AdvSGM: Differentially Private Graph Learning via Adversarial Skip-gram Model

Sen Zhang[1], Qingqing Ye[1], Haibo Hu[1], Jianliang Xu[2]
[1] The Hong Kong Polytechnic University, Hong Kong
[2] Hong Kong Baptist University, Hong Kong
{senzhang, qqing.ye, haibo.hu}@polyu.edu.hk, xujl@comp.hkbu.edu.hk

***Abstract*—The skip-gram model (SGM), which employs a neural network to generate node vectors, serves as the basis for numerous popular graph embedding techniques. However, since the training datasets contain sensitive linkage information, the parameters of a released SGM may encode private information and pose significant privacy risks. Differential privacy (DP) is a rigorous standard for protecting individual privacy in data analysis. Nevertheless, when applying differential privacy to skip-gram in graphs, it becomes highly challenging due to the complex link relationships, which potentially result in high sensitivity and necessitate substantial noise injection. To tackle this challenge, we present AdvSGM, a differentially private skip-gram for graphs via adversarial training. *Our core idea is to leverage adversarial training to privatize skip-gram while improving its utility*. Towards this end, we develop a novel adversarial training module by devising two optimizable noise terms that correspond to the parameters of a skip-gram. By fine-tuning the weights between modules within AdvSGM, we can achieve differentially private gradient updates without additional noise injection. Extensive experimental results on six real-world graph datasets show that AdvSGM preserves high data utility across different downstream tasks. Our code and datasets have been made available at https://github.com/NetDPer/AdvSGM.**

***Index Terms*—Differential privacy, Skip-gram model, Adversarial training, Graph embedding**

I. Introduction

Graph embedding, which has attracted increasing research attention, represents nodes by low-dimensional vectors while preserving the inherent properties and structures of the graph. In this way, well-studied machine learning algorithms can be easily applied for further mining tasks like clustering, classification, and prediction. Skip-gram models (SGMs) are a popular class of graph embedding models thanks to their simplicity and effectiveness, including DeepWalk [1], LINE [2], and node2vec [3]. However, SGMs, which capture not only general data characteristics but also specific details about individual data, are vulnerable to adversarial attacks, particularly user-linkage attacks [4] that exploit the linkage information between nodes to infer whether an individual is present in the training dataset. Therefore, node embeddings need to be sanitized with privacy guarantees before they can be released to the public.

Differential privacy (DP) [5] is a well studied statistical privacy model recognized for its rigorous mathematical underpinnings. In this paper, we study the problem of achieving privacy-preserving skip-gram for graphs under differential privacy. The work most related to ours is by Ahuja *et al.* [6], where private learning from sparse location data is achieved by leveraging SGM in conjunction with differentially private stochastic gradient descent (DPSGD) [7], a prevalent approach for differentially private training in deep neural networks. Despite its success, this method cannot be extended to skip-gram for graphs, as DPSGD is inherently more suitable for structured data with well-defined individual gradients and does not adapt effectively to graph data. The primary reason for this limitation is that individual examples in a graph are no longer independently computed for their gradients. As a result, achieving differentially private skip-gram for graphs typically encounters significant noise due to high sensitivity, making it challenging to strike a balance between privacy and utility.

Adversarial training, originally developed to protect against adversarial attacks in computer vision [8], has proven effective in enhancing the robustness of deep learning models. Recently, adversarial training is used to enhance the robustness and generalizability of skip-gram [9]. Inspired by this, we propose AdvSGM, a differentially private skip-gram that injects differential noise into the activation function of adversarial training. This approach leverages adversarial training to achieve privacy protection while enhancing the utility of skip-gram. Nevertheless, achieving this is challenging as the injected noise may be eliminated during gradient calculations. To overcome this issue, we design a novel adversarial training module by creating two optimizable noise terms associated with the parameters of the skip-gram that require optimization. Through fine-tuning the weights of the skip-gram and adversarial training modules, we demonstrate that the combined noise effectively contributes to privacy protection throughout the optimization process. Additionally, we provide empirical evidence to support the rationale behind this tuning.

Our main contributions are listed as follows.

- We present AdvSGM, a novel differentially private skip-gram for graphs that incorporates adversarial training. To our best knowledge, it is the ***first*** approach which utilizes adversarial training to privatize skip-gram while improving its utility.
- We design a novel adversarial training module by introducing two carefully crafted noise terms in activation functions. By fine-tuning the weights between modules, we can achieve differential privacy without requiring additional noise injection during optimization.
- Through formal privacy analysis, we prove that our Ad-

vSGM guarantees node-level differential privacy. Extensive experiments on six real graph datasets demonstrate that our solution significantly surpasses existing state-of-the-art private graph embedding methods in both link prediction and node clustering tasks.

The remainder of this paper is organized as follows. Section II covers the preliminaries. In Section III, we formulate the problem and introduce a naive solution. Our proposed AdvSGM is detailed in Section IV, and its privacy and time complexity are discussed in Section V. Comprehensive experimental results are presented in Section VI. Section VII reviews related works. Finally, a conclusion is drawn in Section VIII.

II. Preliminaries

In this section, we will provide a brief overview of adversarial skip-gram model, differential privacy and DPSGD, highlighting their key concepts and important properties. The notations frequently used in this paper are summarized in Table I.

TABLE I
Frequently Used Symbols

Symbol	Description
ϵ, δ	Privacy parameters
$\mathcal{G}, \overline{\mathcal{G}}$	Original graph and neighboring graph
$\mathcal{A}$	A randomized algorithm
α	Order of Rényi divergence
D_α	Rényi divergence of order α
$\|V\|, \|E\|$	Number of nodes and edges in $\mathcal{G}$
$\mathbf{x}, \mathbf{y}$	Lowercase letters denoting vectors
$\mathbf{x} \cdot \mathbf{y}$	Inner product between two vectors
$\mathbf{X}, \mathbf{Y}$	Bold capital letters denoting matrices
$\mathbf{I}$	Identity matrix
$\mathbf{W}_{in}, \mathbf{W}_{out}$	Embedding matrices of skip-gram
k	Negative sampling number
r	Dimension of low-dimensional vectors
B	Number of samples
γ	Sampling probability
G, D	Generator and discriminator
L^D_{sgm}	Loss function for structure preservation in D
L^D_{adv}	Loss function for adversarial training in D
L^G	Loss function of generator G
λ	Weight between modules
$\mathcal{N}_{D,1}, \mathcal{N}_{D,2}$	Gaussian noise for D
$\mathcal{N}_{G,1}, \mathcal{N}_{G,2}$	Gaussian noise for G
Θ^D, Θ^G	Parameter sets of D and G
n^D, n^G	Number of epochs for D and G
$f \circ g$	Composition of two functions

A. Notations

We define an undirected graph as $\mathcal{G} = (V, E)$, where V is the set of nodes and E is the set of edges. For nodes $v_i, v_j \in V$, the pair $(i, j) \in E$ represents an edge in $\mathcal{G}$. The goal of graph embedding is to learn a function $f : V \to \mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{|V| \times r}$ is a matrix with embedding dimension $r \ll |V|$, while preserving the intrinsic properties and structures of the original graph $\mathcal{G}$. We denote $\mathbf{v}_i = f(v_i)$ as the vector embedding of node v_i.

B. Adversarial Skip-gram Model

The skip-gram model is a neural network architecture that learns word embeddings by predicting the surrounding words given a target word. In the context of graph embedding, each node in the network can be thought of as a "word", and the surrounding words are defined as the nodes that co-occur with the target node in the network. Inspired by this setting, DeepWalk [1] achieves graph embedding generation by treating the paths traversed by random walks over networks as sentences and using skip-gram to learn latent representations of nodes. With the advent of DeepWalk, several skip-gram based graph embedding generation models have emerged, including LINE [2], PTE [10], and node2vec [3]. Recently, adversarial training is used to improve the utility of skip-gram [9]. The enhanced model mainly consists of two components: generator and discriminator.

1) Generator: The generator G incorporates two generators $G_{v'_i}$ and $G_{v'_j}$. The neighbor generator $G_{v'_j} = \phi(\mathcal{N}_{G,1}(\sigma^2\mathbf{I}) \cdot \theta^G_{v'_j})$ for the real node v_i and the neighbor generator $G_{v'_i} = \phi(\mathcal{N}_{G,2}(\sigma^2\mathbf{I}) \cdot \theta^G_{v'_i})$ for the real node v_j, where ϕ denotes an activation function that enables non-linear mappings, such as the *Sigmoid* function, $\theta^G_{v'_j}$ and $\theta^G_{v'_i}$ denote the parameters to be optimized, and Gaussian distributions $\mathcal{N}_{G,1}$ and $\mathcal{N}_{G,2}$ generate noise vectors. Both generators produce embeddings $\mathbf{v}'_i$ and $\mathbf{v}'_j$, representing the generated ***fake neighbors*** v'_i and v'_j for v_j and v_i respectively. By doing so, we obtain two node pairs: (v_i, v'_j) and (v'_i, v_j), where v_i and v_j are real nodes obtained from the discriminator. The primary purpose of generator G is to deceive the discriminator, and thus its loss function L^G is determined as follows:

$$L^G = \min \mathbb{E}_{(i,j)\in E}(\log(1 - F(\mathbf{v}_i \cdot \mathbf{v}'_j)) + \log(1 - F(\mathbf{v}'_i \cdot \mathbf{v}_j))). \tag{1}$$

The function $F(\cdot)$ acts as the *discriminant function* within the discriminator. Its output ranges between 0 and 1, representing the probability that the input node pairs (v_i, v'_j) and (v'_i, v_j) are considered positive.

2) Discriminator: The discriminator is divided into two modules: one module is a graph structure reservation module (i.e., skip-gram) for learning graph structure and the other is an adversarial training module to improve the performance of the model.

Skip-gram Module. The purpose of the graph structure preservation module is to preserve the original graph structure in the low-dimensional embedding space. Many skip-gram based graph embedding methods can be used directly as the graph structure preservation module, such as DeepWalk [1], LINE [2], etc. Taking LINE as an example, for each node pair (v_i, v_j), the loss function L^D_{sgm} is

$$L^D_{sgm} = \log \sigma(\mathbf{v}_i \cdot \mathbf{v}_j) + \sum_{n=1}^{k} \mathbb{E}_{v^n_j \sim \mathbb{P}_n(v)} \left[\log \sigma(-\mathbf{v}^n_j \cdot \mathbf{v}_i)\right], \tag{2}$$

where k is the number of negative samples, and $\mathbb{P}_n(v)$ is the sampling distribution of ***negative samples***.

