This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944

A Robust and Generalized Framework for Adversarial Graph Embedding

Jianxin Li, Xingcheng Fu, Shijie Zhu, Hao Peng, Senzhang Wang, Qingyun Sun, Philip S. Yu, *Fellow, IEEE,* and Lifang He

Abstract—Graph embedding is essential for graph mining tasks. With the prevalence of graph data in real-world applications, many methods have been proposed in recent years to learn high-quality graph embedding for various types of graphs, among which the Generative Adversarial Networks (GAN) based methods attract increasing attention among researchers. However, most GAN-based generator-discriminator frameworks randomly generate the negative samples from the original graph distributions to enhance the training process of the discriminator without considering the noise. In addition, most of these methods only focus on the explicit graph structures and cannot fully capture complex semantics of edges such as various relationships or asymmetry. In order to address these issues, we propose a robust and generalized framework named AGE. It generates fake neighbors as the enhanced negative samples from the implicit distribution, and enables the discriminator and generator to jointly learn robust and generalized node representations. Based on this framework, we propose three models to handle three types of graph data and derive the corresponding optimization algorithms, namely the UG-AGE and DG-AGE for undirected and directed homogeneous graphs, respectively, and the HIN-AGE for heterogeneous information networks. Extensive experiments show that our methods consistently and significantly outperform existing state-of-the-art methods across multiple graph mining tasks.

Index Terms—Graph representation learning, generative adversarial networks, directed graph, heterogeneous information networks.

1 Introduction

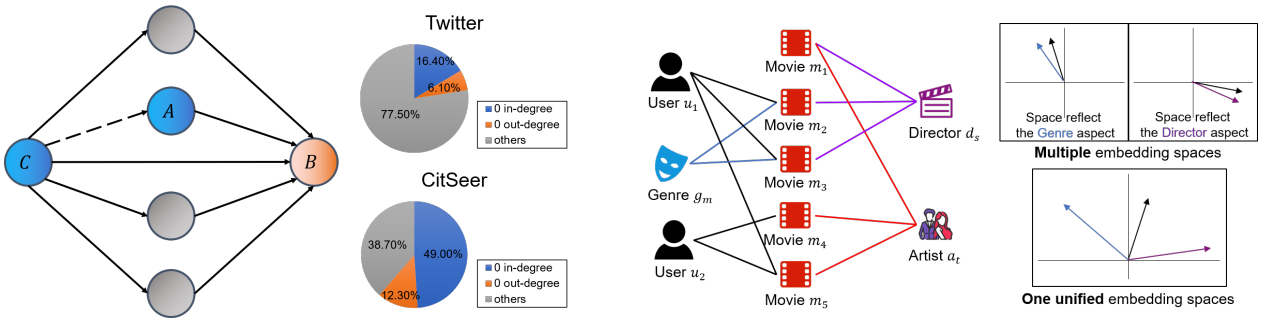
Graph representation learning aims to learn a low-dimensional vector of each node in a graph, and has gained increasing research attention recently due to its broad mining tasks, such as link prediction [1], graph reconstruction [2], and node classification [3]. Recently, many graph representation learning methods have been proposed for various types of graphs. These methods can be roughly divided into three types including matrix factorization based methods [4], [5], [6], random walk based methods [7], [8], [9], [10], and deep learning based methods [11], [12], [13], [14]. Most of these methods rely on strict proximity measures [15] and low rank assumption of the graph adjacent matrix. They focus on representing both structures and features information of the graph. However, for different tasks, it is necessary to model data as graphs with semantic information in real-world scenarios. These latent semantic information may perturb the representation learning of graph structure, and thus lead to the over-fitting problem in the learning process. Moreover, most of these methods perform negative sampling from the original graph to speed up and ensure the effect of training. These negative samples are limited to the existing samples of graph, and they are unable to make good use of graph semantic information. Many generative adversarial networks (GAN) based methods [16], [17], [18] have been proposed to solve the above problem by adversarial training regularization. Although these methods can learn robust node representations, their generators focus on learning the discrete node connection distribution in the original graph. The lack of consideration of invisible semantic information leads to the lower generalization ability of these models.

Meanwhile, many graphs in the real-world contain complex semantics (e.g., social networks, citation networks and web-page networks). For graphs with semantics, we argue that existing works have two major limitations on improving the robustness of model and preserving semantic information at the same time. First, for the structure of graphs with semantic information such as asymmetry, existing methods focus on preserving the structure proximity [15], [19] but ignore the underlying semantic information of the nodes. For the nodes with only out-degree or in-degree edges, their target or source embeddings cannot be effectively trained. Fig. 1a presents a toy example of a directed graph. For predicting the link between nodes A and C in Fig. 1a, A and C are the nodes with only out-degree edges and AC is a potential link. Since the node pair (A, C) is regarded as negative samples, it is hard for existing methods to predict the link AC. As shown in Fig. 1a, the nodes with zero out-degree or in-degrees (e.g., A and B) account for a large proportion of the graph. It means that these nodes with asymmetric semantic information are ubiquitous in some real-world graphs. Second, for the graph with various attributes or types, such as heterogeneous information networks, existing GAN-based

- *J. Li, X. Fu, S. Zhu, H. Peng and Q. Sun are with Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100083, and also with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100083, China. E-mail: {lijx, fuxc, zhusj, penghao, sunqy}@act.buaa.edu.cn*
- *S. Wang is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China. E-mail: szwang@csu.edu.cn.*
- *PS. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607, USA. E-mail: psyu@uic.edu.*
- *L. He is with the Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18015 USA. E-mail: lih319@lehigh.edu.*

Manuscript received April 2021. (Corresponding author: Jianxin Li.)

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944



(a) Directed graph. (b) Heterogeneous information networks.

Fig. 1. (a) An example of asymmetric semantics (directed graph). The two pie charts are statistics from the social network of Twitter and the citation network of CiteSeer, respectively. (b) An example of heterogeneous semantics (heterogeneous information networks). On the left sub-figure, different colored lines represent different types of relationships. On the sub-figure, the differences of each node in multiple relationships embedding spaces and unified embedding spaces.

methods cannot directly and explicitly model the semantic information of different relationships. Mapping different types of nodes into a unified low-dimensional space may lead to significant information loss. The lack of explicit representation of the graph complex semantics may cause many problems, such as embedding distortion and semantic ambiguity [20]. Fig. 1b shows an example of a film network, where the user u_1 has relations with both the musical(genre) g_m and the director d_s. An assumption is that the director d_s is not good at the genre g_m, and he has only made two films of this genre. In other words, g_m and d_s have a low correlation, and it is not completely contained in the network, i.e., "invisible information". If all nodes are embedded into a unified low-dimensional space, u_1 can only be embedded in the middle of g_m and d_s, and make u_1 not similar to g_m and d_s anymore. In summary, the above toy examples show that different semantic information of the graph has various impacts on the representational learning of the graph structures and types. Therefore, learning good representations for graphs with complex semantics becomes extremely challenging.

