Adversarial Privacy Preserving Graph Embedding against Inference Attack

Kaiyang Li, Guangchun Luo, Yang Ye, Wei Li, *Member, IEEE*, Shihao Ji, Zhipeng Cai, *senior Member, IEEE*

***Abstract*—Recently, the surge in popularity of Internet of Things (IoT), mobile devices, social media, etc. has opened up a large source for graph data. Graph embedding has been proved extremely useful to learn low-dimensional feature representations from graph structured data. These feature representations can be used for a variety of prediction tasks from node classification to link prediction. However, existing graph embedding methods do not consider users' privacy to prevent inference attacks. That is, adversaries can infer users' sensitive information by analyzing node representations learned from graph embedding algorithms. In this paper, we propose Adversarial Privacy Graph Embedding (APGE), a graph adversarial training framework that integrates the disentangling and purging mechanisms to remove users' private information from learned node representations. The proposed method preserves the structural information and utility attributes of a graph while concealing users' private attributes from inference attacks. Extensive experiments on real-world graph datasets demonstrate the superior performance of APGE compared to the state-of-the-arts. Our source code can be found at https://github.com/uJ62JHD/Privacy-Preserving-Social-Network-Embedding.**

***Index Terms*—data privacy, graph embedding, inference attack, adversarial learning**

I. Introduction

Nowadays, an unprecedented volume of data generated by computer networks and services have a graph structure. For example, from network structure to control flow graph, IoT naturally generates a large amount of graph data continuously. IoT expands human being's abilities to understand and control the physical world and grows increasingly in popularity. By the end of 2018, there were an estimated 22 billion IoT connected devices in use around the world [1]. IoT applications and services span almost all the economic and social sectors, including environment monitoring, smart industry, public safety, smart medical systems, smart grid, smart agriculture, smart transportation, behavioral patterns, etc. [2]–[4]. Another major source for graph data is social network, where people/mobile devices correspond to nodes and links represent the relationships of nodes. People use social networks to converse and connect with people sharing similar interests in the real world [5]–[7]. Social media has become a primary option for people to connect to new friends and interact with their current friends. Currently, about 2.95 billion people have social media accounts worldwide and this number will increase to almost 3.43 billion in 2023 [8].

Graph embedding has made prominent progress in graph analysis in the past few years [9] [10]. It's a technique that learns a low-dimensional representation for each node of a graph based on network topology and nodes' attributes. There are several advantages of using graph embedding for graph analysis: (1) Through this technique, the downstream applications such as node classification [11], clustering [12], link prediction [13] and graph visualization [14] can be performed on standard machine learning algorithms whose input must be vectors. (2) Many graph mining algorithms contain iterative or combinational computations, whose complexities are typically NP-hard [15]. By graph embedding, the computation of these tasks can be reduced dramatically. (3) It's very challenging to design parallel or distributed algorithms on graph data directly [16]; more efficient parallel and distributed algorithms can be developed readily on learned node representations. However, the existing graph embedding algorithms do not consider users' privacy to prevent inference attacks. That is, from the learned node representation, adversaries can infer the sensitive information that users have no intent to disclose originally [17].

On the one hand, graph data owners, such as IoT service providers and social networks, collect a huge amount of graph data and publish data to the third parties for research and business purposes. The graph data can be exploited by third party analysts to predict users' purchase interest [18], discover trending events [19] and so on. However, the third parties may also use the sensitive information, such as sexual orientation and political tendency, for malicious acts. Worse, even if these sensitive attributes are deleted from graph, adversaries can still mine the privacy information via inference attacks [20] [21]. For examples, Social IoT topology and users' behavior data can be used to infer users' geolocations [22], and users' sexual orientation could be inferred based on their Facebook data, such as users' linkages, genders, and other attributes [23] [24]. All of these demonstrate that our graph data is facing a serious privacy issue. Since the vector representations of users contain a lot of individual information, adversaries could also launch an inference attack by mining the information from the embeddings. E.g., a social network, like Facebook, publishes the embeddings of users for academic research. An adversary obtains the released embeddings and meanwhile collects part of the users' political tendency by trading [25] [26], crawling the network [27], or utilizing other resources. If the adversary trains a classifier on the embeddings and the collected political tendency, the adversary could infer the other users' political

K. Li and G. Luo are with school of computer science and engneering, University of Electronic Science and Technology of China, Chengdu 600731,China (e-mail: kaiyang.li@outlook.com; gcluo@uestc.edu.cn).

Y. Ye, W. Li, S. Ji, Z. Cai are with the Department of Computer Science, Georgia State University, Atlanta, GA 30302 USA (e-mail: yye10@student.gsu.edu; wli28@gsu.edu; sji@gsu.edu; zcai@gsu.edu).

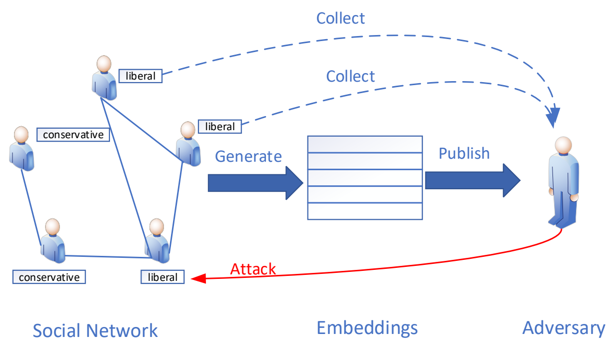


Fig. 1: A example of inference attack on Graph Embedding

tendency via the classifier, as shown in Fig. 1. Therefore, before the data owners publish their graph embedding, they must preprocess their data to prevent inference attacks.

On the other hand, existing privacy protection methods typically prevent inference attacks by pruning the graph data directly [22] [28] [29]. The subsequent graph embedding on such preprocessed networks will induce suboptimal performance for privacy protection and data utility reservation. This is because: (1) Graph embedding and privacy protection are decoupled in this process and limit the expressiveness of the model. (2) Existing methods have limited options to preprocess users' attributes and links. For example, we can either swap gender attribute of a user or delete it from the node completely to conceal users' private information. With graph embedding, users' gender information will be embedded in a continuous space and therefore we can fine-tune the latent space through gradient updates to protect users' privacy precisely. (3) Traditional graph privacy-preserving methods only consider the first-order friendship, while graph embedding concerns global graph topology. Therefore, high-order friendship can also be used to infer users' sensitive information. For instance, two users don't have direct connections, but they may have shared friends; if only the first-order friendship is utilized, these two users might be considered irrelevant; but they are closely related to each other due to the high-order friendship, which means their private attributes might be similar with a high probability.

What's more, the only existing graph embedding privacy preserving works [30] [31] try to achieve differential privacy on link information. Let $\mathbf{G_1}$ and $\mathbf{G_2}$ be the neighbor graphs differing by an edge. $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are the embedding matrix derived from $\mathbf{G_1}$ and $\mathbf{G_2}$, respectively. These works aim to ensure the statistical difference on $\mathbf{Z}_1$ and $\mathbf{Z}_2$ is smaller than a predefined bound. Since the purpose of the differentially private mechanisms is not against inference attacks, these mechanisms could not defend inference attacks on sensitive attribute well.