Remark 1. *In this paper, positive samples are node pairs drawn from the edge set E. For negative sampling, we use the starting node of a positive sample and pair it with a randomly selected node from the node set V. As a result, negative samples can include pairs where the edge (i, j) is either in E or not in E (see Algorithm 2 for details).*

Adversarial Training Module. The objective of the adversarial training module is to determine the authenticity of the input node pair. Given an input node pair (v_i, v_j), we utilize the *Sigmoid* function as the discriminant function, which provides the probability that the node pair is true. In order to achieve this, the generator G generates synthetic neighbor nodes v_j' and v_i' for nodes v_i and v_j respectively. Subsequently, two node pairs (v_i, v_j') and (v_i', v_j) are formed. By incorporating the adversarial training module using the discriminant function $F(\cdot)$, the loss function can be derived as

$$\begin{aligned} L_{adv}^D =& \mathbb{E}_{(i,j)\in E}(-\log(1 - F(\mathbf{v}_i \cdot \mathbf{v}_j')) \\ & - \log(1 - F(\mathbf{v}_i' \cdot \mathbf{v}_j))). \end{aligned} \tag{3}$$

By combining the graph structure preservation module and the adversarial training module, the loss function of the discriminator D can be expressed as

$$L^D = L_{sgm}^D + \lambda L_{adv}^D, \tag{4}$$

where $\lambda > 0$ denotes a weight that controls the relative importance of L_{sgm}^D and L_{adv}^D.

Remark 2. *In Eq. (2), $\sigma(\cdot)$ denotes the Sigmoid function, which is widely used in skip-gram models. In this paper, both the discriminant function $F(\cdot)$ in Eqs. (1) and (3) and the activation function $\phi(\cdot)$ also use the Sigmoid function. These settings are beneficial for achieving our goal of utilizing adversarial training to privatize the skip-gram model while improving its utility.*

C. Differential Privacy

(ϵ, δ)-DP. Differential privacy (DP) [11] is the prevailing concept of privacy for algorithms operating on statistical databases. In simple terms, DP restricts the extent to which the output distribution of a mechanism can change when there is a small change in its input. When dealing with graph data, the notion of neighboring databases is established by considering two graph datasets: $\mathcal{G}$ and $\overline{\mathcal{G}}$. These datasets are considered neighbors if their difference is limited to at most one edge or node.

Definition 1 (Edge (Node)-Level DP [12])**.** *A graph analysis mechanism $\mathcal{A}$ is said to satisfy edge- or node-level (ϵ, δ)-DP, if, for any two neighboring graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ (which differ by at most one edge or node), and for all possible sets $O \subseteq Range(\mathcal{A})$, we have $\mathbb{P}[\mathcal{A}(\mathcal{G}) \in O] \leq \exp(\epsilon) \cdot \mathbb{P}[\mathcal{A}(\overline{\mathcal{G}}) \in O] + \delta$.*

The concept of neighboring datasets $\mathcal{G}$ and $\overline{\mathcal{G}}$ can be categorized into two types. Specifically, if $\overline{\mathcal{G}}$ is derived by replacing a single data point in $\mathcal{G}$, it is referred to as **bounded DP** [11]. On the other hand, if $\overline{\mathcal{G}}$ is obtained by adding or removing a data point from $\mathcal{G}$, it is known as **unbounded DP** [13]. The privacy parameter ϵ represents the privacy budget, which quantifies the trade-off between privacy and utility in the algorithm. A smaller value of ϵ indicates a stronger privacy guarantee. The parameter δ is typically chosen to be very small and is informally referred to as the failure probability, representing the likelihood of a privacy violation.

Important properties. DP has the following important properties that help us design complex algorithms from simpler ones:

Theorem 1 (Sequential Composition [5])**.** *Let $f(\mathcal{G})$ be (ϵ_1, δ_1)-DP and $g(\mathcal{G})$ be (ϵ_2, δ_2)-DP, then the mechanism $F(\mathcal{G}) = (f(\mathcal{G}), g(\mathcal{G}))$ which releases both results satisfies $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

A nice property of DP is that the privacy guarantee is not affected by postprocessing steps, as stated in the following theorem.

Theorem 2 (Post-Processing [5])**.** *If $f(\mathcal{G})$ satisfies (ϵ, δ)-DP, then for any function g that does not have direct or indirect access to the private database $\mathcal{G}$, $g(f(\mathcal{G}))$ satisfies (ϵ, δ)-DP.*

(α, ϵ)-RDP. In this paper, we adopt an alternative definition of DP known as Rényi Differential Privacy (RDP) [14], which enables stronger results for sequential composition.

Definition 2 (RDP [14])**.** *Given $\alpha > 1$ and $\epsilon > 0$, a randomized algorithm $\mathcal{A}$ satisfies (α, ϵ)-RDP if for every adjacent datasets $\mathcal{G}$ and $\overline{\mathcal{G}}$, we have $D_\alpha\left(\mathcal{A}(\mathcal{G}) \| \mathcal{A}\left(\overline{\mathcal{G}}\right)\right) \leq \epsilon$, where $D_\alpha(P\|Q)$ is the Rényi divergence of order α between probability distributions P and Q defined as $D_\alpha(P\|Q) = \frac{1}{\alpha-1} \log \mathbb{E}_{x\sim Q}\left[\frac{P(x)}{Q(x)}\right]^\alpha$.*

A key property of RDP is that it can be converted to standard (ϵ, δ)-DP using Proposition 3 from [14], as outlined below.

Theorem 3 (From RDP to (ϵ, δ)-DP [14])**.** *If $\mathcal{A}$ is an (α, ϵ)-RDP algorithm, then it also satisfies $(\epsilon + \log(1/\delta)/\alpha - 1, \delta)$-DP for any $\delta \in (0, 1)$.*

Gaussian mechanism. Consider a function f that maps a graph $\mathcal{G}$ to an r-dimensional output in $\mathbb{R}^r$. To ensure differential privacy for the function f, it is common to add random noise to its output. The amount of noise added is determined by the sensitivity of f, which is defined as $\Delta_f = \max_{\mathcal{G},\overline{\mathcal{G}}} \|f(\mathcal{G}) - f(\overline{\mathcal{G}})\|_2$, where $\mathcal{G}$ and $\overline{\mathcal{G}}$ are neighboring graphs. A widely used method to achieve RDP is the Gaussian mechanism, which adds Gaussian noise to the output of an algorithm to protect privacy. By adding Gaussian noise with variance σ^2 to the function f, the mechanism can be defined as $\mathcal{A}(\mathcal{G}) = f(\mathcal{G}) + \mathcal{N}(\sigma^2\mathbf{I})$, where $\mathcal{N}(\sigma^2\mathbf{I})$ represents noise drawn from a Gaussian distribution with variance σ^2. This results in an (α, ϵ)-RDP algorithm for all $\alpha > 1$, where the privacy parameter ϵ is given by $\epsilon = \frac{\alpha\Delta_f^2}{2\sigma^2}$.

It is important to note that the concept of sensitivity makes satisfying node-level differential privacy challenging, as changing a single node could potentially remove $|V| - 1$

edges in the worst case, where $|V|$ represents the number of nodes. Consequently, a large amount of noise must be added to ensure privacy protection.

Amplification by Subsampling. Subsampling introduces a non-zero probability of an added or modified sample not to be processed by the randomized algorithm. Random sampling will enhance privacy protection and reduce privacy loss [15]–[17]. In this paper, we focus on the "subsampling without replacement" setup, which adheres to the following privacy amplification theorem for (ϵ, δ)-DP.

Theorem 4 (RDP for Subsampled Mechanisms [15])**.** *Given a dataset of n points drawn from a domain $\mathcal{X}$ and a mechanism $\mathcal{A}$ that accepts inputs from $\mathcal{X}^m$ for $m \leq n$, we consider the randomized algorithm $\mathcal{A}$ for subsampling, which is defined as follows: 1) sample m data points without replacement from the dataset, where the sampling parameter is $\gamma = m/n$, and 2) apply $\mathcal{A}$ to the subsampled dataset. For all integers $\alpha \geq 2$, if $\mathcal{A}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP, then the subsampled mechanism $\mathcal{A} \circ \text{subsample}$ satisfies $(\alpha, \epsilon'(\alpha))$ RDP in which*

$$\epsilon'(\alpha) \leq \frac{1}{\alpha - 1} \log \Big(1 + \gamma^2 \binom{\alpha}{2} \Big) \min \Big\{ 4\big(e^{\epsilon(2)} - 1\big), e^{\epsilon(2)} \min \Big\{ 2, \big(e^{\epsilon(\infty)} - 1\big)^2 \Big\} \Big\} + \sum_{j=3}^{\alpha} \gamma^j \binom{\alpha}{j} e^{(j-1)\epsilon(j)} \min \Big\{ 2, \big(e^{\epsilon(\infty)} - 1\big)^j \Big\}.$$

D. DPSGD

One common technique for achieving differentially private training is the combination of noisy Stochastic Gradient Descent (SGD) and advanced composition theorems such as Moments Accountant (MA) [7]. This combination, known as DPSGD, has been widely studied in recent years for publishing low-dimensional node vectors, as *the advanced composition theorems can effectively manage the problem of excessive splitting of the privacy budget during optimization*. In DPSGD, the gradient $\mathbf{g}(x_i)$ is computed for each example x_i in a batch with size B of random examples. The ℓ_2 norm of each gradient is then clipped using a threshold C to control the sensitivity of $\mathbf{g}(x_i)$. The clipped gradients $\mathbf{g}(x_i)$ are summed and combined with Gaussian noise $\mathcal{N}\big(C^2\sigma^2\mathbf{I}\big)$ to ensure privacy. The average of the resulting noisy accumulated gradient $\tilde{\mathbf{g}}$ is then used to update the model parameters. The expression for $\tilde{\mathbf{g}}$ is given by:

$$\tilde{\mathbf{g}} = \frac{1}{B}\Big(\sum_{i=1}^{B} \mathbf{clip}\big(\mathbf{g}(x_i), C\big) + \mathcal{N}\big(C^2\sigma^2\mathbf{I}\big)\Big), \tag{5}$$

where $\mathbf{clip}(\cdot)$ is a clipping function, and specifically, $\mathbf{clip}\big(\mathbf{g}(x_i), C\big) = \mathbf{g}(x_i) / \max\big(1, \frac{\|\mathbf{g}(x_i)\|_2}{C}\big)$. *For the convenience of presentation, we will replace $\mathbf{clip}\big(\mathbf{g}(x_i), C\big)$ with $\mathbf{clip}\big(\mathbf{g}(x_i)\big)$ in the following sections.*

III. Problem Definition and A First-Cut Solution

A. Problem Definition

The advantages of graph embeddings, including low dimensionality, information richness, and task independence, have led to a growing willingness among data owners to publish them for data exploration and analysis, rather than disclosing the original graph data. This work specifically concentrates on a differentially private graph embedding generation. Instead of releasing a sanitized version of the original embeddings, we release a privacy-preserving SGM that is trained on the original data while maintaining privacy. Once equipped with this privacy-preserving model, the analyst can generate synthetic embeddings for the intended analysis tasks.