To address the above challenges, we propose a novel robust and generalized framework for **A**dversarial **G**raph **E**mbedding (**AGE**). Specifically, a generator generates fake neighborhoods for each node from a learnable implicit continuous distribution of node representations. Competition between the generator and discriminator drives both of them to improve their capability until the generated distribution is indistinguishable from the true connectivity distribution. Unlike existing GAN-based methods that sample from the original graph, our method generates fake neighbors as negative samples directly from the implicit distribution of each node. Generating fake neighbors directly from a continuous distribution makes our framework more flexible and scalable because it is not sensitive to different graph structures. Our method can efficiently preserve the semantic information. The source code of this work is publicly available at `https://github.com/RingBDStack/AGE`.

A preliminary version appeared in the proceedings of AAAI 2021 [21], which focuses on adversarial network embedding for directed graphs and only considered the asymmetric (directional) semantics information on graph structure. This journal version has extended it into more various semantics such as edge types or relationship attributes of the graph, and finally integrate a robust and generalized framework named AGE. The core idea is to learn more robust and generalized graph representations by fusing the implicit semantic distribution and designing model according to semantic rules. In practice, we implement corresponding variant models based on AGE for the three typical graphs representation learning: undirected graphs, directed graphs, and heterogeneous graphs. (1) For undirected graphs, we propose a more robust and efficient method named UG-AGE. The generator samples noise from the implicit Gaussian distribution of each node, and directly generates fake neighbors as negative samples for adversarial training. (2) For directed graphs, the challenge is that asymmetric semantic causes the difficulty of learning representations of nodes with zero out-degrees or in-degrees. To preserve asymmetric semantic, we propose an asymmetric-aware model named DG-AGE, which has two generators for generating fake source neighbors and fake target neighbors, respectively. (3) For heterogeneous information networks, the challenge is how to learn the node representation with different relationship semantics. To preserve heterogeneity semantics, we propose a relationship-aware model named HIN-AGE, which can be combined with the translate models [22], [23], [24] and the heterogeneous graph neural networks (HGNNs) [25] with simple modifications to learn the various relationship semantics. Extensive experimental results on real-world graph datasets show that the proposed models consistently and significantly outperform various unsupervised state-of-the-art methods on the tasks of link prediction, node classification and graph reconstruction.

We highlight the advantages of AGE as follows:

- **Robustness and Generality.** AGE generates adversarial samples from the implicit distribution calculated by the latent node representations. It can be generalized to non-existent nodes and not restricted to the original graph.
- **Semantic-preserving.** AGE can modify the implicit distribution according to different graph semantics, which can effectively preserve the complex semantics of the graph.
- **Scalability.** Since the implicit node distribution is continuous, AGE can be generalized to large-scale graphs.
- **Flexibility.** Many other graph embedding methods and external knowledge can be plugged into AGE.

The rest of the paper is organized as follows. We introduce the overall framework in Section 2 and propose three variant models for different graphs in Section 3. The experimental results and analysis are presented in Section 4. We review related work in Section 5. Finally, conclusion and future work are given in Section 6.

2 Overall Framework of AGE

The proposed framework AGE mainly consists of two components: generator and discriminator, which jointly learn the robust and generalized node representations. Specifically, the generator learns a semantic-aware sampling distribution for each node and generates negative samples based on semantic rules, which might result in higher-quality negative samples than those obtained directly from the original data. The discriminator learns to distinguish negative samples from the positive ones. The overall process is described by pseudo-code in Algorithm 1 and Fig. 3 present the workflow of Algorithm 1 on three models we proposed in Chapter 3.

2.1 The Implicit Distribution of Graph

The implicit distribution of the graph contains the prior semantic information of the graph and sampling from it can enhance the quality of negative samples, improving the robustness and generalizability of the model. We use node implicit representations and other auxiliary information to construct alternative noise distributions for different networks in order to learn as much generic and potential graph information as feasible. In general, for a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we obtain the implicit node representation by any graph embedding encoder. The implicit node distribution based on d-dimensional Gaussian distribution $N(\boldsymbol{\mu}, \sigma^2\mathrm{I})$ can be calculated by node representations as:

$$\boldsymbol{\eta} \sim N\left(\mathbf{Z}, \sigma^2\mathrm{I}\right), \tag{1}$$

where $\boldsymbol{\eta}$ is generated noise vector, $\mathbf{Z}$ is the d-dimensional implicit node representations, and $\sigma^2\mathrm{I} \in R^{d\times d}$ is a covariance variable that can be learned. The noise distribution used in our adversarial mechanism is selected based on the implicit feature representation of the graph, which improves the robustness and generalizability of the graph embedding.

2.2 Generator

To use semantic information as much as possible, the generation of fake negative samples needs to conform to semantic rules. Here we propose the basic and semantic-preserved generators for learning graphs with various types.

Basic generator structure. The generator with the implicit node distribution is defined as:

$$G\left(\cdot; \theta^G\right) = f\left(\boldsymbol{\eta}; \theta^f\right), \tag{2}$$

where θ^G is the parameters for generator G, and the input of $G\left(\cdot; \theta^G\right)$ can be the node representations and semantic information of the graph. θ^f is the parameters of the transform function f. The generator samples the noise from the implicit node distribution $\boldsymbol{\eta} \sim N\left(\mathbf{z}_u^T, \sigma^2\mathrm{I}\right)$ according to Eq. 1, where $\mathbf{z}_u \in R^{d\times 1}$ is the embedding vector of node u. The parameter of generator G is $\theta^G = \left\{\boldsymbol{\eta} : \mathcal{G}, \theta^f\right\}$.