Therefore, in this paper we propose a privacy-preserving graph embedding framework, which integrates graph embedding and privacy protection into an end-to-end pipeline against inference attack. As we will demonstrate, our method enables graph to release their privacy-preserved node embeddings to the third parties with reduced risks of privacy concern. The contributions of this paper are

- To the best of our knowledge, this is the first graph embedding algorithm against inference attack.
- To hide the sensitive information from graph embeddings, we propose two mechanisms: Privacy-Disentangling and Privacy-Purging from different perspectives.
- We propose Adversarial Privacy Graph Embedding (APGE), an adversarial training based graph embedding algorithm that integrates the mechanisms Privacy-Disentangling and Privacy-Purging to preserve users' privacy in an end-to-end graph convolution pipeline.
- Extensive experiments on graph datasets demonstrate the superior performance of our method over previous approaches in terms of private protection and users' utility reservation.

The rest of the paper is organized as follows. Section 2 presents the problem definition and preliminary knowledge. Section 3 introduces our proposed methods. The experimental evaluations and comparison results are reported in Section 4. Section 5 briefly summarizes the related works on graph embedding and graph privacy preserving. Section 6 concludes the paper.

II. Problem Definition and Preliminary Knowledge

In this section, we will introduce the problem definitions and preliminary knowledge of this work.

A. Problem Definition

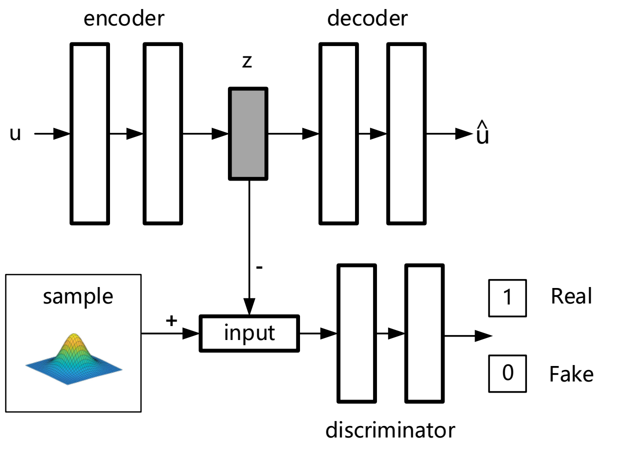
A graph can be defined by $\mathbf{G}(\mathbf{V}, \mathbf{E}, \mathbf{X})$, where $\mathbf{V}$ denotes the set of N nodes, $\mathbf{E}$ denotes the set of all edges, and $\mathbf{X} \in \mathbb{R}^{N\times P}$ is the attribute matrix, each row of which representing the feature vector of a node. Moreover, we introduce $\mathbf{A} \in \mathbb{R}^{N\times N}$ as the adjacent matrix of $\mathbf{G}$. In this paper, we use capital variables (e.g., $\mathbf{X}$) to denote matrices and lower-case variables (e.g., $\mathbf{x}$) to denote row vectors. Accordingly, we use $\mathbf{x}_i$ to denote the i-th row of the matrix $\mathbf{X}$.

Attack model. We assume the adversary could obtain the released graph embeddings $\mathbf{Z}$, and the set of partial users' private attribute labels $\mathcal{Y}_k^p$. Specifically, $\mathcal{Y}_k^p$ could be captured by trading [25], crawling [27], hacking [32], etc. The adversary trains a classifier on $\mathbf{Z}$ and $\mathcal{Y}_k^p$ to predict the other users' private attribute as follows:

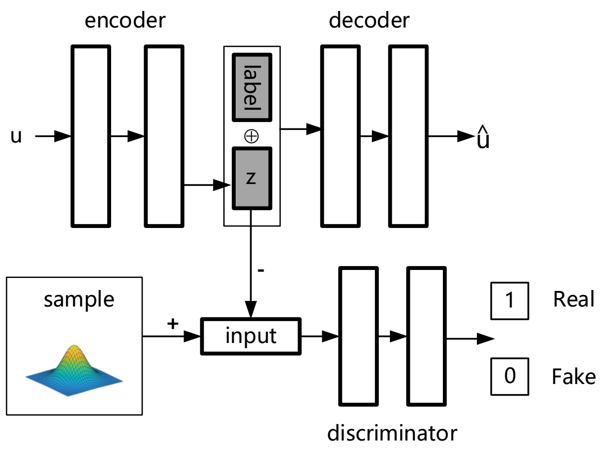
$$\hat{y}_i^p = \arg\max_{y_i^p} P_\Lambda(Y_i^p = y_i^p | \mathbf{Z}, \mathcal{Y}_k^p), \tag{1}$$

where Λ denotes the attacking classifier, Y_i^p is the random variable of the i-th user's sensitive attribute, y_i^p and $\hat{y}_i^{(p)}$ denote the possible value and the predicting result of the i-th user's sensitive attribute, respectively. For generality, we do not constrain the types and parameters of the classifier Λ. Note that the differential privacy is to protect the information that if a record is in the dataset, therefore the differential privacy could not defend the attack method we defined.

Privacy and Utility. In this work, we define the sensitive attributes of users as privacy and the topology of the graph and users' non-sensitive attributes as utility. That means the learned low dimensional node representations $\mathbf{Z}$ should satisfy two



(a) Classcial AAE



(b) Supervised AAE

Fig. 2: Adversarial Autoencoder

properties: (1) the network structure and node utility attributes are preserved; and (2) the sensitive information is concealed, which means the privacy preserving graph embedding is robust to all kinds of inference attacks regardless the attacking classifier's types and parameters. Therefore, if we use the learned node representation matrix $\mathbf{Z}$ to analyze graph, the accuracy of link prediction and utility attribute classification should be preserved, while the inference accuracy of private attribute should be reduced.

B. Graph Autoencoder

Graph autoencoder (GAE) [33] embeds a graph $\mathbf{G}(\mathbf{V}, \mathbf{E}, \mathbf{X})$ to a low-dimensional space. The encoder of GAE is a graph convolutional network (GCN) [34], which updates hidden layer of a graph by

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{L}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{2}$$

where $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized graph Laplacian, $\mathbf{H}^{(l)}$ is the output of l-th graph convolutional layer, $\mathbf{W}^{(l)}$ is the weight matrix of l-th layer that is to be learned during training, and σ represents the activation function, such as ReLU. Additionally, $\widetilde{\mathbf{A}}$ denotes the adjacency matrix $\mathbf{A}$ with diagonal elements set to 1, i.e., every node contains a self-loop, and $\mathbf{D}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$ denotes the degree of node i. Usually, we stack two graph convolutional layers as the encoder, with the propagation rule of the encoder expressed as follows:

$$GCN(\mathbf{A}, \mathbf{X}) = \mathbf{L}\sigma(\mathbf{L}\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)}. \tag{3}$$

The purpose of GAE is to embed the structural information of a graph to a low-dimensional space. Therefore, the decoder of GAE reconstructs adjacency matrix via the inner product of embedding matrix $\mathbf{Z}$. The reconstructed adjacency matrix $\hat{\mathbf{A}}$ and the embedding matrix $\mathbf{Z}$ can be represented as follows:

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T), \quad \text{with} \quad \mathbf{Z} = GCN(\mathbf{A}, \mathbf{X}). \tag{4}$$

To preserve the structural information of a graph, GAE optimizes the link prediction by minimizing the cross-entropy loss:

$$L_{link} = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbf{A}_{ij}\log(\hat{\mathbf{A}}_{ij}), \tag{5}$$

where $\mathbf{A}_{ij}$ and $\hat{\mathbf{A}}_{ij}$ are the elements of $\mathbf{A}$ and $\hat{\mathbf{A}}$, respectively.