Threat Model. In our scenario, we consider the white-box attack [18], where the adversary has the full knowledge of our proposed model, including their architectures and parameters. In other words, attackers can access the published model instead of the training process. The goal of the proposed scheme is that even though the attackers have the ability to obtain other data samples in the training dataset, they cannot infer the target training data sample.

Privacy Model. As stated in Sections II-C and II-D, DP ensures that although attackers can have all information from the training dataset except one data sample, they still cannot get this data sample after launching attack. The post-processing property (Theorem 2) allows us to move the burden of differential privacy to the discriminator, with the generator's differential privacy being guaranteed by the theorem. Formally, we define differentially private adversarial skip-gram as follows.

Definition 3 (Adversarial Skip-gram under Bounded DP[1])**.** *Let $\Theta = [\mathbf{W}_{in}, \mathbf{W}_{out}]$ be the set of parameters to be optimized in the skip-gram module (as shown in Eq. (2)), where $\mathbf{v}_i \in \mathbf{W}_{in}$ and $\mathbf{v}_j \in \mathbf{W}_{out}$. The adversarial skip-gram model L^D satisfies node-level (ϵ, δ)-DP if two neighboring graphs $\mathcal{G}$ and $\overline{\mathcal{G}}$ differ in only one node and its corresponding edges, and for all possible $\Theta_S \subseteq Range(L^D)$, we have*

$$\mathbb{P}(L^D(\mathcal{G}) \in \Theta_S) \leq \exp(\epsilon) \times \mathbb{P}\left(L^D\left(\overline{\mathcal{G}}\right) \in \Theta_S\right) + \delta,$$

where Θ_S denotes the set comprising all possible values of Θ.

Leveraging the robustness to post-processing, we can immediately conclude that the (ϵ, δ)-private graph embedding generation is robust to graph downstream tasks, as formalized in the following theorem:

Theorem 5. *Let L^D be a private graph embedding generation model that satisfies node-level (ϵ, δ)-DP, and let f be any graph downstream task that takes the privacy-preserving graph embedding matrix (i.e., $\mathbf{W}_{in}$ or $\mathbf{W}_{out}$) as input. Then, the composition $f \circ L^D$ also satisfies node-level (ϵ, δ)-DP.*

B. DP-ASGM: A First-Cut Solution

As stated in Section II-D, DPSGD with the advanced composition mechanism can manage the issue of excessive splitting of the privacy budget during optimization. One

[1]Since our goal is to generate privacy-preserving node embeddings where the number of embeddings matches the number of nodes in the original graph, we define the node-level privacy-preserving graph embedding under bounded DP.

straightforward approach, called as DP-ASGM, to achieve differentially private skip-gram with adversarial training is to perturb the sum of clipped gradients for discriminator. Using $\mathbf{v}$ as a general notation representing either $\mathbf{v}_i$ or $\mathbf{v}_j$ in Eq. (4), the noisy gradient $\widetilde{\nabla}_{\mathbf{v}} L^D$ is expressed as follows:

$$\widetilde{\nabla}_{\mathbf{v}} L^D = \frac{1}{B}\Big(\sum_{(i,j)\in E_B} \mathbf{clip}\big(\frac{\partial L^D}{\partial \mathbf{v}}\big) + \mathcal{N}\left(B^2 C^2 \sigma^2 \mathbf{I}\right)\Big), \quad (6)$$

in which E_B denotes a batch sample set, and the sensitivity of $\sum_{(i,j)\in E_B} \mathbf{clip}\big(\frac{\partial L^D}{\partial \mathbf{v}}\big)$ may be up to BC under Definition 3. The main reason is that clipping in DPSGD is designed to minimize the amount of noise added to gradients during the backward pass of each data point. While this method naturally suits structured data with well-defined individual gradients, it cannot be seamlessly extended to deep graph learning. In graph learning, individual examples no longer compute their gradients independently because changing a single node in the graph may affect the gradients of all nodes in a batch. Therefore, the sensitivity of the gradient sum in Eq. (6) is proportional to the batch size.

Limitation. However, the approach described above leads to poor utility. This is primarily due to the large sensitivity, which introduces significant noise and hampers the effectiveness of the optimization process.

IV. Our Proposal: AdvSGM

To resist the high sensitivity of DP-ASGM, we present AdvSGM, a differentially private skip-gram model that leverages adversarial training to privatize skip-gram while improving its utility. First, we provide an overview of the approach. Next, we describe how we introduce optimizable noise terms and achieve gradient perturbation by adjusting the weights between modules. Finally, we present the complete training algorithm.

A. Overview

AdvSGM aims to privately learn the embedding vectors for each node in an undirected graph. Fig. 1 illustrates the proposed framework, which consists of two main components: the generator and the discriminator. Given a node pair (v_i, v_j), two generators are employed to generate fake neighbor nodes v_j' and v_i' from two Gaussian distributions. And one discriminator, including skip-gram and adversarial training module, is assigned to distinguish whether the neighborhoods of the generated nodes (v_j' and v_i') are real or fake. Specifically, the adversarial training module is to ensure that discriminator's update satisfies DP, without introducing additional noise. Following this, the generator will naturally obey DP due to the post-processing property of DP (Theorem 2). Through the minimax game between the generators and the discriminator, AdvSGM can privately learn more accurate and robust node embeddings.

Challenge. Based on Eq. (4), we take $\frac{\partial L^D}{\partial \mathbf{v}_i} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \lambda \frac{\partial L^D_{adv}}{\partial \mathbf{v}_i}$ as an example to illustrate the challenge in achieving a differentially private discriminator through adversarial training. Our *target gradient* is to decompose $\frac{\partial L^D}{\partial \mathbf{v}_i}$ into the form described in Eq. (7),

$$\frac{\partial L^D}{\partial \mathbf{v}_i} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \ell + \mathcal{N}(\sigma^2 \mathbf{I}), \quad (7)$$

where we suppose $\lambda \frac{\partial L^D_{adv}}{\partial \mathbf{v}_i} = \ell + \mathcal{N}(\sigma^2 \mathbf{I})$.

To achieve *private gradient* update without additional noise injection, we can clip the first two terms in Eq. (7) as follows:

$$\frac{\partial L^D}{\partial \mathbf{v}_i} = \mathbf{clip}\big(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \ell\big) + \mathcal{N}(C^2 \sigma^2 \mathbf{I}). \quad (8)$$

However, the *real gradient* of $\frac{\partial L^D}{\partial \mathbf{v}_i}$ is as follows:

$$\begin{aligned}
\frac{\partial L^D}{\partial \mathbf{v}_i} &= \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \lambda \frac{\partial L^D_{adv}}{\partial \mathbf{v}_i} && (9)\\
&= \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \lambda \frac{\partial\big(-\log(1 - F(\mathbf{v}_i \cdot \mathbf{v}_j'))\big)}{\partial \mathbf{v}_i} && (10)\\
&\overset{(1)}{=} \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \lambda F(\mathbf{v}_i \cdot \mathbf{v}_j') \cdot \mathbf{v}_j', && (11)
\end{aligned}$$

where (1) holds because $F(\cdot)$ is a *Sigmoid* function, and its derivative is $\frac{\partial F(x)}{\partial x} = F(x)(1 - F(x))$.

Clearly, this real gradient cannot be expressed in the form given in Eq. (7). Therefore, the main challenge is to develop an adversarial loss function that can produce the desired gradient while maintaining high data utility.

Our Solution. We address this challenge through two steps:

- We design a novel adversarial training module by introducing optimizable noise terms, $\mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i$ and $\mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j$, to ensure that

$$\frac{\partial L^D}{\partial \mathbf{v}_i} = \big(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \ell_1\big) + \lambda \ell_2 \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}). \quad (12)$$

- We fine-tune the weights that control relative importance between modules to achieve the desired target gradient form, and more importantly, we show that this configuration is reasonable.

B. Introducing Optimizable Noise Terms

To address the second objective while ensuring the first objective is not compromised, we incorporate two optimizable noise terms that correspond to the parameters of a skip-gram, namely $\mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i$ and $\mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j$, into the input of the *Sigmoid* functions. The loss function can be derived as follows:

$$\begin{aligned}
\widetilde{L}^D_{adv} = &\underbrace{\mathbb{E}_{(i,j)\in E}(-\log(1 - F(\mathbf{v}_i \cdot \mathbf{v}_j' + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i)))}_{\widetilde{L}^D_{adv1}} \\
&+ \underbrace{\mathbb{E}_{(i,j)\in E}(-\log(1 - F(\mathbf{v}_i' \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j)))}_{\widetilde{L}^D_{adv2}}.
\end{aligned} \quad (13)$$

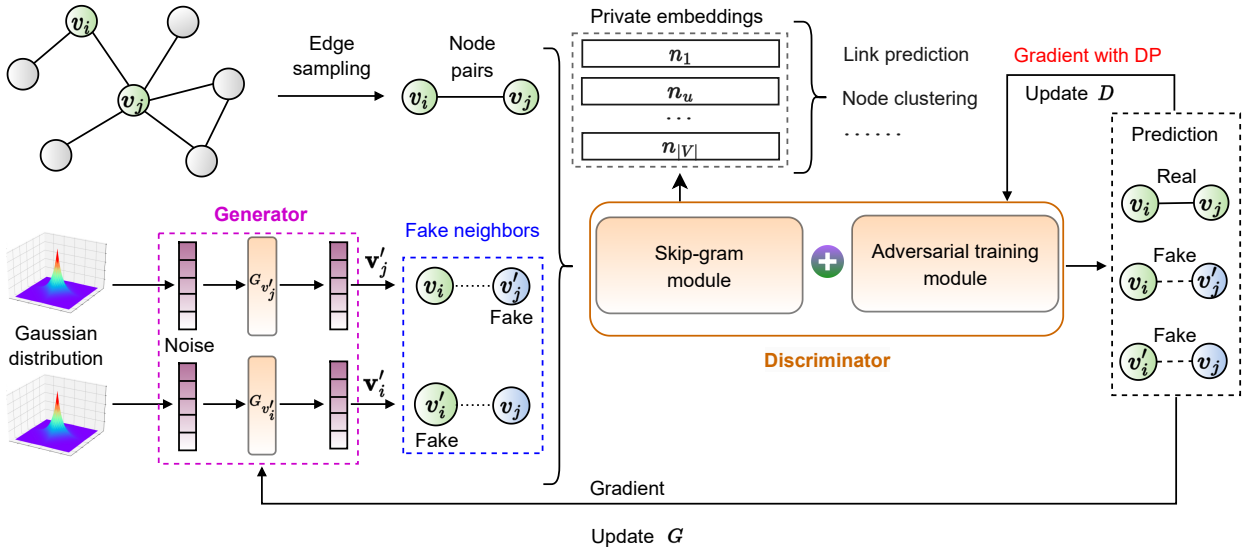

Fig. 1. Architecture of AdvSGM. The discriminator can be divided into two modules: skip-gram (graph structure preservation module) for learning the features of the input data, and adversarial training module for improving the performance of skip-gram. Two generators are employed to generate fake neighbors for the real node pair (v_i, v_j). These fake node pairs are designed to deceive the discriminator with a high probability, while the discriminator is trained to distinguish between real and fake node pairs.