Given a node u, the generator outputs the embedding $\mathbf{e}_{u'} \sim G\left(u : \mathcal{G}, \theta^G\right)$ of the fake neighbor node u'. In this way, we obtain a negative node pair (u, u'). For an undirected graph, the implementation of a basic generator is shown in Fig. 2a. First, we get the one-hot encoding of the input node u and then input it to an embedding layer for a dense vector representation. Note that one-hot encoding can be replaced with other embedding models as required. At the same time, the generator randomly samples a noise vector from a Gaussian distribution. The dense vector and the noise vector are added as the vector z as the input of $f(\cdot)$. $f(\cdot)$ outputs the embedding $\mathbf{e}_{u'}$ of the generated fake node.

Semantics preserved generator. For graphs with complex semantics, we need to make full use of their semantic information. Therefore, we need to preserve the semantic of the graph in the generated samples, consistent with the sampling of the positive samples. We design a distribution that fuses semantics and implicit node representations as the noise distribution of the generator. The generator samples random noise from this semantic preserved implicit node distribution and generates fake neighbor nodes for adversarial training. For two typical graph semantic information, asymmetry and heterogeneity, we give the semantic-preserved implicit node distribution respectively as follows:

• *Asymmetry.* To preserve the asymmetric proximity, each node u of a directed graph $\mathcal{G}$ needs to possess two different representations based on two roles (i.e., the source role and target role), represented by $\mathbf{s}_u \in \mathbb{R}^{d\times 1}$ and $\mathbf{t}_u \in \mathbb{R}^{d\times 1}$, respectively, which need to be obtained by joint learning. For this scenario that is difficult to model asymmetry semantics directly such as directed graphs (DG), we use two generators to learn the source and target representations of the nodes, respectively. As shown in Fig. 2b, the source generator G^s and target generator G^t share an implicit node distribution:

$$G^s\left(u; \theta^{G^s}\right) = f^s\left(\boldsymbol{\eta}; \theta^{f^s}\right), G^t\left(u; \theta^{G^t}\right) = f^t\left(\boldsymbol{\eta}; \theta^{f^t}\right), \tag{3}$$

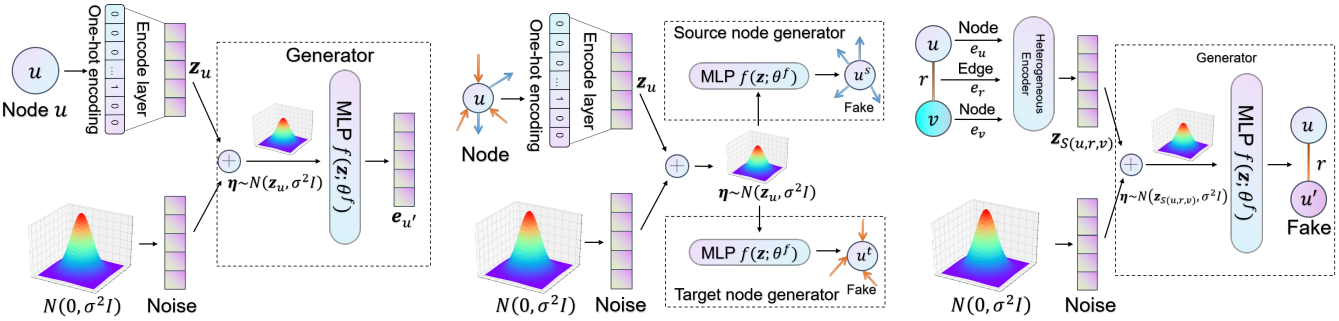
where $\boldsymbol{\eta}$ is sampled from the same implicit node distribution. By jointly learning, two generators can both capture the source and target semantic information of each node.

• *Heterogeneity.* For scenarios that can explicitly model heterogeneous semantics (e.g., heterogeneous information network [26], knowledge graph [27]), we first fuse the node implicit representations and heterogeneous semantic representations as shown in Fig. 2c. Then the noise distribution of the generator can be obtained based on the fused distribution. Formally, the definition of semantics preserved implicit node distribution can be derived from Eq. (2) as:

$$p\left(\boldsymbol{\eta}|\mathbf{s}\right) = N\left(z_{\mathcal{S}(\mathbf{e}_u, \mathbf{e}_r)}, \sigma^2\mathrm{I}\right), \tag{4}$$

where $\mathbf{e}_u$ and $\mathbf{e}_r$ are the node representation of node u and semantics representation of relation r, respectively, and $\mathcal{S}(\cdot)$ is a function that fuses the node and semantics representations. The generator samples noise from this distribution and generates fake nodes as negative samples.

To sum up, as the generator is designed with full consideration of the semantics of a graph and uses continuous implicit distribution to generate fake samples directly, our framework is more adaptive, scalable and computationally efficient for different graphs.



(a) Basic generator (b) Asymmetry preserved generator (c) Heterogeneity preserved generator

Fig. 2. Illustration of the generator of the proposed framework. (a) The basic generator calculates the implicit node distribution and generates the fake neighbors for each node. (b) The Asymmetry preserved generators represent source and target properties of nodes with a shared implicit node distribution for graphs with asymmetry semantic. (c) The Heterogeneity preserved generator fuses representations of nodes and relationships to obtain the implicit node distribution.

Algorithm 1: The process of overall framework.

Input: Graph $\mathcal{G}$, the number of maximum training epochs n^{epoch}, the numbers of generator and discriminator training iterations per epoch n^G, n^D, the number of samples n^s.
Output: θ^G, θ^D.

Initialize θ^G and θ^D;
for $epoch = 0; epoch < n^{epoch}$ **do**
  for $n = 0; n < n^D$ **do**
    // For each node
    for $u \in \mathcal{V}$ **do**
      $\boldsymbol{\eta} \leftarrow$ Eq. (1) ; // Fake neighbor
      $\theta^D \leftarrow$ Eq. (10) ; // Update discriminator
    end
  end
  for $n = 0; n < n^G$ **do**
    // For each node
    for $u \in \mathcal{V}$ **do**
      $\boldsymbol{\eta} \leftarrow$ Eq. (1) ; // Fake neighbor
      $\theta^G \leftarrow$ Eq. (7) ; // Update generator
    end
  end
end

2.3 Discriminator

The discriminator aims to distinguish strongly connected node pairs from weak ones and calculate the possibility that an edge exists between nodes. For any node pair (u, v), we sample the connected node pairs as the positive samples according to adjacency matrix A, and connect the generated fake node u' and the original node u as the negative samples. Then the discriminant function outputs a number from 0 to 1 indicating the probability that node pair is true. In this paper, we use the sigmoid function as the discriminator D:

$$D\left(\mathrm{Enc}(u), \mathrm{Enc}(v); \theta^D\right) = \frac{1}{1 + \exp\left(-\mathbf{e}_u^{\mathrm{T}} \cdot \mathbf{e}_v\right)}, \quad (5)$$

where θ^D is the parameters of D. Any required graph embedding method, such as network embedding methods or graph neural networks (GNNs) can be used as the encoder $\mathrm{Enc}(\cdot)$, making it flexible and easy to extend.