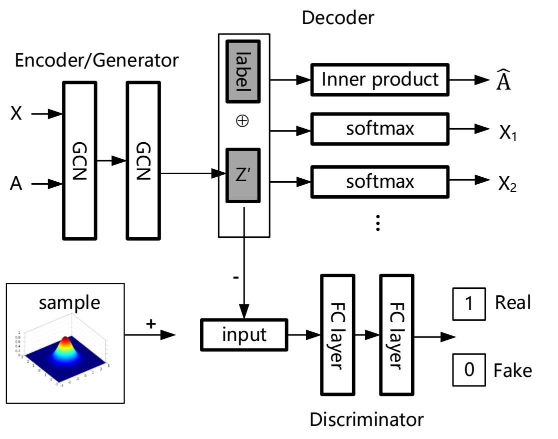
C. Adversarial Autoencoder

Adversarial Autoencoder (AAE) [35] is a probabilistic autoencoder which uses the generative adversarial network (GAN) to perform variational inference by enforcing the posterior distribution of the hidden code to a specified prior distribution. The classical AAE architecture is shown in Fig. 2(a). The top row is an autoencoder which encodes the image $\mathbf{u}$ to hidden representation $\mathbf{z}$ and decodes $\mathbf{z}$ to a reconstructed image $\hat{\mathbf{u}}$. The bottom row is a discriminator which classifies the input is the code $\mathbf{z}$ or the noise sampled from the specified prior distribution. As we train AAE, we update the encoder and decoder to minimize the reconstruction error firstly. Then we update the discriminator to distinguish if the input is from true sample or the code generator, and we update the encoder to fool the discriminator.

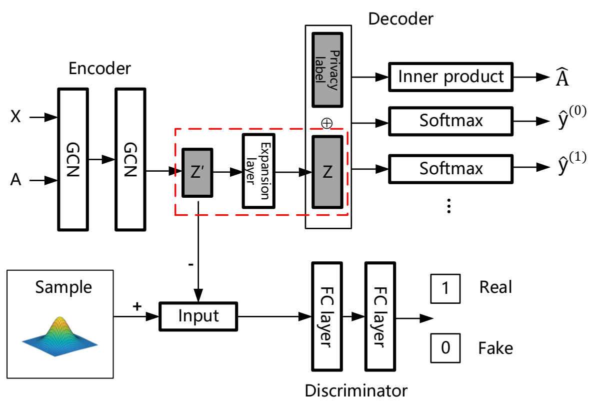
In [35], the above classical AAE has been further extended to supervised AAE that augments the decoder with the one-hot encoding of the label as shown in Fig. 2(b). As the label and $\mathbf{z}$ are utilized jointly for reconstruction, $\mathbf{z}$ should contain the label-invariant information of input $\mathbf{u}$ for reconstruction. In other words, in supervised AAE the label information is disentangled from the latent representation $\mathbf{z}$. In this paper, such the ability of disentangled of supervised AAE is exploited for privacy protection.

III. Proposed Model

We first propose two privacy preserving embedding models corresponding to mechanisms Privacy-Disentangling and Privacy-Pruning, respectively. Then we integrate the two models into one end-to-end pipeline to achieve the best performance. In the sequel, we will introduce these models one by one.



(a) Original APDGE



(b) Final APDGE

Fig. 3: Adversarial Privacy-Disentangled Graph Embedding

A. Adversarial Privacy-Disentangled Graph Embedding

As discussed in section II-C, supervised AAE can be readily adapted for privacy protection. The main issue is that the input of supervised AAE is regular grid data, such as images, while we are interested in graph structured data which are irregular. We therefore extend GAE and supervised AAE and propose Privacy-Disentangling mechanise for privacy preserving.

The architecture of original "Adversarial Privacy-Disentangled Graph Embedding" (APDGE) is shown in Fig. 3(a), which is similar to supervised AAE and consists of encoder, decoder and discriminator. To support graph structured data, the encoder of AAE is replaced by GCN, that is, latent code $\mathbf{Z}'$ is extracted by a GCN: $\mathbf{Z}' = GCN(\mathbf{A}, \mathbf{X})$ as shown in Eq. 3. Note that, the encoder is the stack of two graph convolutional layers, so both first and second order friendships are considered in this model. To disentangle privacy from latent code $\mathbf{Z}'$, we incorporate the privacy label (e.g., gender) to the decoder. On the one hand, the distribution of $\mathbf{Z}'$ is regularized to the specified prior distribution. So the information of $\mathbf{Z}'$ is enforced to be as little as possible. On the other hand, $\mathbf{Z}'$ should contain the sufficient information to reconstruct the topology and utility attributes of the graph. In this way, the private information, which has been provided in the privacy label, is removed from $\mathbf{Z}'$. In other words, the model learns a compressed $\mathbf{Z}'$ that contains as little information of the privacy label as possible, while being sufficient for the decoder to reconstruct the utilities of graph. Therefore, the learned graph embedding $\mathbf{Z}'$ retains the utilities of the graph while preserves the privacy information.

We hope the dimension of $\mathbf{Z}'$ is low such that the privacy could be extruded as much as possible. However, if we directly concatenate $\mathbf{Z}'$ and the one-hot privacy label encoding as the input of decoder, the performance of decoder will be suffering because the dimension of the concatenated vector is too low. Besides, as the dimension of $\mathbf{Z}'$ could be changed to trade off the privacy and utility (which will be analyzed in section IV-C), we hope the dimension of the embeddings does not depend on the dimension of $\mathbf{Z}'$. From extensive experiments, we find a simple trick that successfully address these issues. That is, before feeding the low dimensional representation $\mathbf{Z}'$ to the decoder, we map it to a higher dimensional $\mathbf{Z}$ via a expansion layer; then we concatenate privacy label with $\mathbf{Z}$ to get $\mathbf{Z}^+$, which is fed to the decoder. Specifically, we use a fully connected layer to implement the expansion layer. Therefore, we modify the original APDGE to the final APDGE as shown in Fig. 3(b).

Since the learned embedding $\mathbf{Z}'$ should retain graph topology and utility attributes information, we augment the decoder to have multiple modules for adjacency matrix reconstruction and attribute classifiers. The adjacency reconstruction module calculates the inner production of $\mathbf{Z}^+$ to reconstruct adjacency matrix with the loss function for link prediction as Eq.5. For utility attribute classification, we decode $\mathbf{Z}^+$ using the softmax function, followed by the cross-entropy loss. Precisely, we predict the i-th user's c-th utility attribute by $\hat{\mathbf{y}}_i^{(c)} = \text{softmax}_c(\mathbf{z}_i^+)$, which is evaluated by the cross-entropy loss as:

$$L_{y_c} = -\frac{1}{|\mathbf{V}^{(c)}|} \sum_{i \in \mathbf{V}^{(c)}} \sum_{j=1}^{M_c} \mathbf{y}_{ij}^{(c)} \log \hat{\mathbf{y}}_{ij}^{(c)}, \tag{6}$$

where $\mathbf{V}^{(c)}$ is the set of users on which the c-th utility labels are available for training, $\mathbf{y}_{ij}^{(c)}$ and $\hat{\mathbf{y}}_{ij}^{(c)}$ are the j-th dimension value of the i-th user's one-hot attribute label vectors $\mathbf{y}_i^{(c)}$ and the predicting result $\hat{\mathbf{y}}_i^{(c)}$ respectively , and M_c is the dimension of the c-th utility attribute. The total loss of APDGE is the combination of the link prediction loss and utility attribute classification loss:

$$L_{recon} = L_{link} + \sum_{c \in \mathbf{C}} L_{y_c}, \tag{7}$$

where $\mathbf{C}$ is the set of utility attributes.