Based on the Equation, we can get

$$\frac{\partial \widetilde{L}^D_{adv}}{\partial \mathbf{v}_i} = \frac{\partial \widetilde{L}^D_{adv1}}{\partial \mathbf{v}_i} \tag{14}$$
$$= F(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i) \cdot (\mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I})),$$

and

$$\frac{\partial \widetilde{L}^D_{adv}}{\partial \mathbf{v}_j} = \frac{\partial \widetilde{L}^D_{adv2}}{\partial \mathbf{v}_j} \tag{15}$$
$$= F(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j) \cdot (\mathbf{v}'_i + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I})).$$

By combining L^D_{sgm} and $\widetilde{L}^D_{adv}$, the *novel loss function* of the discriminator D can be expressed as

$$L^D_{Nov} = L^D_{sgm} + \lambda_1 \widetilde{L}^D_{adv1} + \lambda_2 \widetilde{L}^D_{adv2}, \tag{16}$$

where λ_1 and λ_2 are the weights that control the relative importance between L^D_{sgm} and $\widetilde{L}^D_{adv}$. **Using Eqs. (14), (15), and (16), we can derive the target gradient form that aligns with Eq. (12).**

According to Eq. (13), the novel generator $\widetilde{L}^G$ can be expressed as follows:

$$\widetilde{L}^G = \min \mathbb{E}_{(i,j)\in E}(\log(1 - F(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{G,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i)) + \log(1 - F(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{G,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j))). \tag{17}$$

C. Achieving Gradient Perturbation by Tuning Weights Between Modules

In this section, we tune the weights λ_1 and λ_2 to yield the form like Eq. (8) in Theorem 6, which implies that we can achieve gradient perturbation without additional noise injection. We further show that these settings are reasonable.

Theorem 6. *Assuming that* $\lambda_1 = \frac{1}{F(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i)}$ *and* $\lambda_2 = \frac{1}{F(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j)}$, *with the gradient clipping value C, we can achieve gradient perturbation without additional noise injection when optimizing the loss function L^D_{Nov} in Eq. (16). In this case, the sensitivity of the gradient sum is BC, which implies that node-level privacy is preserved.*

Proof. According to Eq. (16), the gradient of L^D_{Nov} with respect to $\mathbf{v}_i$ is

$$\frac{\partial L^D_{Nov}}{\partial \mathbf{v}_i} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \lambda_1 \frac{\partial \widetilde{L}^D_{adv1}}{\partial \mathbf{v}_i} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \tag{18}$$
$$\lambda_1 F(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i) \cdot (\mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) + \mathbf{v}'_j).$$

Let $\lambda_1 = \frac{1}{F(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i)}$. By applying gradient clipping, we can rewrite Eq. (18) as

$$\frac{\partial L^D_{Nov}}{\partial \mathbf{v}_i} = \mathbf{clip}(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \mathbf{v}'_j) + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}). \tag{19}$$

According to Eq. (16), the gradient of L^D_{Nov} with respect to $\mathbf{v}_j$ is

$$\frac{\partial L^D_{Nov}}{\partial \mathbf{v}_j} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_j} + \lambda_2 \frac{\partial \widetilde{L}^D_{adv2}}{\partial \mathbf{v}_j} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_j} + \tag{20}$$
$$\lambda_2 F(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j) \cdot (\mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) + \mathbf{v}'_i).$$

Similarly, let $\lambda_2 = \frac{1}{F(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j)}$. By applying gradient clipping, we can rewrite Eq. (20) as

$$\frac{\partial L^D_{Nov}}{\partial \mathbf{v}_j} = \mathbf{clip}(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_j} + \mathbf{v}'_i) + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}). \tag{21}$$

As shown in Eqs. (19) and (21), we can achieve DP without additional noise.

Using $\widetilde{\nabla}_{\mathbf{v}_i} L^D_{Nov}$ and $\widetilde{\nabla}_{\mathbf{v}_j} L^D_{Nov}$ to denote the gradients of the batch samples, we have

$$\begin{aligned}\widetilde{\nabla}_{\mathbf{v}_i} L^D_{Nov} &= \frac{1}{B} \sum_{(i,j)\in E_B} \frac{\partial L^D_{Nov}}{\partial \mathbf{v}_i} \\ &= \frac{1}{B}\Big(\sum_{(i,j)\in E_B} \mathbf{clip}\Big(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \mathbf{v}'_j\Big) + \mathcal{N}_{D,1}(B^2C^2\sigma^2\mathbf{I})\Big),\end{aligned} \tag{22}$$

and

$$\begin{aligned}\widetilde{\nabla}_{\mathbf{v}_j} L^D_{Nov} &= \frac{1}{B} \sum_{(i,j)\in E_B} \frac{\partial L^D_{Nov}}{\partial \mathbf{v}_j} \\ &= \frac{1}{B}\Big(\sum_{(i,j)\in E_B} \mathbf{clip}\Big(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_j} + \mathbf{v}'_i\Big) + \mathcal{N}_{D,2}(B^2C^2\sigma^2\mathbf{I})\Big),\end{aligned} \tag{23}$$

where E_B denotes a batch sample set, and the sensitivity of both $\sum_{(i,j)\in E_B} \mathbf{clip}(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i} + \mathbf{v}'_j)$ in Eq. (22) and $\sum_{(i,j)\in E_B} \mathbf{clip}(\frac{\partial L^D_{sgm}}{\partial \mathbf{v}_j} + \mathbf{v}'_i)$ in Eq. (23) is BC. The BC is the upper bound of the gradient sum, indicating that node-level privacy is maintained. □

Remark 3. *It is worth noting that the sensitivity in Theorem 6 is not reduced compared to the naive DP-ASGM in Section III-B. However, in this paper our target is not to decrease sensitivity but rather to achieve node-level DP and enhance utility through adversarial training.*

In what follows, we will constrain the *Sigmoid* function to improve the adaptation of Theorem 6. We also show why the choice of λ_1 and λ_2 is appropriate. To simplify the discussion, we use the general notation λ to represent either λ_1 or λ_2.

Constrained *Sigmoid*. $\lambda = \frac{1}{F(\cdot)}$ denotes a *matrix*, where each of its elements is greater than 1 since $F(\cdot)$ is a *Sigmoid* function. However, λ may be ineffective when some of its elements are exceptionally small. To tackle this issue, we use a constrained *Sigmoid* function $S(x) = \frac{1}{1+\exp(-x)}$ to replace $F(\cdot)$ and $\sigma(\cdot)$, where the constrained *Sigmoid* function is achieved by constraining $\exp(\cdot)$ with Algorithm 1[2]. This algorithm enables the control of the sharpness of the corners compared with the traditional clipping method. After that, $F(\cdot)$ and $\sigma(\cdot)$ are replaced with $S(\mathbf{x}) = \frac{1}{1+[a,b]} \in [\frac{1}{1+b}, \frac{1}{1+a}]$, where $b > a > 0$.

Based on the constrained *Sigmoid*, Eq. (2), and Eq. (13), we can rewrite Eq. (16) as follows:

$$L^D_{Nov} = \widehat{L}^D_{sgm} + \lambda_1 \widehat{L}^D_{adv1} + \lambda_2 \widehat{L}^D_{adv2}, \tag{24}$$

where

$$\begin{aligned}\widehat{L}^D_{sgm} &= \log S\left(\mathbf{v}_i \cdot \mathbf{v}_j\right) + \sum_{n=1}^{k} \mathbb{E}\left[\log S\left(-\mathbf{v}^n_j \cdot \mathbf{v}_i\right)\right], \\ \widehat{L}^D_{adv1} &= \mathbb{E}_{(i,j)\in E}(-\log(1 - S(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i))), \\ \widehat{L}^D_{adv2} &= \mathbb{E}_{(i,j)\in E}(-\log(1 - S(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j))),\end{aligned}$$

[2]For more details, please refer to https://gist.github.com/patricksurry.

Algorithm 1: Exponential Clipping

Input: Input x, lower bound a, upper bound b.
Output: Clipped exponential value exp_val.

```
c_tanh = 2/(exp(2) + 1);
c = 1/(2 · c_tanh);
if a is not None and b is not None then
    c/ = (b − a)/2;
end
exp_val = max(min(x, b), a);
if a is not None then
    exp_val += exp(−c · |x − a|)/(2c);
end
if b is not None then
    exp_val -= exp(−c · |x − b|)/(2c);
end
return exp_val;
```

and according to Theorem 6, we have:

$$\lambda_1 = \frac{1}{S(\mathbf{v}_i \cdot \mathbf{v}'_j + \mathcal{N}_{D,1}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_i)},$$
$$\lambda_2 = \frac{1}{S(\mathbf{v}'_i \cdot \mathbf{v}_j + \mathcal{N}_{D,2}(C^2\sigma^2\mathbf{I}) \cdot \mathbf{v}_j)}.$$

Rationality of Weight Setting. As shown in the previous section, λ_1 and λ_2 are presented in matrix form. To evaluate the rationality of these settings, we use $\left|L^D_{Nov}\right|$ as a metric based on Eq. (24) and set $\lambda = 1$ and $\lambda = 0.5$ as two baselines, since $\lambda \in (0, 1]$ is commonly used in deep learning models. With $a = 10^{-5}$ and $b = 120$, Fig. 2 displays the loss function values for $\lambda = \frac{1}{S(\cdot)}$ across four datasets: PPI, Facebook, Wiki, and Blog (see dataset details in Section VI-A). The y-axis represents the average $\left|L^D_{Nov}\right|$ value from five independent runs. From this figure, it can be observed that the gap between $\lambda = 0.5$ and $\lambda = \frac{1}{S(\cdot)}$ is less than 6 across all tested datasets, while the gap between $\lambda = 1$ and $\lambda = \frac{1}{S(\cdot)}$ is less than 2. These small gaps indicate the rationality of the design.