3 Modeling for Different Graphs

In this section, we will introduce the implementation details of our framework on different types of graphs. First, we present the UG-AGE model for the undirected homogeneous graph, which represents basic and simple graph without semantics information. For the graph with rich semantics, we present DG-AGE and HIN-AGE for directed graphs and heterogeneous information networks, respectively.

3.1 UG-AGE: AGE for Undirected Graphs

For undirected homogeneous graphs, we propose UG-AGE based on our framework by a simple modification.

Generator of UG-AGE. According to Eqs. (1) and (2), we define the generator, implicit distribution, and the generator parameters of UG-AGE as follows:

$$\begin{aligned} G\left(u, \theta^G\right) &= f\left(\boldsymbol{\eta}; \theta^f\right), \quad \boldsymbol{\eta} \sim N\left(\mathbf{z}_u^T, \sigma^2 \mathrm{I}\right), \\ \theta^G &= \left\{\mathbf{z}_u : u \in V, \theta^f\right\}. \end{aligned} \quad (6)$$

The generator G outputs the embedding $\mathbf{e}_{u'} \sim G(u, \theta^G)$ of the generated fake neighbor node u'. In this way, a negative sample node pair (u, u') is obtained. The generator G is trained to deceive the discriminator with loss function $\mathcal{L}^G$:

$$\mathcal{L}^G = \mathbb{E}_{u \in \mathcal{V}} \log\left(1 - F\left(u, u'\right)\right), \quad (7)$$

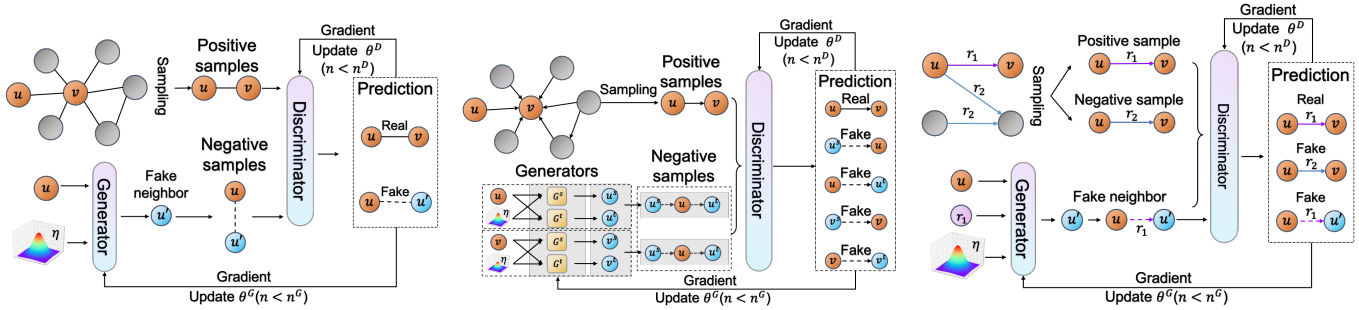
where $F(\cdot)$ is the discriminant function in the discriminator. The output is ranging from 0 to 1, and represents the likelihood of the input node pair (u, u') being positive.

Discriminator of UG-AGE. The discriminator part is divided into two modules: one module is a graph structure reservation module for learning graph structure and the other is an adversarial training module to improve the robustness and generalizability of the model.

• *Graph Structure Preservation Module.* The graph structure preservation module aims to preserve the original graph structure in the low-dimensional embedding space. Many graph embedding methods can be used directly as the graph structure preservation module, such as DeepWalk [7], LINE [9], node2vec [8], etc. Taking DeepWalk as an example, for each node pair (u, v), the loss function $\mathcal{L}_{\mathrm{NE}}^D$ is:

$$\mathcal{L}_{\mathrm{NE}}^D = \log\sigma\left(\mathbf{e}_u^{\mathrm{T}} \cdot \mathbf{e}_v\right) + \sum_{k=1}^{K} \mathbb{E}_{n \sim p(u)} \log\sigma\left(-\mathbf{e}_u^{\mathrm{T}} \cdot \mathbf{e}_n\right), \quad (8)$$

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944



(a) UG-AGE. (b) DG-AGE. (c) HIN-AGE.

Fig. 3. Illustration of Algorithm 1 on UG-AGE, DG-AGE and HIN-AGE. We follow the paradigm of generator-discriminator framework based GAN, which trains the discriminator first and then optimize the generator. (a) UG-AGE: for each node, the generator generates fake neighbors as negative samples. (b) DG-AGE: for each node, the two generators share an implicit distribution and jointly generate a fake source neighbor and a fake target neighbor as the negative samples. (c) HIN-AGE: for nodes with different relationships, the generator generates fake neighbor nodes by the implicit distribution of corresponding relationships.

where $\sigma(\cdot)$ is the sigmoid function, K is the number of negative samples, and $p(u)$ is the sampling distribution of negative samples (usually $p(u) = d_v^{3/4} / \sum_{v \in V} d_v^{3/4}$). Note that the node pair (u, v) comes from the random walk sampling adopted by DeepWalk and can be modified appropriately according to the specific method.

• *Adversarial training module.* The purpose of the adversarial training module is to judge the authenticity of the input node pair. For the input node pair (u, v), we use the sigmoid function as the discriminant function $D(u, v; \theta^D)$ and its output represents the likelihood of the node pair being true. For node u, the generator G generates a fake neighbor node u' and obtains the node pair (u, u'). The discriminant function as the loss function of the adversarial training module can be obtained from Eq. (9):

$$\mathcal{L}_{\text{adv}}^D = \mathbb{E}_{u \in \mathcal{V}} - \log\left(1 - D\left(u, u'\right)\right). \tag{9}$$

Considering the graph structure retention module and the adversarial training module, the loss function of D is

$$\mathcal{L}^D = \mathcal{L}_{\text{NE}}^D + \lambda \mathcal{L}_{\text{adv}}^D, \tag{10}$$

where $\lambda > 0$ is the weight of $\mathcal{L}_{\text{adv}}^D$.