The discriminator D of this model is the same as the discriminator of AAE. We use two fully connected layers to classify the input to be real or fake depending on if the input is a sample $\mathbf{s}$ from a unit Gaussian distribution or $\mathbf{z_i'}$ from the encoder of APDGE. We optimize the discriminator to minimize the following loss:

$$L_{dc} = -\log(D(\mathbf{s})) - \log(1 - D(\mathbf{z_i'})). \tag{8}$$

Overall, the training of APDGE contains three stages: (a) Update the encoder and decoder to minimize the loss Eq. 7, (b) Update the discriminator to distinguish the real samples from the fake samples by minimizing the loss Eq. 8, and (c) Update the encoder to fool the discriminator by maximizing the loss Eq. 8. Upon finishing the training, we get the privacy-preserved node representation (latent code) matrix $\mathbf{Z}$.

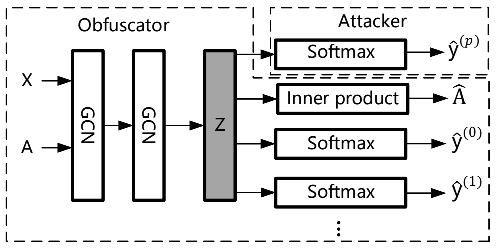


Fig. 4: Adversarial Privacy-Purged Graph Embedding

B. Adversarial Privacy-Purged Graph Embedding

The adversarial training mechanism discussed above disentangles privacy latent factors from $\mathbf{Z}$ by disclosing the privacy labels directly as input to the decoder. We can also achieve a similar effect by disclosing the privacy labels as the output of the decoder. We call the latter mechanism Privacy-Purging and proposed "Adversarial Privacy-Purged Graph Embedding" (APPGE) as shown in Fig. 4. APPGE consists of two networks (1) An Attacker whose purpose is to extract privacy information from embedding matrix $\mathbf{Z}$; and (2) An Obfuscator which attempts to embed utility information and prevent the inference attacks launched by the attacker.

Essentially, Attacker is a softmax classifier that aims to predict privacy labels $\mathbf{Y}^{(p)}$. We optimize the Attacker to minimize the following loss:

$$L_{att} = -\frac{1}{|\mathbf{V}^{(p)}|} \sum_{i \in \mathbf{V}^{(p)}} \sum_{j=1}^{M_p} \mathbf{y}_{ij}^{(p)} \log \hat{\mathbf{y}}_{ij}^{(p)}, \quad (9)$$

where $\mathbf{V}^{(p)}$ is the set of users on which the private attribute labels are available for training, $\mathbf{y}_{ij}^{(p)}$ and $\hat{\mathbf{y}}_{ij}^{(p)}$ are the j-th dimension value of i-th user's privacy one-hot label $\mathbf{y}_i^{(p)}$ and the predicting result of privacy $\hat{\mathbf{y}}_i^{(p)}$ respectively, and M_p is the dimension of private attribute. The architecture of Obfuscator is similar to GAE. The Obfuscator learns the latent code via GCN and decodes latent code to reconstruct network structure and utility attributes. In addition, it also tries to purge the latent code such that the attacker can't predict private attribute accurately. Therefore, we optimize the Obfuscator to minimize the reconstruction error and at the same time maximize the prediction error of the attacker. So the loss of the Obfuscator is as follows:

$$L_{obf} = L_{recon} - \lambda L_{att}, \quad (10)$$

where λ is a hyperparameter that balances between utility and privacy. When we train the model, we update the Attacker and Obfuscator alternately until both modules' loss functions plateau.

C. Adversarial Privacy Graph Embedding

The Privacy-Disentangling and Privacy-Purging mechanisms we discussed above preserve the privacy information at the different stages of the training: the former is at the input of the decoder, while the latter at the output of the decoder, and they may work complementary to each other. Therefore, we integrate both mechanisms into one framework "Adversarial Privacy Graph Embedding" (APGE) to jointly protect users' privacy, with the model shown in Fig. 5. As we can see, the architecture of APGE is very similar to APPGE that consists of an Obfuscator and an Attacker, while we integrate APDGE as the architecture of the Obfuscator. Note that the Attacker launches an inference attack on the latent code matrix $\mathbf{Z}$ rather than the latent code $\mathbf{Z}^+$, which is the concatenation of $\mathbf{Z}$ and privacy label encoding.

Algorithm 1 Adversarial Privacy Graph Embedding

```
Input: G(V, E, X):A graph with links and features; $T$: the number of iterations; $K_{dis}$: the number of steps for iterating Discriminator; $K_{att}$: the number of steps for iterating Attacker; $d$: the dimension of the latent variable.
Output: $\mathbf{Z} \in \mathbb{R}^{n \times d}$;
1:  for iteration = 1 to $T$ do
2:      for $k_1 = 1$ to $K_{att}$ do
3:          Update the Attacker by minimizing the loss of Eq.9
4:      end for
5:      Update the Encoder and Decoder to minimize the loss of Eq.10
6:      for $k_2 = 1$ to $K_{dis}$ do
7:          Sample from Gaussian distribution and $\mathbf{Z}'$
8:          Update the Discriminator to distinguish the real samples from the fake samples by minimizing the loss of Eq.8
9:      end for
10:     Update the Encoder to fool the discriminator by maximizing the loss Eq.8
11: end for
12: return Z
```

We present the training process of APGE in the Algorithm 1. Given a garph $\mathbf{G}$, the Attacker and Obfuscator of APGE can be optimized by alternating gradient descent in two stages. Firstly, for the Attacker, we minimize the prediction error of the private attribute by Eq.9 as shown in Line 2-4. Secondly, for the Obfuscator, whose training process is similar to ADPGE, we update the model as shown in Line 5-10. After finishing the training, we get the privacy-preserved graph embedding matrix $\mathbf{Z}$.

IV. Experiments

In this section, we empirically demonstrate the effectiveness of our method,APGE, and compare it with the start-of-the-art privacy preserving method. We also evaluate the model's robustness to different attack models and the impacts of the hyperparameters and expansion layer in our method.