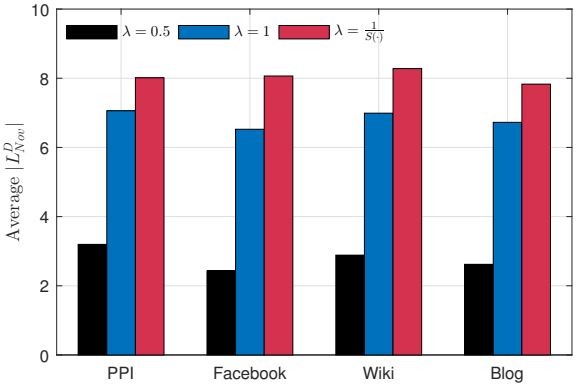

Fig. 2. Effect of weight settings across different datasets.

D. Training Algorithm

The complete training algorithm for AdvSGM is summarized in Algorithm 3. In each training epoch, we perform alternating training between the discriminator D and generator

G. Specifically, during discriminator training, we first fix Θ^G and sample $B(k+1)$ fake neighborhoods. We generate B positive samples and Bk negative samples by Algorithm 2. To compute the probability of privacy amplification more easily, we use E_B and E_{Bk} to optimize Θ^D respectively, instead of fusing B real samples E_B and Bk negative samples E_{Bk} to optimize Θ^D. Nevertheless, the utility of AdvSGM will be similar when comparing with the use of E_B and E_{Bk} simultaneously because the one-hot encoded vector is used in skip-gram models. As a result, only a fraction of the node vectors in $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ are updated. In particular, for the gradient with respect to input weight matrix, that is $\frac{\partial L^D_{sgm}}{\partial \mathbf{W}_{in}}$, it is equivalent to taking the derivative for the hidden layer, that is $\frac{\partial L^D_{sgm}}{\partial \mathbf{W}_{in}} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}_i}$. For the gradient with respect to output weight matrix, that is $\frac{\partial L^D_{sgm}}{\partial \mathbf{W}_{out}} = \frac{\partial L^D_{sgm}}{\partial \mathbf{v}^n_j}$ with negative sampling, only a fraction of the node vectors in $\mathbf{W}_{out}$ are updated. During generator training, we fix Θ^D and optimize Θ^G to generate fake neighborhoods that are close to real ones for each node under the guidance of the discriminator D. The discriminator and generator play against each other until AdvSGM converges.

Algorithm 2: Sample Generation

Input: Original graph $\mathcal{G}$.
Output: Positive samples E_B, negative samples E_{Bk}.

1 Generate a batch sample set E_B by sampling B edges uniformly at random from E;
2 Generate a batch node set $\mathcal{V}_B = \{V_1, \cdots, V_B\}$ by sampling Bk nodes uniformly at random from V, where each V_b includes k nodes;
3 Set $E_{Bk} = \{\}$;
4 **for** $(i, j) \in E_B$ **do**
5 **for** $V_b \in \mathcal{V}_B$ **do**
6 Assign (v_i, v_n) to E_{Bk} for $v_n \in V_b$;
7 **end**
8 **end**
9 **return** E_B, E_{Bk};

V. Privacy and Complexity Analysis

In this section, we provide a detailed analysis of the privacy and complexity aspects of AdvSGM.

Privacy Analysis. Following Theorem 4, we adopt the functional perspective of RDP, where ϵ is a function of α, with $1 < \alpha < \infty$, and this function is determined by the private algorithm. For ease of presentation, we replace $(\alpha, \epsilon'(\alpha))$ with $(\alpha, \epsilon^\gamma(\alpha))$ in the following proof, where γ denotes the sampling probability.

Theorem 7. *Given the number of nodes $|V|$, number of edges $|E|$, number of batch size B, and negative sampling number k, Algorithm 3 satisfies node-level $\left(\alpha, n^{epoch} n^D \epsilon^{\frac{B}{|E|}}(\alpha) + n^{epoch} n^D \epsilon^{\frac{Bk}{|V|}}(\alpha)\right)$-RDP.*

Proof. For the discriminator, the probability of generating B edges uniformly at random from E (line 1 in Algorithm 2) is $\frac{B}{|E|}$. The probability of generating Bk nodes $\mathcal{V}_B$ uniformly at random from V (line 2 in Algorithm 2) is $\frac{Bk}{|V|}$. Thus, given the known E_B, it is easy to determine that the probability of generating E_{Bk} is also $\frac{Bk}{|V|}$ (lines 4-8 in Algorithm 2). E_B and E_{Bk} are used to train the discriminator's parameters in sequence (line 7 in Algorithm 3). Theorem 6 reveals that Line 8 in Algorithm 3 achieves privacy protecting through gradient perturbation. According to the sequential composition property (Theorem 1), after $n^{epoch} n^D$ iterations, the discriminator is node-level $\left(\alpha, n^{epoch} n^D \epsilon^{\frac{B}{|E|}}(\alpha) + n^{epoch} n^D \epsilon^{\frac{Bk}{|V|}}(\alpha)\right)$-RDP. For the generator, the post-processing property (Theo-

Algorithm 3: AdvSGM Algorithm

Input: Original graph $\mathcal{G}$, number of samples B, number of training epochs n^{epoch}, number of discriminator's epochs n^D, number of generator's epochs n^G, learning rates of discriminator and generator η_d, η_g, embedding dimension r, negative sampling number k, privacy parameters ϵ, δ, and σ.
Output: Privacy-preserving embedding matrix $\Theta^D = \{\mathbf{W}_{in}, \mathbf{W}_{out}\}$.

1 Initialize discriminator parameters Θ^D, generator parameters Θ^G;
2 Normalize the parameters of the skip-gram module;
3 **for** $epoch = 0$; $epoch < n^{epoch}$ **do**
  // Train Discriminator
4 **for** $n = 0$; $n < n^D$ **do**
  // Fake neighbors
5 Sample $B(k+1)$ fake neighbors $v_i'^s$, $v_j'^s$;
  // Generate samples
6 Generate B positive samples, denoted as E_B, and Bk negative samples, denoted as E_{Bk}, uniformly at random by Algorithm 2;
7 **for** E_B *and* E_{Bk} **do**
  // Update discriminator
8 Update Θ^D according to Eq. (24) with gradient clipping like Eqs. (19) and (21), which can achieve gradient perturbation without additional noise injection (see Theorem 6);
  // Update privacy accountant of RDP
9 Calculate RDP values;
10 $\hat{\delta} \leftarrow$ get privacy spent given the target ϵ;
11 Stop optimization if $\hat{\delta} \geq \delta$;
12 **end**
13 **end**
  // Train Generator
14 **for** $n = 0$; $n < n^G$ **do**
  // Real neighbors
15 Sample $B(k+1)$ real neighbors v_i^s, v_j^s, where $(v_i, v_j) \in E$;
  // Fake neighbors
16 Generate $B(k+1)$ fake neighbors $v_i'^s$ for each node v_i;
  // Fake neighbors
17 Generate $B(k+1)$ fake neighbors $v_j'^s$ for each node v_j;
  // Update generator
18 Update Θ^G according to Eq. (17);
19 **end**
20 **end**
21 **return** $\mathbf{W}_{in}$, $\mathbf{W}_{out}$;

rem 2) ensures that the generator's privacy level aligns with that of the discriminator. Therefore, Algorithm 3 obeys node-level $\left(\alpha, n^{epoch} n^D \epsilon^{\frac{B}{|E|}}(\alpha) + n^{epoch} n^D \epsilon^{\frac{Bk}{|V|}}(\alpha)\right)$-RDP. Finally, Theorem 3 is applied to convert the RDP back to the standard node-level DP. □

Complexity Analysis. To analyze the time complexity of AdvSGM in Algorithm 3, we can break down the computations involved in each major step. The outer loop runs for n^{epoch} epochs, n^{epoch} denotes the number of training epochs. Within each epoch, the discriminator training loop, which which consists of n^D iterations (see Line 4 of Algorithm 3), samples $B(k+1)$ fake neighbors and generates $B(k+1)$ positive and negative edges for updating the discriminator parameters. The time complexity of updating the DP cost using RDP depends on the specific implementation of RDP. Different versions of RDP may result in slight differences in time complexity, but according to [19], these implementations all have asymptotic complexity of $\mathcal{O}(Br\gamma)$, where γ denotes the sampling probability. Therefore, the time complexity for updating the discriminator within one epoch can be approximated as $\mathcal{O}(n^D B(k+1)r + n^D rB\gamma)$. Within each generator training iteration, the generator loop, which consists of n^G iterations (see Line 14 of Algorithm 3), samples $B(k+1)$ real neighbors and generates $2B(k+1)$ fake neighbors for updating the generator parameters. The time complexity for updating the generator within one epoch can be approximated as $\mathcal{O}(n^G B(k+1)r)$. Therefore, the overall time complexity for the entire algorithm, running for n^{epoch} epochs, is $\mathcal{O}(n^{epoch} n^D Bkr + n^{epoch} n^D Br\gamma + n^{epoch} n^G Bkr)$. **The complexity is linear with respect to the iteration number and batch size, so our method is scalable and can be applied to large-scale networks**.

VI. Experiments

In this section, we evaluate the performance of AdvSGM in three downstream tasks: link prediction, node clustering, and node classification. Link prediction is a commonly used benchmark task in graph learning models, which tests an algorithm's ability to predict missing links in a graph, assessing how well it infers potential connections based on existing node information. Node clustering evaluates an algorithm's ability to identify and group nodes into meaningful clusters based on their similarities, assessing how well it discovers inherent group structures. Node classification is the task of predicting the labels or categories of nodes in a graph, based on their features and relationships with other nodes, evaluating how well the algorithm can generalize from labeled data to unseen instances. We aim to address the following four questions:

- How much do the parameters impact the performance of AdvSGM? (see Section VI-B)
- How is the performance of different differentially private skip-gram models? (see Section VI-C)
- How much does the privacy budget influence the performance of AdvSGM and other private graph models in link prediction? (see Section VI-D)
- How much does the privacy budget influence the performance of AdvSGM and other private graph models in node clustering? (see Section VI-E)

A. Experimental Setup

Datasets. To comprehensively evaluate our proposed method, we conduct extensive experiments on the following six real-world graph datasets: namely PPI, Facebook, Wiki, Blog, Epinions, DBLP. Since we focus on simple graphs in this work, all datasets are pre-processed to remove self-loops. The details of the datasets are provided as follows.