Model Optimization. As shown in Fig. 3a, the model training process is as follows. First, a node $u \in \mathcal{V}$ and its neighbor node $v \in \mathcal{V}$ are selected by random walk to obtain a node pair as a positive sample. For node u, the generator generates a negative sample u' and the negative node pair (u, u') is input into the discriminant function $D(\cdot)$. Second, the loss function $\mathcal{L}^G$ is calculated by the discriminant result of $D(\cdot)$. Finally, we update the parameters of the generator according to $\mathcal{L}^G$. We repeat the above steps to train the generator and discriminator alternatively until convergence.

3.2 DG-AGE: AGE for Directed Graphs

For directed graphs, the key idea is to explicitly learn the asymmetric semantics of the graphs. A critical problem arises that nodes with low in-degree or low out-degree are often difficult to learn due to the edges' asymmetry. To address this problem, we propose DG-AGE to learn more robust source and target vectors for those nodes with low in-degree or low out-degree, even for nodes with zero in-degree or zero out-degree (such as u and v in Fig. 3c).

Asymmetry-Aware Generator. The generator G has three main goals: (1) G should generate corresponding fake samples in a specific direction. Therefore, given a node $u \in \mathcal{V}$, the generator G aims to generate a fake source neighbor u^s and a fake target neighbor u^t. u^s and u^t should be as close as possible to the real neighbor nodes. (2) G should generalize well to non-existent nodes. In other words, the fake nodes u^s and u^t cannot be limited to the original graph. (3) For those nodes with relatively low or zero in-degree or out-degree, G should also be able to effectively generate fake source neighbors and target neighbors.

In order to achieve the first goal, the generator G in DG-AGE contains two generators: the source neighbor generator G^s and the target neighbor generator G^t. For the second and third goals, DG-AGE introduces an implicit variable (noised embedding) $\boldsymbol{\eta}$ shared between G^s and G^t to generate negative samples. DG-AGE applies two transform functions f^s and f^t to the generators to enhance the expression ability of fake samples rather than directly generating samples from the implicit distribution. The formula of generator G is

$$G\left(u; \theta^G\right) = \left\{G^s\left(u; \theta^{G^s}\right), G^t\left(u; \theta^{G^t}\right)\right\}, \tag{11}$$

where θ^{f^s} and θ^{f^t} represent the parameters of f^s and f^t, respectively. The noised embedding $\boldsymbol{\eta}$ serves as a bridge between G^s and G^t. With the help of $\boldsymbol{\eta}$, G^s and G^t update collaboratively to generate better fake source neighbors and target neighbors. According to Eq. (1), we derive $\boldsymbol{\eta}$ from the implicit distribution $\boldsymbol{\eta} \sim N(\mathbf{z}_u^{\mathrm{T}}, \sigma^2 \mathbf{I})$, where $\mathbf{z}_u \in \mathbb{R}^{d \times 1}$ is a learnable variable, representing the implicit representation of $u \in \mathcal{V}$. The parameters of G^s and G^t are $\theta^{G^s} = \{\mathbf{z}_u^{\mathrm{T}} : u \in \mathcal{V}, \theta^{f^s}\}$, $\theta^{G^t} = \{\mathbf{z}_u^{\mathrm{T}} : u \in \mathcal{V}, \theta^{f^t}\}$.

The two generators G^s and G^t aim to deceive the discriminator D by generating fake samples close to real ones. Therefore, the loss function $\mathcal{L}^G$ of generators G^s and G^t is

$$\mathcal{L}^G = \mathbb{E}_{u \in v}\left(\log\left(1 - D\left(u^s, u\right)\right) + \log\left(1 - D\left(u, u^t\right)\right)\right), \tag{12}$$

where u^s and u^t represent the fake source neighbors of node u. The source vector $\mathbf{s}_{u^s}$ and the target vector $\mathbf{t}_{u^t}$ of node u can obtained from $\mathbf{s}_{u^s} \sim G^s\left(u; \theta^{G^s}\right)$, and $\mathbf{t}_{u^t} \sim G^t\left(u; \theta^{G^t}\right)$. The parameters of G^s and G^t can be optimized by minimizing $\mathcal{L}^G$.

TABLE 1
Summary of the noise distribution and the score function of HIN-AGE.

Model	Noise Distribution of Generator	Score Function of Discriminator
TransE	$N(\mathbf{e}_u^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$	$\|\mathbf{e}_u^D + \mathbf{e}_r^D - \mathbf{e}_v^D\|_{L_1/L_2}$
TransH	$N(\mathbf{e}_{u,r}^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$	$\|\mathbf{e}_{u,r}^D + \mathbf{e}_r^D - \mathbf{e}_{v,r}^D\|_{L_1/L_2}$
TransD	$N(\mathbf{e}_u^G\mathbf{M}_{r,u}^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$	$\|\mathbf{e}_u^D\mathbf{M}_{r,u}^D + \mathbf{e}_r^D - \mathbf{e}_v^D\mathbf{M}_{r,v}^D\|_{L_1/L_2}$
HGNN	$N(\mathbf{e}_u^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$	$\mathrm{sigmoid}(\mathrm{HGNN}(\mathbf{e}_u^D)\mathbf{M}_r^D\,\mathrm{HGNN}(\mathbf{e}_v^D))$

Asymmetry-Aware Discriminator. The discriminator D aims to distinguish the negative samples generated by the generator G from the positive inputs sampled from the original graph $\mathcal{G}$. Note that for a given node pair (u, v), D outputs the likelihood of the node v being connected to the node u in the out-degree direction. In particular, the input node pair can be divided into two cases:

• *Positive Sample.* There is indeed a directed edge from u to v on $\mathcal{G}$ (i.e., $(u, v) \in \mathcal{E}$). In this case, the node pair (u, v) is considered to be positive and the loss function is:

$$\mathcal{L}_{\text{pos}}^D = \mathbb{E}_{(u,v)\sim p_{\mathcal{G}}} - \log D(u, v). \tag{13}$$