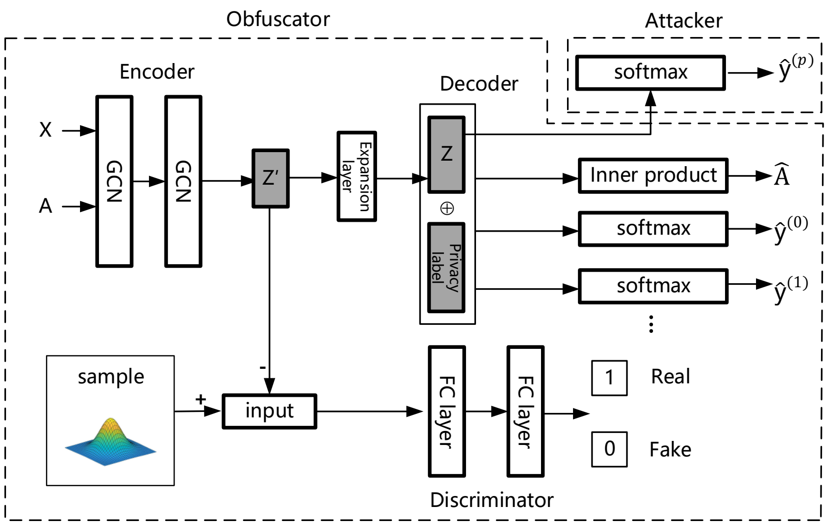


Fig. 5: Adversarial Privacy Graph Embedding

TABLE I: Utility and Privacy Evaluation on Yale

Method	link		utility attributes(status)		private attribute(class year)			
					MLP		SVM	
	ACC	Macro F1	ACC	Macro F1	ACC	Macro F1	ACC	Macro F1
GAE	0.856 ± 0.003	0.856 ± 0.003	0.923 ± 0.002	0.927 ± 0.002	0.931± 0.002	0.930 ± 0.002	0.940 ± 0.001	0.941 ±0.001
GAE_RM	0.854 ± 0.002	0.854 ± 0.002	0.933 ± 0.003	0.930± 0.003	0.907± 0.002	0.906± 0.002	0.918± 0.001	0.918 ± 0.001
CDSPIA	0.824 ± 0.002	0.824 ± 0.002	0.920 ± 0.001	0.916 ± 0.001	0.896± 0.003	0.896± 0.003	0.894± 0.001	0.893 ± 0.001
APDGE	0.840 ± 0.001	0.840 ± 0.001	0.884 ± 0.003	0.877± 0.003	0.824± 0.009	0.825± 0.009	0.798± 0.008	0.801 ± 0.007
APPGE	0.856 ± 0.002	0.856 ± 0.002	0.929 ± 0.005	0.923 ± 0.005	0.890± 0.011	0.890± 0.011	0.891± 0.009	0.890 ± 0.009
APGE	0.820 ± 0.002	0.820 ± 0.002	0.813 ± 0.010	0.780 ± 0.009	**0.568** ± **0.014**	**0.579** ± **0.016**	**0.514** ± **0.013**	**0.518** ± **0.016**

TABLE II: Utility and Privacy Evaluation on Rochester

Method	link		utility attributes(class year)		private attribute(gender)			
					MLP		SVM	
	ACC	Macro F1	ACC	Macro F1	ACC	Macro F1	ACC	Macro F1
GAE	0.862 ± 0.001	0.862 ± 0.001	0.953 ± 0.002	0.947 ± 0.002	0.781± 0.003	0.781 ± 0.003	0.769 ± 0.002	0.768 ± 0.002
GAE_RM	0.857 ± 0.001	0.857 ± 0.002	0.948± 0.002	0.941± 0.002	0.732± 0.002	0.731± 0.002	0.724± 0.001	0.722 ± 0.001
CDSPIA	0.848 ± 0.002	0.848 ± 0.002	0.862 ± 0.002	0.851 ± 0.002	0.666± 0.002	0.666± 0.002	0.655± 0.001	0.654 ± 0.001
APDGE	0.849 ± 0.002	0.849 ± 0.002	0.924 ±0.003	0.915 ± 0.003	0.654± 0.003	0.651± 0.003	0.655± 0.004	0.652 ± 0.004
APPGE	0.857 ± 0.002	0.857 ± 0.002	0.895 ± 0.003	0.889 ± 0.003	0.717± 0.004	0.717± 0.004	0.717± 0.003	0.717 ± 0.003
APGE	0.848 ± 0.002	0.848 ± 0.002	0.923 ± 0.005	0.914 ± 0.005	**0.636** ± **0.004**	**0.634** ± **0.004**	**0.649** ± **0.007**	**0.645** ± **0.007**

A. Dataset and Baseline Methods

We conduct experiments on two real-world graph datasets: **Yale** and **Rochester** [1] which collect all the facebook friendships of Yale University and Rochester University in 2005 as well as a number of user attributes including student/faculty status (status for short), gender, major, second major, class year, dorm/house, and high school. The two networks contain 8578 nodes, 405450 edges and 4563 nodes, 167653 edges, respectively. For Yale, class year (6 categories) is set as the sensitive attribute and status is set as the utility attribute; while for Rochester, we regard gender (2 categories) as a sensitive attribute and class year as the utility attribute. Indeed, if the labels of sensitive attribute are provided, our methods can also deal with the cases where the other attributes are private, *e.g.,* sexual orientation and political opinion.

We evaluate the performance of proposed models against the following approaches: (1) **GAE** [33] learns network representation by autoencoder where encoder is graph convolutional network and decoder consists of inner product to reconstruct adjacency matrix and softmax function to predict utility attributes. (2) **GAE_RM** [33] is a framework similar to that of GAE. But the sensitive attributes are removed from the input attribute matrix $\mathbf{X}$ directly. (3) **CDSPIA** [28] is a state-of-the-art approach to defending inference attacks on graph by deleting or perturbing users' attributes and linkages which are closely related to privacy. We use CDSPIA to sanitize the graph dataset and then embed the graph data processed via GAE.

All the methods embed the graphs to a 64-dimensional space. Particularly, in both APDGE and APGE, we first compress each node's information to a 16-dimensional vector $\mathbf{z}'_i$ for Yale and a 8-dimensional vector for Rochester. For APPGE and APGE, the λ in Eq. 10 is set as 1 and 10 for Yale and Rochester, respectively

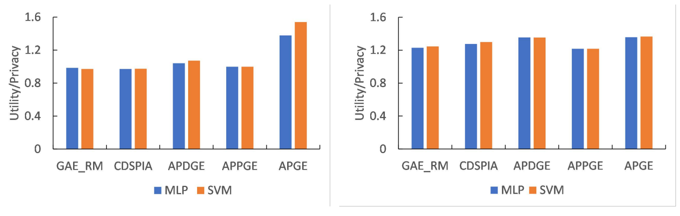
[1]https://escience.rpi.edu/data/DA/fb100/

B. Utility and Privacy Evaluation

In this part of experiments, we verify the effectiveness of the three models we proposed on utility reservation and privacy protection and compare them with the baselines. We qualify the utility in term of prediction accuracy of link and utility attribute from the embeddings. For link prediction, we compare models based on the classifier's ability to correctly classify edges and non-edges. Here the classifier is multi-layer perceptron (MLP) [36] trained on embeddings. For utility attribute prediction, an MLP is utilized to classify the utility attributes based embeddings. Meanwhile, we qualify the privacy in term of prediction accuracy of sensitive attribute. Both MLP and support vector machine (SVM) [37] are employed as the adversaries to predict the sensitive attribute. For each models, we repeat the experiment 10 times and report the average results and standard deviation in Table I and Table II.