- **PPI [20]:** This dataset represents a human Protein-Protein Interaction network, consisting of 3,890 nodes from 50 classes and 76,584 edges. The nodes represent proteins, and the edges indicate interactions between these proteins.
- **Facebook**[3]**:** This dataset represents a social network with 4,039 nodes and 88,234 edges. The nodes represent users, and the edges represent the relationships between them.
- **Wiki**[4]**:** This dataset consists of a network of hyperlinks between Wikipedia pages, with 4,777 nodes from 40 categories and 92,517 edges. Each node represents a Wikipedia page, and each edge represents a hyperlink between two pages.
- **Blog**[5]**:** This dataset is an online social network containing 10,312 nodes from 39 categories and 333,983 edges. The nodes represent users, and the edges represent relationships between these users.
- **Epinions**[6]**:** The Epinions dataset is a trust network with 75,879 nodes and 508,837 edges. The nodes represent users, and the directed edges represent trust relationships between them.
- **DBLP**[7]**:** This dataset represents a scholarly network with 2,244,021 nodes and 4,354,534 edges. The nodes represent papers, authors, and venues, and the edges represent authorships and the venues where papers are published.

Evaluation Metrics. We evaluate the performance of AdvSGM by testing its effectiveness across three downstream tasks: link prediction, node clustering, and node classification.

- For the link prediction task, all existing links in each dataset are randomly split into a training set 90% and a test set 10%. For the test set, we sample the same number of node pairs without connected edges as negative test links to evaluate link prediction performance. For the training set, we additionally sample the same number of node pairs without edges to construct negative training data. AUC is used to measure performance.
- For the node clustering task, we feed the embedding vectors generated by each algorithm into a node clustering algorithm. Following [21], we adopt the Affinity

[3]https://snap.stanford.edu/data/ego-Facebook.html
[4]https://www.mattmahoney.net/dc/text.html
[5]http://datasets.syr.edu/datasets/BlogCatalog3.html
[6]https://snap.stanford.edu/data/soc-Epinions1.html
[7]https://www.aminer.cn/citation

Propagation algorithm [22] as the clustering method and evaluate the clustering results in terms of mutual-information (MI).

- For the node classification task, we randomly sample 90% of the nodes as training data and randomly sample 10% of the nodes outside the training set as test data. We follow the procedure of [23] and evaluate our embeddings using Micro-F1 score.

For each result, we measure it over five experiments to report the average value. **The larger the AUC, MI, and Micro-F1 scores, the better the utility of the algorithm.**

Competitive Methods. To establish a baseline for comparison, we utilize four state-of-the-art private graph learning methods, namely DPGGAN [24], DPGVAE [24], GAP [25], and DPAR [26]. In this study, we simulate a scenario where the graphs only contain structural information, while GAP and DPAR rely on node features. To ensure a fair evaluation, similar to prior research [27], we use randomly generated features as inputs for GAP and DPAR. Additionally, we design different versions of skip-gram for comparison: SGM (No DP), DP-SGM and DP-ASGM. Here, SGM (No DP) refers to the original skip-gram model (i.e., LINE [2]), DP-SGM denotes the skip-gram model with DPSGD, and DP-ASGM represents the skip-gram model with adversarial training based on DPSGD. Note that skip-gram (see Eq. (2)) incorporates two vectors for each node, namely a context (output) vector and a node (input) vector. However, during our testing phase, we do not observe any performance improvement by utilizing both vectors together. We only employ the node vectors for our experiments, similar to previous methods [2], [28].

Parameter Settings. In the link prediction and node clustering tasks, we use training epochs $n^{epoch} = 50$ for each dataset. In each epoch, we set $n^D = 15$ and $n^G = 5$ for both tasks. We set the embedding dimension $r = 128$. It is worth noting that we do not specifically highlight the impact of r as it is commonly used in various graph embedding methods [1], [2], [29], [30]. To maintain compatibility with most skip-gram based methods [2], [27], [30], we set $k = 5$. We normalize the parameters of skip-gram module in AdvSGM to ensure that $C = 1$. For privacy parameters, we follow existing works [7], [24], [31] to fix $\sigma = 5$. Then, we vary the privacy budget ϵ among the values $\{1, 2, 3, 4, 5, 6\}$ to see how much utilities are preserved under different privacy budgets. Also, we vary the learning rates η_d, η_g, the batch size B, and the privacy parameter δ to verify the effect on the utility of AdvSGM. We fix $a = 10^{-5}$ to ensure that the upper bound of $S(\mathbf{x})$ approaches 1, and vary b to verify the effect on the utility of AdvSGM. To ensure consistency with the original papers, we utilize the official GitHub implementations for DPGGAN, DPGVAE, GAP and DPAR. We replicate the experimental setup as described in those papers.

B. Impact of Parameters

In this section, taking link prediction as an example, we investigate the effects of different parameters on the performance of AdvSGM. We conduct experiments on all datasets with varying the learning rate η from 0.01 to 0.3, the batch size B from 16 to 512, and the parameters b and δ from 40 to 140 and from 10^{-5} to 10^{-7}, respectively.

1) Parameter η: In this experiment, we examine the influence of the learning rate on the performance of AdvSGM in the context of link prediction task. We conduct experiments on three datasets: PPI, Facebook, and Blog, with different values of the learning rates η_d and η_g: 0.01, 0.05, 0.1, 0.15, 0.2, 0.25 and 0.3. The results are summarized in Table II. From the table, we can observe that the best performance is achieved when $\eta_d = \eta_g = 0.1$ for the tested three datasets. Therefore, we set $\eta_d = \eta_g = 0.1$ as the default parameter configuration for all subsequent experiments. Additionally, we consistently find that all standard deviations are no more than 0.02 across all datasets, indicating that AdvSGM exhibits good stability.

2) Parameter B: In this experiment, we investigate the impact of the parameter B on the performance of AdvSGM in terms of link prediction task. Specifically, we consider different values of B, namely, $16, 32, 64, 128, 256, 512$ for all datasets. As illustrated in Table III, for PPI and Facebook, the optimal results are achieved when $B = 128$. For Blog, while the AUC performs better when $B = 512$, using $B = 128$ and $B = 256$ can still yield competitive results in terms of AUC. Therefore, we set $B = 128$ as the default parameter configuration for all subsequent experiments. Also, we consistently observe that all standard deviations remain consistently below 0.02 across all datasets, indicating that AdvSGM is very stable.

3) Parameter b: Recall from Section IV-C that both λ_1 and λ_2 are defined as $\frac{1}{S(\cdot)}$, where $S(\cdot)$ is constrained within the range $[\frac{1}{1+b}, \frac{1}{1+a}]$. In this experiment, we set a to 10^{-5} to ensure that $S(\cdot)$ converges to 1. With b varying from 40 to 140, Table IV illustrates the impact of parameter b on AdvSGM for link prediction. As b increases, there is a gradual improvement in performance, and the optimal results across all datasets are achieved when $b = 140$. However, the performance gain for $b = 140$ compared to $b = 120$ is minimal. Therefore, we choose $b = 120$ as the default parameter configuration for all subsequent experiments.

4) Parameter δ: In differential privacy, δ is a critical parameter that allows for a small, controlled probability of privacy failure. We vary the $\delta \in \{10^{-5}, 10^{-6}, 10^{-7}\}$ to explore its impact. The experiment results are presented in the following Table. In summary, AdvSGM with $\delta = 10^{-5}$ achieves better utility in terms of AUC values. However, AdvSGM with $\delta = 10^{-6}$ and AdvSGM with $\delta = 10^{-7}$ produce results that are very close to those of AdvSGM with $\delta = 10^{-5}$. This indicates that setting δ to 10^{-5} is reasonable choice, as it provides a better balance between privacy and utility.

C. Comparison Between Private Skip-gram Models

In this section, we compare different private skip-gram models. Table VI presents the AUC and MI results for all the methods. From this table, we make three important observations. First, AdvSGM(No DP) achieves better utility on all tested datasets compared to SGM(No DP), which indicates that the adversarial training module can effectively

TABLE II
SUMMARY OF AUC VALUES WITH DIFFERENT η_d AND η_g, GIVEN $\epsilon = 6$ (RESULT: AVERAGE AUC $\pm$ STANDARD DEVIATION). **BOLD**: BEST

$\eta_d = \eta_g$	PPI	Facebook	Blog
0.01	0.5327±0.0092	0.6289±0.0024	0.5393±0.0065
0.05	0.5753±0.0050	0.6834±0.0010	0.5623±0.0081
0.1	**0.6095±0.0101**	**0.7070±0.0064**	**0.6595±0.0200**
0.15	0.5687±0.0096	0.6478±0.0048	0.5876±0.0055
0.2	0.5246±0.0107	0.6057±0.0021	0.5368±0.0023
0.25	0.5264±0.0044	0.5673±0.0027	0.5216±0.0043
0.3	0.5224±0.0077	0.5471±0.0029	0.5147±0.0004

TABLE III
SUMMARY OF AUC VALUES WITH DIFFERENT B, GIVEN $\epsilon = 6$ (RESULT: AVERAGE AUC $\pm$ STANDARD DEVIATION). **BOLD**: BEST

B	PPI	Facebook	Blog
16	0.5057±0.0087	0.5273±0.0042	0.5052±0.0010
32	0.5250±0.0040	0.5602±0.0067	0.5142±0.0053
64	0.5600±0.0019	0.6403±0.0024	0.5465±0.0049
128	**0.6095±0.0101**	**0.7070±0.0064**	0.6595±0.0200
256	0.5676±0.0069	0.6599±0.0031	0.6625±0.0076
512	0.5077±0.0056	0.5286±0.0053	**0.6722±0.0154**

improve the performance of SGM(No DP). Second, AdvSGM significantly outperforms the other private models, DP-SGM and DP-ASGM, across all privacy budgets and datasets in terms of both AUC and MI. This suggests that our proposed adversarial training module, which introduces two additional noise terms into the activation functions, is more effective than the direct perturbation approach based on DPSGD used in DP-SGM and DP-ASGM. **Another important observation from Table VI is that as ϵ increases, AdvSGM tends to produce results comparable to those of the non-private SGM. Notably, at $\epsilon = 6$, AdvSGM even surpasses the non-private SGM, indicating that our adversarial training module not only preserves the privacy of the skip-gram but also enhances its utility.** Overall, the findings clearly demonstrate the advantages of AdvSGM over other privacy-preserving skip-gram models, highlighting its ability to achieve a better balance between privacy and utility.