• *Negative sample.* Given node $u \in \mathcal{V}$, u^s and u^t represent its fake source neighbors and fake target neighbors, which are generated by G^s and G^t, respectively. In this case, node pairs such as (u^s, u) and (u, u^t) are considered to be negative and the loss function is:

$$\mathcal{L}_{\text{neg}}^D = \mathbb{E}_{u\in v} - \log\left(1 - D\left(u^s, u\right)\right) - \log\left(1 - D\left(u, u^t\right)\right). \tag{14}$$

Note that the discriminator D treats the fake node representations $\mathbf{s}_{u^s}$ and $\mathbf{t}_{u^t}$ as unlearnable inputs. Integrating the above two cases together, the discriminator D can be optimized by minimizing the loss function $\mathcal{L}^D$:

$$\mathcal{L}^D = \mathcal{L}_{\text{pos}}^D + \mathcal{L}_{\text{neg}}^D. \tag{15}$$

Model Optimization of DG-AGE. In each training epoch, DG-AGE uses mini-batch gradient descent to train the discriminator D and the generator G alternatively. Specifically, DG-AGE first fixes θ^G and generates corresponding fake neighbors for each node pair of the graph to optimize θ^D. Then, DG-AGE fixes θ^D and each node generates fake neighbor nodes close to the real ones to optimize θ^G under the guidance of D. The generator and discriminator conduct adversarial training until DG-AGE converges.

3.3 HIN-AGE: AGE for Heterogeneous Information Networks

For heterogeneous information networks, an essential problem is how to explicitly model the various relationship semantics of the graph. To preserve different relationship semantics, given a node $u \in \mathcal{V}$ and a relation $r \in \mathcal{R}$, we need to generate a fake node u' that may be connected to u with a relationship r in the context.

Relationship-Aware Generator. The generator $G(\cdot; \theta^G)$ has two main goals. First, G can generate negative nodes close to the real sample. Second, G must be relationship-aware and the generated fake neighbor u' should be as close to the real node as possible under this relationship.

In order to meet the above requirements, we design HIN-AGE model based on two commonly used heterogeneous graph encoders: Translate models [22], [23], [24] and Heterogeneous Graph Neural Networks (HGNNs) [25].

• *Translate Model Based Encoder.* We consider designing our encoder based on three commonly used models in the knowledge graph: TransE [22], TransH [23] and TransD [24]. Specifically, we first obtain initial embeddings of nodes and edges from a translate model as the encoder.

According to different translate models, the generator uses the corresponding Gaussian distribution, as shown in Table 1. Taking the TransE method as an example, the noise distribution $N(\mathbf{e}_u^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$ is a Gaussian distribution with mean value $\mathbf{e}_u^G + \mathbf{e}_r^G$ and covariance $\sigma^2\mathbf{I} \in \mathbb{R}^{d\times d}$.

• *HGNNs Based Encoder.* In recent years, HGNNs are widely used in heterogeneous graphs due to strong power and excellent performance. To further demonstrate the generality of our framework, we also present an HGNNs version of HIN-AGE based on the Simple-HGN [25], a simple and effective HGNNs method. Simple-HGN is an enhanced version of GAT [28] for the heterogeneous graph, and consists of three well-known techniques: learnable edge-type embedding, residual connections, and L_2 normalization on the output embeddings. We calculate the relationship embedding $\mathbf{e}_r^G$ with different semantics by learnable edge-type embedding, and the nodes and edges representations:

$$\begin{aligned}
\hat{\alpha}_{uv} &= \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^T\left[\boldsymbol{W}\mathbf{h}_u \,\|\boldsymbol{W}\mathbf{h}_v\|\, \boldsymbol{W}_r\mathbf{e}_r^G\right]\right)\right)}{\sum_{k\in\mathcal{N}_u}\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^T\left[\boldsymbol{W}\mathbf{h}_u \,\|\boldsymbol{W}\mathbf{h}_k\|\, \boldsymbol{W}_r\mathbf{e}_r^G\right]\right)\right)}, \\
\alpha_{uv}^{(l)} &= (1-\beta)\hat{\alpha}_{uv}^{(l)} + \beta\alpha_{uv}^{(l-1)}, \\
\boldsymbol{e}_u^G &= \mathrm{Norm}_{L2}\left(\sigma\left(\sum_{k\in\mathcal{N}_u}\alpha_{uk}^{(l)}\boldsymbol{W}^{(l)}\mathbf{e}_k^{D^{(l-1)}} + \mathbf{e}_u^{D^{(l-1)}}\right)\right),
\end{aligned} \tag{16}$$

where $\boldsymbol{W}_r$ is a learnable matrix to transform type embeddings, and $\beta \in [0, 1]$ is a hyperparameter for scaling factor.

Intuitively, the mean value represents the representation vector of fake nodes that may be connected to u through the relationship r, and the covariance represents the potential deviation. For generators, HIN-AGE uses a transform function f instead of direct sampling for higher quality negative samples, and the generator can be formulated as follows:

$$G\left(u, r; \theta^G\right) = f\left(\boldsymbol{\eta}; \theta^f\right), \tag{17}$$

where $\boldsymbol{\eta} \sim N(\mathbf{e}_u^G + \mathbf{e}_r^G, \sigma^2\mathbf{I})$, θ^f is the parameters of f and $\theta^G = \{\mathbf{e}_u^G : u \in \mathcal{V}, \mathbf{e}_r^G : r \in \mathcal{R}, \theta^f\}$ is the parameters of G. The HIN-AGE generator structure is shown in Fig. 2c. The parameters θ^G can be optimized by minimizing $\mathcal{L}^G$:

$$\mathcal{L}^G = \mathbb{E}_{(u,r)\sim \mathrm{p}_{\mathcal{G}}, \boldsymbol{\theta}_{u'}\sim G(u,r;\theta^G)}\log\left(1 - D\left(\mathbf{e}_{u'} \mid u, r\right)\right). \tag{18}$$

Relationship-Aware Discriminator. For heterogeneous information networks, the discriminator aims to distinguish between real and fake nodes under a given relationship. Specifically, given a heterogeneous information network $\mathcal{G}$ and a relation r, the discriminator $D(\mathbf{e}_v \mid u, r; \theta^D)$ outputs the likelihood of sample v being connected to u under r. It can be quantified as the score function as shown in Table 1. Given a node u and a relation r, sample a node v. Each triple (u, r, v) belongs to one of the following three cases:

• *Real nodes connect under a real relation:* (u, r, v). The nodes u and v connect under the relation r in the heteroge-

TABLE 2
Adversarial training complexity of UG-AGE, DG-AGE and HIN-AGE.