Comparing the performance of GAE and GAE_RM, we can see that even if the private attribute is removed from the input of embedding model, the attacker is still able to infer the user's privacy from embeddings by the inherent relation between the private attribute and the input of embedding model. It can be seen from Table I that APDGE obtains an average Macro-F1 of 0.813 on private attribute, outperforming the previous approach CDSPIA. Meanwhile, we observe that APDGE and CDSPIA have similar performances on preserving utilities. This indicates that the end to end training of graph embedding and privacy protection provides a stronger system than the decoupled method that processes the two steps separately for preserving privacy in graph embedding.

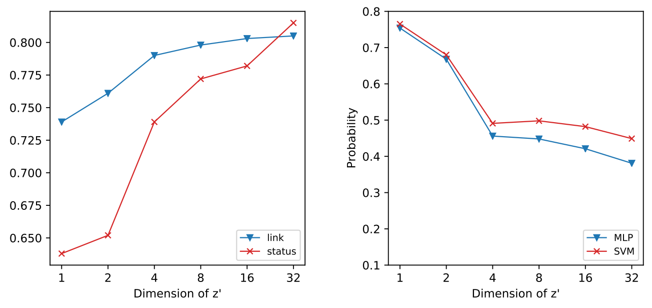
From the results on both the Yale and Rochester datasets, we find that APGE demonstrates the strongest capability of protecting user's privacy while retaining the utilities. On Yale, APGE achieves an average Macro-F1 of 0.549 on private attribute, outperforming APDGE by a significant margin of 30%. At the same time, the accuracies of link and utility attribute are preserved largely. This demonstrates that the integration of disentangling and purging mechanisms brings an additional and significant gain in performance. We observe that the preserving privacy performance of APGE is more significant on Yale than on Rochester. We note that this is because the privacy is more closely correlated with the utility in the Yale dataset than Rochester. That is, user's friendship and status are strongly related to her class year. At the same time, user's linkage and class year are almost irrelevant to her gender.



(a) Yale (b) Rochester

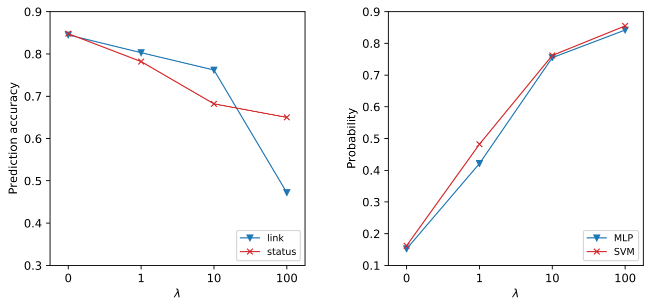
Fig. 6: Evaluation on the trade off between utility and privacy

Moreover, to illustrate the trade off between utility and privacy, Fig. 6 reports the ratio of utility (that is the average Macro-F1 of link and utility attributes) to privacy (*i.e.*, the Macro-F1 of private attribute). As we can see, APGE achieves the best performance when both utility preservation and privacy protection are taken into account.



(a) Impact on utility (b) Impact on privacy

Fig. 7: Impact of the hidden code's dimension

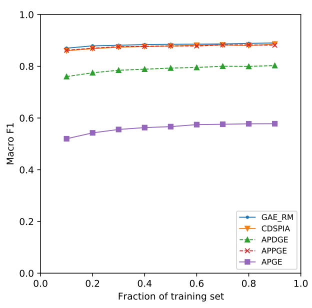


(a) Impact on utility (b) Impact on privacy

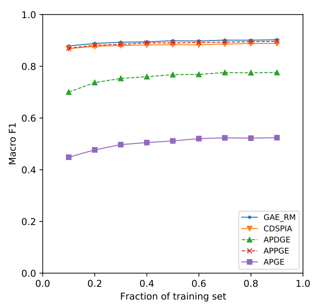
Fig. 8: Impact of the λ

C. Impact of hyperparameters

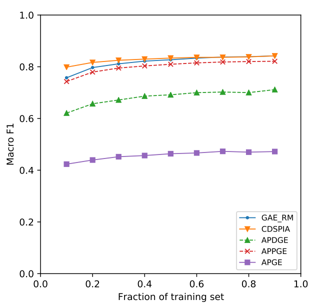
The dimension of hidden code $\mathbf{z}'_i$ and the hyperparameter λ have significant impacts to the performance of APGE. Fig. 7 shows the performance of APGE on privacy pretection and utility preserving on Yale when the hidden code dimension increases. Here, we define "1 - prediction accuracy of private attribute" as the probability of successfully preventing inference attack. As we can see, when the dimension of hidden code increases, link and utility attributes can be recovered more accurately. On the other hand, the higher dimension of hidden code is, the more private information is retained, leading to a lower probability of preventing inference attack. Similarly, Fig. 8 shows the performance of APGE on Yale when λ increases from 0 to 100. As expected, when λ increases, we observe a monotonic decrease of utility prediction accuracy and a monotonic increase of the probability of resisting inference attack. These results demonstrate that we could adjust the dimension of $\mathbf{z}'_i$ and λ to achieve a desired utility-privacy tradeoff.



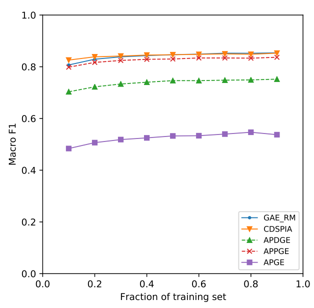
(a) MLP



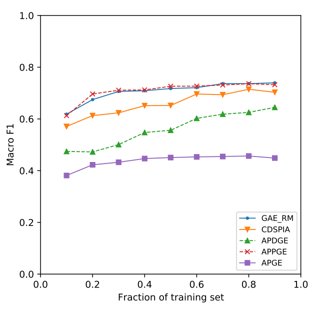
(b) SVM



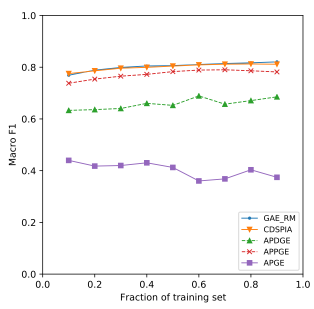
(c) KNN



(d) Random Forest

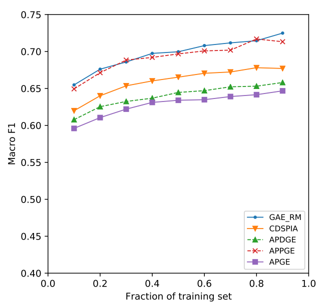


(e) Adaboost

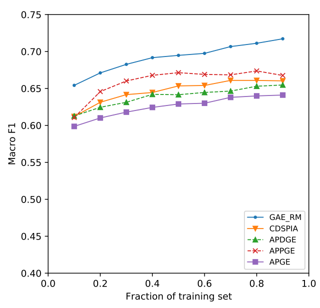


(f) Gradient boosting

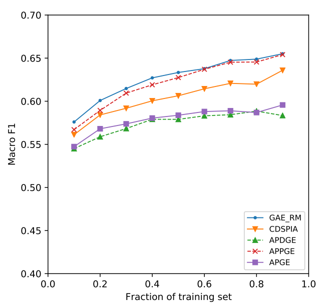
Fig. 9: Inference attack on Yale with different attack models