TABLE IV
SUMMARY OF AUC VALUES WITH DIFFERENT b, GIVEN $\epsilon = 6$ (RESULT: AVERAGE AUC $\pm$ STANDARD DEVIATION). **BOLD**: BEST

b	PPI	Facebook	Blog
40	0.5053±0.0033	0.5261±0.0044	0.5087±0.0025
60	0.5203±0.0054	0.5557±0.0052	0.5142±0.0037
80	0.5302±0.0036	0.6022±0.0019	0.5424±0.0107
100	0.5694±0.0063	0.6627±0.0029	0.5905±0.0129
120	0.6095±0.0101	0.7070±0.0064	0.6595±0.0200
140	**0.6584±0.0140**	**0.7434±0.0057**	**0.7157±0.0175**

TABLE V
SUMMARY OF AUC VALUES WITH DIFFERENT δ, GIVEN $\epsilon = 6$ (RESULT: AVERAGE AUC $\pm$ STANDARD DEVIATION). **BOLD**: BEST

δ	PPI	Facebook	Blog
10^{-5}	**0.6095±0.0101**	**0.7070±0.0064**	**0.6595±0.0200**
10^{-6}	0.5750±0.0021	0.6598±0.0025	0.6154±0.0088
10^{-7}	0.5740±0.0067	0.6515±0.0052	0.6105±0.0061

TABLE VI
SUMMARY OF AUC/MI VALUES WITH DIFFERENT ϵ. BOLD: BEST; UNDERLINED: SECOND-BEST

Algorithms	AUC			MI	
	PPI	Facebook	Blog	PPI	Blog
SGM(No DP)	0.5924	0.6447	0.6214	0.7385	0.5363
AdvSGM(No DP)	**0.6914**	**0.7588**	**0.7078**	**1.1927**	**0.9942**
DP-SGM(ϵ=1)	0.5063	0.5077	0.5030	0.5768	0.3869
DP-ASGM(ϵ=1)	0.5077	0.5071	0.5036	0.5798	0.4530
AdvSGM(ϵ=1)	**0.5083**	**0.5398**	**0.5187**	**0.5810**	**0.5284**
DP-SGM(ϵ=2)	0.5076	0.5040	0.5027	0.5704	0.4654
DP-ASGM(ϵ=2)	0.5078	0.5069	0.5028	0.5820	0.4731
AdvSGM(ϵ=2)	**0.5152**	**0.5459**	**0.5964**	**0.5847**	**0.6787**
DP-SGM(ϵ=3)	0.5095	0.5033	0.5027	0.5686	0.4510
DP-ASGM(ϵ=3)	0.5047	0.5052	0.5054	0.6130	0.4601
AdvSGM(ϵ=3)	**0.5271**	**0.5779**	**0.6593**	**0.6332**	**0.7921**
DP-SGM(ϵ=4)	0.5057	0.5047	0.5020	0.7200	0.4679
DP-ASGM(ϵ=4)	0.5061	0.5066	0.5023	0.7225	0.4560
AdvSGM(ϵ=4)	**0.5655**	**0.6368**	**0.6629**	**0.7374**	**0.7926**
DP-SGM(ϵ=5)	0.5074	0.5085	0.5022	0.5512	0.3663
DP-ASGM(ϵ=5)	0.5096	0.5097	0.5040	0.6189	0.4549
AdvSGM(ϵ=5)	**0.5878**	**0.6829**	**0.6688**	**0.9038**	**0.7978**
DP-SGM(ϵ=6)	0.5077	0.5054	0.5020	0.5851	0.3711
DP-ASGM(ϵ=6)	0.5084	0.5079	0.5024	0.6140	0.3668
AdvSGM(ϵ=6)	**0.6095**	**0.7070**	**0.6595**	**1.0818**	**0.8777**

D. Impact of Privacy Budget on Link Prediction

We compare the AUC result of different methods under six privacy budgets: 1, 2, 3, 4, 5, and 6. The AUC results of all methods are illustrated in Fig. 3. From this figure, we can see that AdvSGM are all significantly better than other private models in all privacy budgets across all datasets. This main reason is that our designed adversarial training module enhances the utility of skip-gram without compromising privacy. For DPAR, despite achieving higher AUC scores than DPGGAN, DPGVAE, and GAP across all datasets and privacy budgets, it still falls short of AdvSGM in terms of AUC score. DPGGAN and DPGVAE use the MA mechanism [7] to address the issue of excessive privacy budget splitting during optimization, but they still produce poor results. This occurs because these methods often converge prematurely under MA, especially with a limited privacy budget, leading to reduced performance in both privacy and utility. GAP employs the aggregation perturbation (AP) technique to ensure differential privacy in GNNs, yet it also yields poor results. The main drawback of AP is its incompatibility with standard GNN architectures due to the high privacy costs involved. Conventional GNN models frequently query aggregation functions with each parameter update, requiring the re-perturbation of all aggregate outputs in every training iteration to maintain differential privacy. This process results in a significant increase in privacy costs.

E. Impact of Privacy Budget on Node Clustering

The MI results for node clustering are shown in Fig. 4. Note that we only evaluate MI on the PPI, Wiki, and Blog datasets, due to the absence of labeled data in the other datasets. From this figure, a key observation is that our AdvSGM consistently achieves the highest MI accuracy among all private methods. In conclusion, the results from these experiments strongly

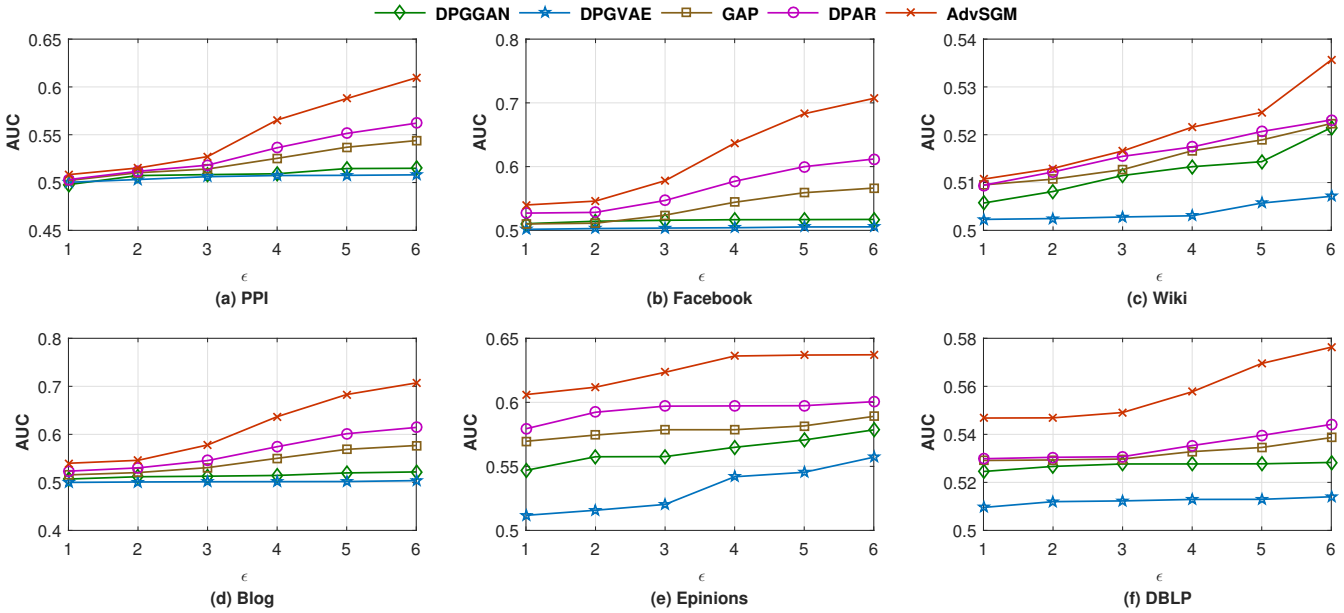

Fig. 3. Impact of Privacy Budget on Link Prediction.

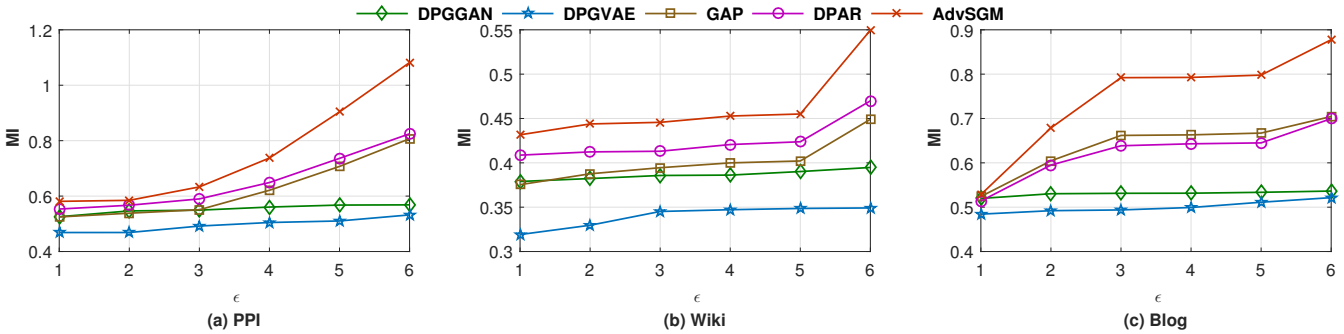

Fig. 4. Impact of Privacy Budget on Node Clustering.

support our claim. The adversarial training component in AdvSGM enables it to strike an effective balance between privacy preservation and task utility. As such, AdvSGM proves to be a highly effective and versatile privacy-preserving approach that can be readily applied to a variety of downstream graph-based tasks, offering significant benefits in scenarios where both privacy and performance are critical considerations.

F. Impact of Privacy Budget on Node Classification

The Micro-F1 results for node clustering are presented in Fig. 5. As with the node clustering task, we evaluate Micro-F1 only on the PPI, Wiki, and Blog datasets, since the other datasets lack labeled data. The results clearly show that AdvSGM outperforms other privacy methods across all evaluated datasets in terms of Micro-F1. This further highlights AdvSGM as an exceptional off-the-shelf solution for a wide range of downstream graph tasks.

VII. Related Work

Related work of this paper includes differentially private deep learning and differentially private graph learning.