Model	Time Complexity	Space Complexity
UG-AGE	$O(n^s \cdot (n^D + n^G) \cdot \lvert\mathcal{V}\rvert \cdot d^2)$	$O(d \cdot \lvert\mathcal{V}\rvert + \lvert\mathcal{E}\rvert)$
DG-AGE	$O(n^s \cdot (n^D \cdot \lvert\mathcal{E}\rvert + n^G \cdot \lvert\mathcal{V}\rvert) \cdot d^2)$	$O(2 \cdot d \cdot \lvert\mathcal{V}\rvert + \lvert\mathcal{E}\rvert)$
HIN-AGE	$O(n^s \cdot (n^D + n^G) \cdot \lvert\mathcal{E}\rvert \cdot d^2)$	$O(d \cdot \lvert\mathcal{V}\rvert + (d+1) \cdot \lvert\mathcal{E}\rvert)$

neous information network $\mathcal{G}$. Such triples can be modeled by the following loss function:

$$\mathcal{L}_1^D = \mathbb{E}_{(u,v,r)\sim p_{\mathcal{G}}} - \log D(\mathbf{e}_v^D \mid u, r). \tag{19}$$

In this case, the triple $(u, r, v) \sim p_{\mathcal{G}}$ is sampled from $\mathcal{G}$ and the discriminator should judge them as positive.

• *Real nodes connect under a fake relation:* (u, r', v). A node pair u and v connect in the fake relationship $r' \neq r$. The discriminator should judge them as negative because their connection does not match the given relation r. The loss function of this case is:

$$\mathcal{L}_2^D = \mathbb{E}_{(u,v)\sim p_g, r'\sim p_{R'}} - \log(1 - D(\mathbf{e}_v^D \mid u, r')). \tag{20}$$

In this case, a pair of nodes (u, v) is sampled from $\mathcal{G}$ and the fake relation r' is generated by uniformly sampling from $\mathcal{R}' = \mathcal{R} \setminus \{r\}$.

• *Fake nodes connect under a real relation:* (u, r, v'). Given a node $u \in \mathcal{V}$ and a relation r, the generator $G(u, r; \theta^G)$ generates a fake neighbor v' for u under the relation r. Similarly, the discriminator should judge this triple as negative, and the loss function is as follows:

$$\mathcal{L}_3^D = \mathbb{E}_{(u,r)\sim p_{\mathcal{G}}, e_{v'}\sim G(u,r;\theta^G)} - \log\left(1 - D\left(\mathbf{e}_{v'} \mid u, r\right)\right). \tag{21}$$

Note that the embedding $\mathbf{e}_{v'}$ of fake neighbor v' is sampled from the distribution learned by the generator G. The discriminator D just treats $\mathbf{e}_{v'}$ as an unlearnable input and only optimizes its own parameters θ^D.

We consider the above three cases and integrate their loss functions to train the discriminator. The parameters θ^D of the discriminator can be optimized by minimizing $\mathcal{L}^D$:

$$\mathcal{L}^D = \mathcal{L}_1^D + \mathcal{L}_2^D + \mathcal{L}_3^D. \tag{22}$$

Model Optimization of HIN-AGE. We adopt an iterative optimization strategy to train HIN-AGE. In each iteration, the generator and the discriminator are alternately trained. Specifically, we first fix θ^G and generate fake samples to optimize θ^D for the discriminator training. Then, we fix θ^D and optimize θ^G to generate better fake samples. Repeat the above process for some iterations until the model converges.

3.4 Model Complexity Analysis

In this section, we analyze the time complexity and space complexity of the adversarial training module in the proposed three models (i.e., UG-AGE, DG-AGE, and HIN-AGE). For the three models based on our framework, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the input graph, n^s is the number of samples, n^G and n^D are the numbers of training iterations of the generator and discriminator respectively, and d is the dimension of the node embedding vectors. The detailed explanation and analysis for our models are shown in Table 2.

Although the three models differ in time and space complexity, the overall complexity of the adversarial training module is linear to the number of nodes and edges. In conclusion, our framework is both time and space efficient and is scalable for large-scale graphs.

TABLE 3
Statistics of the datasets. (UG: homogeneous undirected graph; DG: directed graph; HIN: heterogeneous information networks.)

Dataset	#Nodes	#Edges	Avg. degree	#Node classes	#Graph type
Cora	2,708	5,278	3.90	7	UG/DG
Citeseer	3,264	4,551	2.79	6	UG
Facebook	6,637	249,967	37.66	3	UG
Ogbn-products	2,449,029	61,859,140	50.5	47	UG
CoCit	44,034	195,361	8.86	15	DG
Twitter	465,017	834,797	3.59	-	DG
Epinions	75,879	508,837	13.41	-	DG
Google	15,763	171,206	21.72	-	DG
DBLP	37,791	170,794	9.04	4	HIN
Yelp	3,913	38,680	19.77	3	HIN
Aminer	312,776	599,951	3.84	6	HIN
Ogbn-mag	1,939,743	21,111,007	21.7	349	HIN

4 Experiment

In this section, we conduct extensive experiments on several datasets to investigate the performance of UG-AGE, DG-AGE and HIN-AGE, respectively.

4.1 Datasets and Experiment Setting

We evaluate the proposed framework on three types of graphs including undirected and directed homogeneous networks, heterogeneous information networks. The statistics of these datasets are summarized in Table 3.

Undirected graph. *Cora* [29] and *Citeseer* [30] are citation networks of academic papers, where nodes are papers, edges are the citation relationships between papers, and labels are the conferences in which papers are published. *Facebook* [31] is a social network where nodes are users and edges are the relationships between users.

Ogbn-products [32] is a large-scale product co-purchasing network of Amazon where nodes are products and the edges indicate that two products are purchased together.

Directed graph.

Unlike the above scenario of undirected graphs, for citation networks *Cora* [29] and *CoCit* [2], we consider the direction of the citation relationships between papers.

For social network *Twitter* [33], nodes represent users and directed edges represent following relationships between users. For trust network *Epinions* [34], nodes represent users and directed edges represent trust between users. For hyperlink network *Google* [35], nodes represent pages and directed edges represent hyperlink between pages.

Heterogeneous information network. *DBLP* [36] and *Aminer* [37] are scholar networks where nodes are papers, authors and venues, edges are authorships and papers' venues. *Yelp* [36] is a social network where nodes are users, businesses, cities, and categories, and edges are user-user, users' reviews, business-city, and businesses' categories.