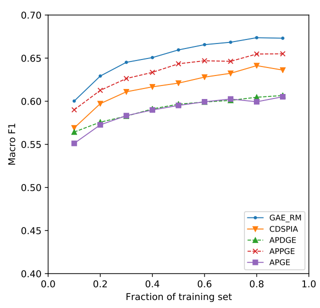
(a) MLP



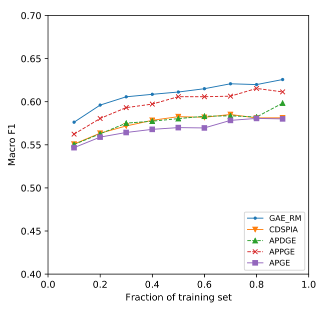
(b) SVM



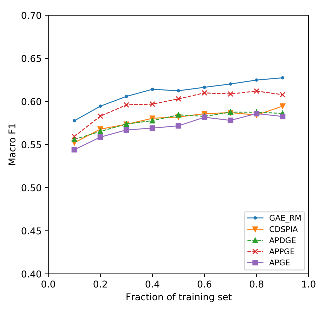
(c) KNN



(d) Random Forest



(e) Adaboost



(f) Gradient boosting

Fig. 10: Inference attack on Rochester with different attack models

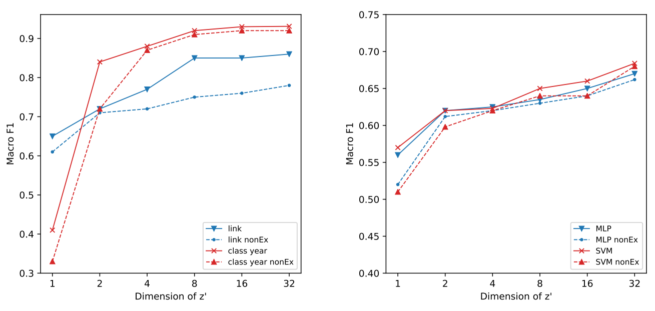
D. Robustness against attack models

For generality, we assume the adversary could launch inference attack with any classifiers. As the precious works [22] [28], we select different classifiers, including MLP, SVM, k-nearest neighbors (KNN) [38], Random forest [39], Adaboost [40], Gradient boost [40], as attack models to evaluate the robustness of our proposed methods. We train the attack models using 10%-90% users' embeddings and private attribute labels to infer the other users' sensitive attributes and show the results in Fig. 9 and 10. As we can see, for all the six attack models, APGE outperforms the other methods significantly on reducing the Macro-F1 of sensitive attribute prediction for both datasets. That's because we attempt to conceal the private information from embeddings with APGE, if the information is removed from embeddings enough, any classifier could not infer the sensitive attribute from the embeddings.

There's another observation, when we only remove the sensitive attribute labels directly from the input attribute matrix (i.e., GAE_RM), the adversary could predict the other users' privacy with the Macro F1 more than 0.8 on Yale dataset, even if the adversary just obtain 10% users' sensitive attribute,. That means the users' privacy faces serious threat, if the data owner publishes graph embedding without considering privacy preserving.

E. Effect of the Expansion Layer

As introduced in the section III-A, we expand $\mathbf{Z}'$ to $\mathbf{Z}$ via an expansion layer in APDGE to improve the embedding performance. Since the structure of the Obfuscator of APGE is the same as the ADPGE, the expansion layer is also used in APGE as shown in Fig. 5. In this section, we remove the expansion layer of APGE and directly input $\mathbf{Z}'$ to Attacker and Decoder to evaluate the impact of the expansion layer.



(a) Impact on utility (b) Impact on privacy

Fig. 11: Impact of the Expansion layer

After finishing training the non-expansion layer model (non-Ex model for short), we obtain the $\mathbf{Z}'$ as the embeddings. We qualify the utility and privacy of the embeddings learned by APGE and non-Ex model on Rochester, and show the results in Fig. 11. We can observe that the link prediction performance of APGE is significantly larger than the non-Ex model. The dimension of $\mathbf{Z}'$ must be small enough to disentangle the private information. Therefore, When we learn the embeddings via the non-Ex model, the dimension of embeddings (i.e., the dimension of $\mathbf{Z}'$) is too small to sufficiently contain the utility information. On the other hand, as a lot of private information is concealed in both APGE and the non-Ex model, even if the embeddings' dimension is small, the embeddings could contain almost all the remaining private information. Therefore, if we remove the expansion layer, the inference accuracy of sensitive attributes does not reduce remarkably, as shown in Fig. 11(b).

V. Related Work

In this section, we briefly survey the existing graph embedding methods and introduce the state-of-the-art privacy preservation research on graph.

A. Graph Embedding Method

DeepWalk [41] is the early work to learn graph representation by generating the node context via random walk and mapping nodes to vectors based on skip-gram [42]. Node2vec [43] exploits a new method to generate node context with considering both local and global structure information and then embeds graph using DeepWalk. The above methods just utilize topology information, and some works have been proposed to improve embedding performance taking into account both the structure and attribute information. TirDNR [44] combines DeepWalk and Doc2Vec [45] to learn the inner-node relationship, node-word correspondence, and word-label correlation. UPP-SNE [46] uses the attribute matrix of social network as input and the node context as label to supervised training model. Different from skip-gram architecture, GAE [33] is a graph convolutional network via an approximation of spectral graph convolutions for learning the graph representation and outperforms skip-gram architecture. Recently, ARGA [47] is proposed to add regularization to GAE via an adversarial framework. Unfortunately, these above graph embedding methods do not consider privacy preservation; that is, adversaries can infer the user's sensitive attribute from the embedding results.

B. Privacy Preservation on Graph

The only existing works to preserve privacy on graph embedding is to achieve the link information differential privacy. That is, even if we add or delete an edge from the original graph, there is no significant statistic difference between the embeddings of the original graph and the changed graph . Xu *et al.* [30] adopts the objective perturbation mechanism [48] on the loss function of the embedding model to achieve differential privacy. Zhang and Ni [31] propose a Lipschitz condition [49] on the objective function of the embedding model and a gradient clipping strategy to ensure the differential privacy on link information. However, both of the above approaches do not consider to defend inference attack on users' sensitive attributes.

There has been a few attempts to prevent inference attack with directly sanitizing graph data. Jia and Gong [50] build a set of noise via adversarial machine learning and randomly select the noise to mislead the inference attacker. However, they only manipulate the users' attributes and ignore the linkages, and thus their method cannot resist the inference attack utilizing topology information well. Cai *et al.* [28] first propose an inference attack method with the mixture of non-sensitive attribute and link relationship and then design a privacy-preserving method by removing or perturbing users' attributes and links that are closely related to private attribute. He *et al.* [29] obtain the data-sanitization strategy by solving an optimization problem, which can balance the trade off between utility and privacy. However, these prior works do not integrate the processes of privacy protection and graph embedding, and thus is not suitable for privacy-preserving embedding on graphs.

VI. Conclusion

This paper proposes a novel graph embedding framework that produces node representations with users' privacy preserved. To the best of our knowledge, this is the first graph embedding method that integrates privacy protection into GCN. We introduce two different mechanisms for privacy protection in graph embedding: latent factor disentangling and adversarial training. The resulting algorithm APGE which integrates the two mechanisms into one end-to-end pipeline demonstrates superior performance as compared to the state-of-the-arts in privacy protection while retaining similar accuracy in link prediction and users' utility classification.