Private Deep Learning. DP is a crucial framework for safeguarding the privacy of individual data during the training of machine learning models, particularly in deep learning, where large-scale datasets and complex models are prevalent. Early work by Dwork *et al.* [32] introduced the concept of strong composition, which aims to provide privacy guarantees when combining multiple queries or updates. However, as pointed out by Abadi *et al.* [7], existing composition theorems lack the necessary precision to accurately evaluate the privacy cost in deep learning models. To address this issue, the MA mechanism was introduced, which tracks the logarithmic moments of privacy loss variables and provides more accurate privacy loss estimates when combining Gaussian mechanisms under random sampling. Mironov *et al.* [17] propose a new analysis method named RDP, which surpasses the MA mechanism and greatly enhances DPSGD performance. Further research [33]–[36] explores modifications to model structures or learning algorithms, such as replacing traditional activation functions in CNNs with smoother *Sigmoid* functions. Other studies [37], [38] focus on optimizations like adaptive gradient clipping

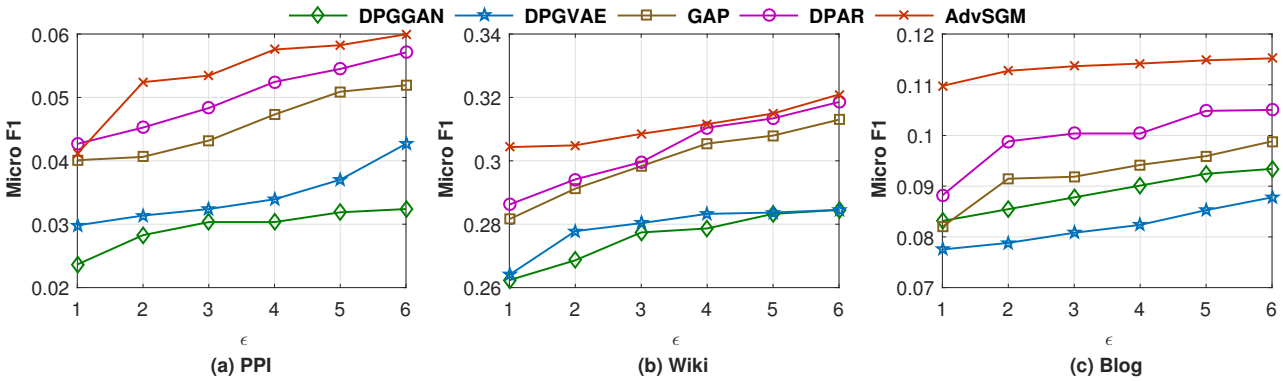

Fig. 5. Impact of Privacy Budget on Node Classification.

bounds or dynamic privacy budget partitioning.

Private Graph Learning. The most closely related work is by Ahuja *et al.* [6], which combines SGM and DPSGD [7] for private learning from sparse location data. However, this method doesn't apply to graphs due to noise from complex node relationships. Peng *et al.* [39] propose a decentralized framework for privacy-preserving learning of embeddings from multiple knowledge graphs. Han *et al.* [40] develop differentially private knowledge graph embeddings. Pan *et al.* [41] present a federated framework for unsupervised node embedding with DP and high communication efficiency. Despite these advancements, like Ahuja *et al.*, these methods face utility issues due to their perturbation mechanisms. Yang *et al.* [7] develop differentially private GAN and VAE models for graph synthesis and link prediction, but these approaches often converge prematurely with limited privacy budgets, reducing both privacy and utility. Epasto *et al.* [23] present a differentially private graph learning algorithm that outputs an approximate personalized PageRank and have provably bounded sensitivity to input edges. Unfortunately, the proposed private PageRank only achieves the weak edge-level DP. Another area of research in differential private graph learning focuses on GNNs [25], [26], [42]–[45]. The aggregation perturbation technique is often used to ensure DP in GNNs. Unlike traditional DPSGD algorithms, many differentially private GNN methods perturb the aggregate information from the GNN neighborhood aggregation step. Yet, the aggregation perturbation faces compatibility issues with standard GNN architectures, requiring the re-perturbation of all aggregate outputs at each training iteration, which leads to high privacy costs.

VIII. Conclusion and Future Work

In this paper, we have presented a differentially private skip-gram for graphs via adversarial training, called AdvSGM. The main features lie in two aspects. First, we design a novel adversarial training module by introducing two optimizable noise terms in activation functions. Second, we achieve DP during optimization by fine-tuning the weighs between modules. Furthermore, we demonstrate that AdvSGM obeys node-level differential privacy. Extensive experiments on real-world graph datasets demonstrate that our solution outperforms state-of-the-art competitors. In our future work, we plan to extend our method to attribute graphs. It is worth noting that the attributes associated with the nodes are independent and can be easily managed due to their low sensitivity. Additionally, we also plan to expand adversarial training to matrix factorization-based network embeddings. Specifically, we will focus on the common matrix decomposition problem of minimizing $\|\mathbf{S} - \mathbf{U}\mathbf{V}^\top\|_F^2$, where $\mathbf{S}$ represents the node similarity matrix of the graph. This approach can be integrated with our adversarial training module and Theorem 6. In this context, we only need to modify the equation to the maximization form: $\max_{\mathbf{U},\mathbf{V}} \log_a \|\mathbf{S} - \mathbf{U}\mathbf{V}^\top\|_F^2$, where $0 < a < 1$.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[2] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *ACM International Conference on World Wide Web*, 2015, pp. 1067–1077.

[3] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[4] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *ACM International Conference on Information and Knowledge Management*, 2008, pp. 289–298.

[5] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science.*, vol. 9, no. 3–4, pp. 211–407, 2014.

[6] R. Ahuja, G. Ghinita, and C. Shahabi, "Differentially-private next-location prediction with neural networks," in *International Conference on Extending Database Technology*, 2020, pp. 121–132.

[7] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[9] J. Li, X. Fu, S. Zhu, H. Peng, S. Wang, Q. Sun, S. Y. Philip, and L. He, "A robust and generalized framework for adversarial graph embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11 004–11 018, 2023.

[10] J. Tang, M. Qu, and Q. Mei, "PTE: Predictive text embedding through large-scale heterogeneous text networks," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1165–1174.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, 2006, pp. 265–284.

[12] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *IEEE International Conference on Data Mining*, 2009, pp. 169–178.

[13] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages, and Programming*, 2006, pp. 1–12.

[14] I. Mironov, "Rényi differential privacy," in *IEEE Computer Security Foundations Symposium*, 2017, pp. 263–275.

[15] Y. Wang, B. Balle, and S. P. Kasiviswanathan, "Subsampled rényi differential privacy and analytical moments accountant," in *International Conference on Artificial Intelligence and Statistics*, 2019, pp. 1226–1235.

[16] Y. Zhu and Y. Wang, "Poission subsampled rényi differential privacy," in *International Conference on Machine Learning*, 2019, pp. 7634–7642.

[17] I. Mironov, K. Talwar, and L. Zhang, "Rényi differential privacy of the sampled gaussian mechanism," *arXiv preprint arXiv:1908.10530*, 2019.

[18] Z. He, T. Zhang, and R. B. Lee, "Model inversion attacks against collaborative inference," in *Proceedings of the Annual Computer Security Applications Conference*, 2019, pp. 148–162.

[19] Z. Bu, Y.-X. Wang, S. Zha, and G. Karypis, "Differentially private optimization on large model at small cost," in *International Conference on Machine Learning*, 2023, pp. 3192–3218.

[20] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, "BioGRID: A general repository for interaction datasets," *Nucleic Acids Research*, vol. 34, pp. D535–D539, 2006.

[21] D. Nguyen, W. Luo, T. D. Nguyen, S. Venkatesh, and D. Phung, "Learning graph representation via frequent subgraphs," in *SIAM International Conference on Data Mining*, 2018, pp. 306–314.

[22] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[23] A. Epasto, V. Mirrokni, B. Perozzi, A. Tsitsulin, and P. Zhong, "Differentially private graph learning via sensitivity-bounded personalized pagerank," in *International Conference Neural Information Processing Systems*, 2022, pp. 22 617–22 627.

[24] C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun, "Secure deep graph generation with link differential privacy," in *International Joint Conference on Artificial Intelligence*, 2021, pp. 3271–3278.

[25] S. Sajadmanesh, A. S. Shamsabadi, A. Bellet, and D. Gatica-Perez, "GAP: Differentially private graph neural networks with aggregation perturbation," in *USENIX Security Symposium*, 2023, pp. 3223–3240.

[26] Q. Zhang, H. k. Lee, J. Ma, J. Lou, C. Yang, and L. Xiong, "DPAR: Decoupled graph neural networks with node-level differential privacy," in *Proceedings of the ACM on Web Conference*, 2024, pp. 1170–1181.

[27] L. Du, X. Chen, F. Gao, Q. Fu, K. Xie, S. Han, and D. Zhang, "Understanding and improvement of adversarial training for network embedding from an optimization perspective," in *ACM International Conference on Web Search and Data Mining*, 2022, pp. 230–240.

[28] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[29] Y. Lai, C. Hsu, W. H. Chen, M. Yeh, and S. Lin, "PRUNE: Preserving proximity and global ranking for network embedding," in *International Conference Neural Information Processing Systems*, 2017, pp. 5257–5266.

[30] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin, "A unified framework for community detection and network representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1051–1065, 2018.

[31] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1310–1321.

[32] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *IEEE Annual Symposium on Foundations of Computer Science*, 2010, pp. 51–60.

[33] C. Chen and J. Lee, "Stochastic adaptive line search for differentially private optimization," in *IEEE International Conference on Big Data*. IEEE, 2020, pp. 1011–1020.

[34] M. Nasr, R. Shokri, and A. Houmansadr, "Improving deep learning with differential privacy using gradient encoding and denoising," *arXiv preprint arXiv:2007.11524*, 2020.

[35] N. Papernot, A. Thakurta, S. Song, S. Chien, and Ú. Erlingsson, "Tempered sigmoid activations for deep learning with differential privacy," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9312–9321.

[36] F. Tramer and D. Boneh, "Differentially private learning needs better features (or much more data)," *arXiv preprint arXiv:2011.11660*, 2020.

[37] L. Xiang, J. Yang, and B. Li, "Differentially-private deep learning from an optimization perspective," in *IEEE INFOCOM Conference on Computer Communications*, 2019, pp. 559–567.

[38] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *IEEE Symposium on Security and Privacy*, 2019, pp. 332–349.

[39] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, "Differentially private federated knowledge graphs embedding," in *ACM International Conference on Information and Knowledge Management*, 2021, pp. 1416–1425.

[40] X. Han, D. Dell'Aglio, T. Grubenmann, R. Cheng, and A. Bernstein, "A framework for differentially-private knowledge graph embeddings," *Journal of Web Semantics*, vol. 72, p. 100696, 2022.

[41] Q. Pan and Y. Zhu, "FedWalk: Communication efficient federated unsupervised node embedding with differential privacy," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1317–1326.

[42] I. E. Olatunji, T. Funke, and M. Khosla, "Releasing graph neural networks with differential privacy guarantees," *Transactions on Machine Learning Research*, pp. 2835–8856, 2023.

[43] A. Daigavane, G. Madan, A. Sinha, A. G. Thakurta, G. Aggarwal, and P. Jain, "Node-level differentially private graph neural networks," *arXiv preprint arXiv:2111.15521*, 2021.

[44] S. Sajadmanesh and D. Gatica-Perez, "ProGAP: Progressive graph neural networks with differential privacy guarantees," in *ACM International Conference on Web Search and Data Mining*, 2024, pp. 596–605.

[45] Z. Xiang, T. Wang, and D. Wang, "Preserving node-level privacy in graph neural networks," in *IEEE Symposium on Security and Privacy*, 2024, pp. 4714–4732.