.1}$ | $64.4_{\pm 0.4}$ | $32.8_{\pm 0.3}$ | $39.6_{\pm 0.9}$ | $69.9_{\pm 0.9}$ |
| | **74.9**$_{\pm 1.3}$ | **66.6**$_{\pm 0.5}$ | **83.1**$_{\pm 0.5}$ | **79.2**$_{\pm 0.5}$ | **85.8**$_{\pm 0.2}$ | **76.5**$_{\pm 0.7}$ | **66.4**$_{\pm 0.5}$ | **84.2**$_{\pm 0.6}$ | **84.9**$_{\pm 0.5}$ | **84.1**$_{\pm 0.3}$ | **75.5**$_{\pm 0.4}$ | **66.9**$_{\pm 0.3}$ | **83.8**$_{\pm 1.1}$ | **82.1**$_{\pm 0.6}$ | **84.1**$_{\pm 0.4}$ |

TABLE 4: Classification accuracy. The setup is identical to Table 2, and the graph setting is inductive.

**What if not enforcing no overlapping between *peripheral* nodes and *central* nodes**? The distribution of $\rho$ changes under this case. It is easy to find out the expression for $\Pr_\rho(k)$. We consider two separate cases: 1) the differing node is sampled as a *central* node, then $\Pr_\rho(k+0.5) = q_b\mathbf{Bi}(k; D_{ot}, \frac{q_b M}{D_{ot}})$ for $k \in [D_{ot}]$; 2) otherwise $\Pr_\rho(k) = (1-q_b)\mathbf{Bi}(k; D_{ot}, \frac{q_b M}{D_{ot}})$ for $k \in [D_{ot}]$.

## B.4. Proof of Theorem 3

*Proof.* we uniformly randomly pick node $i$ and w.l.o.g., suppose the class of interest is 1. The goal is to upper-bound the precision w.r.t. label 1, *i.e.*, $\Pr[\mathbf{Y}[i]=1|h(u_i)=1]$. By Bayes' theorem, we have:

$$\Pr[\mathbf{Y}[i]=1|h(u_i)=1] = \frac{\Pr[h(u_i)=1|\mathbf{Y}[i]=1]\Pr[\mathbf{Y}[i]=1]}{\Pr[h(u_i)=1]}$$

Note that the classifier is Class-1-Aligned, *i.e.*, $\Pr[\mathbf{Y}[i]=1] = \Pr[h(u_i)=1]$. Hence, we have:

$$\Pr[\mathbf{Y}[i]=1|h(u_i)=1] = \Pr[h(u_i)=1|\mathbf{Y}[i]=1] \tag{10}$$

Based on Definition 6, we know Equation (6) holds for any adjacent graph pair $(\mathcal{G}^*, \mathcal{G}')$, and this also includes the case that the current node $i$ of interest is the differing node. This means that the class of node with ID $i$ in $\mathcal{G}^*$ and $\mathcal{G}'$ can differ, *i.e.*, in $\mathcal{G}^*$ we have $\mathbf{Y}[i]=1$ and in $\mathcal{G}'$ we can have $\mathbf{Y}[i]=2$. Define $S=\{u_i|h(u_i)=1\}$, we then have:

$$\Pr[h(u_i)=1|\mathbf{Y}[i]=1] \le e^\varepsilon \Pr[h(u_i)=1|\mathbf{Y}[i]=2] + \delta,$$

similarly, we also have:

$$\Pr[h(u_i)=1|\mathbf{Y}[i]=1] \le e^\varepsilon \Pr[h(u_i)=1|\mathbf{Y}[i]=3] + \delta,$$

$$\cdots$$

$$\Pr[h(u_i)=1|\mathbf{Y}[i]=1] \le e^\varepsilon \Pr[h(u_i)=1|\mathbf{Y}[i]=C] + \delta,$$

note that the events $\mathbf{Y}[i]=l, \forall l \in 1,2,\cdots,C$ are mutually exclusive, then we have:

$$\Pr[h(u_i)=1|\mathbf{Y}[i]=1] = 1 - \sum_{l=2}^{C} \Pr[h(u_i)=1|\mathbf{Y}[i]=l],$$

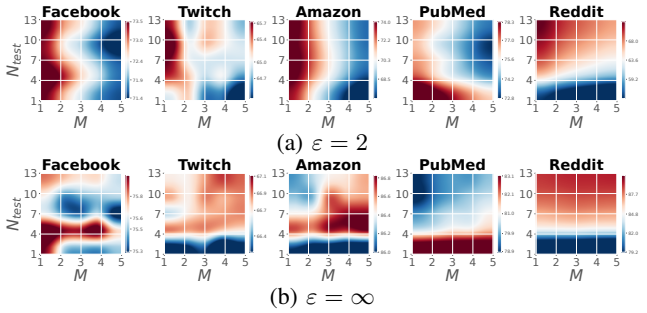
adding both the right-hand side and left-hand side of the above $C-1$ inequalities, rearrange, we get:

$$\Pr[h(u_i)=1|\mathbf{Y}[i]=1] \le \frac{e^\varepsilon + \delta(C-1)}{C-1+e^\varepsilon}$$

based on Equation (10), we have the final result:

$$\Pr[\mathbf{Y}[i]=1|h(u_i)=1] \le \frac{e^\varepsilon + \delta(C-1)}{C-1+e^\varepsilon}$$

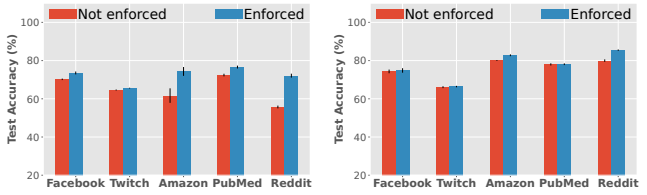
□




(a) $\varepsilon = 2$

(b) $\varepsilon = \infty$

Figure 9: Trade-offs in neighborhood sampling in inductive graph setting. The setup is identical to Figure 7




(a) $\varepsilon = 2$ (b) $\varepsilon = 16$

Figure 10: Performance comparison. The setup is identical to Figure 6, and the graph setting is inductive.

# Appendix C. More experimental results

Additional counterpart results on the inductive graph setting are in this section. The accuracy is shown in Table 4, and the investigation for the inductive graph is presented in Figure 10. Ablation study on neighborhood sampling under is provided in Figure 9.

# Appendix D.
# Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

## D.1. Summary

This research tackles privacy concerns in Graph Neural Networks (GNNs). They propose a new algorithm, Heter-Poisson, to safeguard node privacy during training. Heter-Poisson utilizes neighborhood sampling and injects noise for privacy protection, achieving a balance between privacy and model performance.

## D.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

## D.3. Reasons for Acceptance

1) The paper tackles a critical issue (privacy risks in GNNs) and introduces a new approach (Heter-Poisson) to mitigate these risks. This demonstrates the potential for the paper to make a significant contribution to the field.
2) The research employs various evaluation methods, including ablation studies, privacy audits, and comparisons with baselines. Using symmetric multivariate Laplace noise ensures a measurable level of privacy protection.
3) The authors demonstrate the effectiveness of HeterPoisson through experiments on real-world datasets. The results show that the approach achieves privacy goals while maintaining acceptable performance, making it a practical solution.