References

[1] "Number of internet of things (iot) connected devices worldwide in 2018, 2025 and 2030," https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[3] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart internet of things systems: a consideration from a privacy perspective," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 55–61, 2018.

[4] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial iots," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[5] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[6] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan, "Group formation in large social networks: membership, growth, and evolution," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 44–54.

[7] M. Siddula, Y. Li, X. Cheng, Z. Tian, and Z. Cai, "Anonymization in online social networks based on enhanced equi-cardinal clustering," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 4, pp. 809–820, 2019.

[8] J. Clement, "Number of social network users worldwide from 2010 to 2023," https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/.

[9] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[10] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 833–852, 2019.

[11] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[12] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[13] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[14] I. Herman, G. Melançon, and M. S. Marshall, "Graph visualization and navigation in information visualization: A survey," *IEEE Transactions on visualization and computer graphics*, vol. 6, no. 1, pp. 24–43, 2000.

[15] F. Gavril, "Some np-complete problems on graphs," Computer Science Department, Technion, Tech. Rep., 2011.

[16] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry, "Challenges in parallel graph processing," *Parallel Processing Letters*, vol. 17, no. 01, pp. 5–20, 2007.

[17] M. Ellers, M. Cochez, T. Schumacher, M. Strohmaier, and F. Lemmerich, "Privacy attacks on network embeddings," *arXiv preprint arXiv:1912.10979*, 2019.

[18] Y. Zhang and M. Pennacchiotti, "Predicting purchase behaviors from social media," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1521–1532.

[19] P. Rozenshtein, A. Anagnostopoulos, A. Gionis, and N. Tatti, "Event detection in activity networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1176–1185.

[20] J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Inferring private information using social network data," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 1145–1146.

[21] R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Preventing private information inference attacks on social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1849–1862, 2012.

[22] N. Z. Gong and B. Liu, "Attribute inference attacks in online social networks," *ACM Transactions on Privacy and Security (TOPS)*, vol. 21, no. 1, p. 3, 2018.

[23] C. Jernigan and B. F. Mistree, "Gaydar: Facebook friendships expose sexual orientation," *First Monday*, vol. 14, no. 10, 2009.

[24] D. Garcia, "Leaking privacy and shadow profiles in online social networks," *Science advances*, vol. 3, no. 8, p. e1701172, 2017.

[25] N. Steinfeld, "Trading with privacy: the price of personal information," *Online Information Review*, 2015.

[26] A. Acquisti, L. K. John, and G. Loewenstein, "What is privacy worth?" *The Journal of Legal Studies*, vol. 42, no. 2, pp. 249–274, 2013.

[27] R. Meusel, S. Vigna, O. Lehmberg, and C. Bizer, "Graph structure in the web—revisited: a trick of the heavy tail," in *Proceedings of the 23rd international conference on World Wide Web*, 2014, pp. 427–432.

[28] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[29] Z. He, Z. Cai, and J. Yu, "Latent-data privacy preserving with customized data utility for social network data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, 2018.

[30] D. Xu, S. Yuan, X. Wu, and H. Phan, "Dpne: differentially private network embedding," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 235–246.

[31] S. Zhang and W. Ni, "Graph embedding matrix sharing with differential privacy," *IEEE Access*, vol. 7, pp. 89 390–89 399, 2019.

[32] K. Steinmetz and J. Gerber, "" it doesn't have to be this way": Hacker perspectives on privacy," *Social Justice*, vol. 41, no. 3 (137, pp. 29–51, 2015.

[33] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *NIPS Workshop on Bayesian Deep Learning*, 2016.

[34] T. N. kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[35] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *ICLR workshop*, 2016.

[36] S. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, 1992.

[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[38] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[39] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[40] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[41] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *ICLR Workshop*, 2013.

[43] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[44] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[45] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

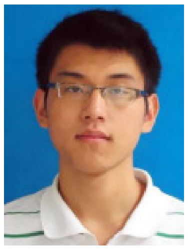
[46] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "User profile preserving social network embedding." in *IJCAI*, 2017, pp. 3378–3384.

[47] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *IJCAI*, pp. 2609–2615, 2018.

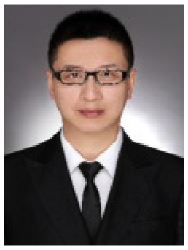
[48] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[49] M. Jha and S. Raskhodnikova, "Testing and reconstruction of lipschitz functions with applications to data privacy," *SIAM Journal on Computing*, vol. 42, no. 2, pp. 700–731, 2013.

[50] J. Jia and N. Z. Gong, "Attriguard: A practical defense against attribute inference attacks via adversarial machine learning," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 513–529.



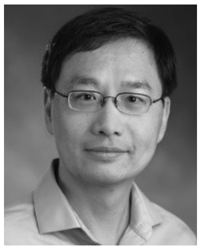
Kaiyang Li received the M.S. degree from the School of Energy Science and Engineering University of Electronic Science and Technology of China, Chengdu, China, in 2016. He is currently a Ph.D. student in the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include privacy and machine learning.



Guangchun Luo received the Ph.D. degree in computer science from University of Electronic Science and Technology of China, Chengdu, China, in 2004. He is currently a professor of computer science at the University of Electronic Science and Technology of China. He has published over sixty journal and conference papers in his fields. His research interests include computer networks and big data.

Yang Ye received the M.S. degree in computer science from Huazhong University of Science and Technology, Wuhan, Hubei, China, in 2014. He is currently working toward the Ph.D. degree in the Department of Computer Science, Georgia State University, Atlanta, GA, USA. His research interests include deep learning and data minig.

Wei Li currently is an Assistant Professor in the Department of Computer Science at Georgia State University. Dr. Li received her Ph.D. degree in Computer Science from The George Washington University in 2016 and her M.S. degree in Computer Science from Beijing University of Posts and Telecommunications in 2011. She won the Best Paper Awards in ACM MobiCom CRAB 2013 and WASA 2011, respectively. Her current research mainly spans the areas of security and privacy for the Internet of Things and Cyber-Physical Systems, secure and privacy-aware computing, Big Data, Blockchain, Game Theory, and algorithm design and analysis. She is a member of IEEE and ACM.



Shihao Ji received the PhD degree in electrical and computer engineering from Duke University, in 2006. After that, he was an research associate with Duke for about 1.5 years. He is an associate professor with the Computer Science Department, Georgia State University. His principal research interests lie in the area of machine learning and deep learning with an emphasis on high-performance computing. He is interested in developing efficient algorithms that can learn from a variety of data sources (e.g., image, audio, and text) on a large scale and automate decision-making processes in dynamic environments. Prior to joining GSU, he spent about 10 years in industry research labs.

Zhipeng Cai received the B.S. degree from the Beijing Institute of Technology, Beijing, China, and the M.S. and Ph.D. degrees in computing science from the University of Alberta, Edmonton, AB, Canada. He is currently an Associate Professor with the Department of Computer Science, Georgia State University. His research interests include networking, privacy, and big data. He has received an NSF CAREER Award. He is an editor/guest editor of the IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Vehicular Technology, IEEE Networking Letters, Algorithmica, Theoretical Computer Science, the Journal of Combinatorial Optimization, and the IEEE/ACM Transactions on Computational Biology and Bioinformatics. He is a senior member of the IEEE.