Ogbn-mag [32] is a large-scale heterogeneous network of the Microsoft Academic Graph containing four types of nodes and four types of directed edges.

The parameter settings of all baselines follow the settings in the original model. The number of walks, walk length and window size are set to 10, 80 and 10 for comparison. node2vec is optimized with grid search over its return

TABLE 4
Summary of link prediction AUC scores (%), node classification Accuracy(%) and Macro-F1 scores (%) on undirected homogeneous graphs. (Result: average score ± standard deviation; **Bold**: best; Underline: runner-up.)

Method	Cora			Citeseer			Facebook			Ogbn-products		
	AUC	Micro-F1	Macro-F1	AUC	Micro-F1	Macro-F1	AUC	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1
DeepWalk [7]	80.6±0.12	77.8±0.39	76.4±0.66	73.6±0.31	50.1±0.08	46.8±0.92	85.2±0.37	80.3±0.68	78.2±0.04	71.1±0.15	71.1±0.15	33.0±0.65
LINE-1 [9]	73.8±0.30	78.0±0.61	76.5±0.16	72.4±0.30	52.1±0.34	47.5±0.22	82.3±0.84	79.6±0.22	75.6±0.43	63.4±0.38	63.4±0.38	25.2±0.52
node2vec [8]	83.8±0.09	78.4±1.48	77.1±0.90	77.6±0.10	53.8±1.28	50.1±0.70	85.5±0.43	81.2±0.56	79.3±0.54	72.4±0.11	72.4±0.11	33.3±0.37
GraphGAN [16]	82.5±0.64	76.4±0.21	76.8±0.34	74.5±0.02	49.8±1.02	45.7±0.13	84.2±0.23	78.5±1.33	75.2±0.96	-	-	-
ANE [17]	83.1±0.57	78.5±0.51	77.0±1.40	75.0±1.20	50.2±0.12	49.5±0.61	85.6±1.35	82.1±0.14	79.6±0.38	72.5±0.30	72.5±0.30	34.3±0.29
GAE [38]	86.1±0.87	80.9±0.99	79.6±1.32	87.3±1.26	58.5±0.31	50.4±3.32	86.6±0.13	77.0±0.35	73.1±0.05	75.4±0.66	75.4±0.66	35.4±0.31
VGAE [38]	85.9±0.05	80.0±0.95	78.8±0.97	85.7±2.20	58.8±1.39	55.5±1.34	87.1±0.18	79.5±0.85	77.7±0.30	74.1±0.64	74.1±0.64	35.1±0.40
ARGA [39]	86.9±0.07	81.0±0.03	77.5±0.24	88.6±0.02	59.1±0.35	56.0±0.93	87.5±0.05	82.5±0.21	80.8±0.70	76.9±0.46	76.9±0.46	36.5±0.58
UG-AGE-DW	92.6±0.05	83.6±0.06	82.7±0.07	89.8±0.06	63.5±1.29	59.8±0.32	87.5±0.03	84.5±0.38	82.6±0.09	77.5±0.59	77.5±0.59	37.6±0.37
UG-AGE-NV	**92.6±0.20**	83.9±0.57	83.1±0.09	**90.3±0.01**	64.1±1.02	60.5±0.28	**88.3±0.96**	**85.6±0.04**	**83.5±0.72**	78.1±0.58	78.1±0.58	38.2±0.28
UG-AGE-GNN	90.2+0.42	**85.6±0.34**	**84.7±0.20**	88.3±0.53	**64.2±0.40**	**61.6±0.23**	86.2±0.18	84.9±0.35	82.2±0.24	**78.8±0.33**	**78.8±0.33**	**38.8±0.37**

and in-out parameters $(p, q) \in \{0.25, 0.50, 1, 2, 4\}$ on each dataset and task. For each proposed model, we choose parameters by cross-validation and fix the numbers of generator and discriminator training iterations per epoch n^G=5, n^D=15 across all datasets and tasks. For the samples of experiments, we set the same number of positive and negative samples sampled, and generate the same number of fake samples for negative sample enhancement. Throughout our experiments, the default setting of the dimension of node embeddings is 128. The reported results are the average performance of 10 times experiments.

4.2 Baselines and Evaluation Metrics

We compare the proposed UG-AGE, DG-AGE and HIN-AGE with several unsupervised graph embedding methods.

Traditional graph embedding methods. We focus on several classical methods based on random walk. DeepWalk [7] and node2vec [8] learn node embeddings by using different random walk algorithms, and LINE [9] learns large-scale network embedding using first-order and second-order proximities namely LINE-1 and LINE-2, respectively.

GAN-based graph embedding methods. Since our work is based on the GAN framework, we focus on two important GAN-based graph embedding methods. GraphGAN [16] proposes a structure-aware graph softmax function to compute each node's probability and randomly samples the nodes as the generated neighbor. ANE [17] trains a discriminator to push the embedding distribution to a fixed prior.

Unsupervised Graph Neural Networks. For unsupervised graph neural networks, we compare GNN methods based on the autoencoder training framework. GAE [38] is the popular graph autoencoder and VGAE [38] is the extending variational version. ARGA [39] employs adversarial training for graph autoencoders to regularize the latent codes and enforce the latent codes to match a prior distribution. Note that we adopt GNN method [40] as the encoder for all the graph autoencoders.

Directed graph embedding methods. HOPE [6] preserves the asymmetric information of the nodes by approximating high-order proximity. APP [19] proposes a random walk based method to encode Rooted PageRank proximity.

HIN embedding methods. We compare three types of methods: Meta-path based methods (Metapath2vec [10] and HIN2vec [36]), Translate model based methods (TransE [22], TransD [23], TransH [24]) and Heterogeneous graph neural network method (Simple-HGN [25]). Note that we mainly focus on unsupervised learning setting, we use the translate models and HGNN as the encoder for HIN-AGE.

For link prediction, the evaluation metric is the area under curve (AUC) score of the ROC. For node classification, the evaluation metrics are the Accuracy, Micro-F1 score and Macro-F1 score. Since our framework and all baselines are unsupervised learning model, we randomly sample a fraction of the labeled nodes as the training data and train a standard one-vs-rest L_2-norm regularized logistic regression classifier. Then we predict the labels of the other nodes. For graph reconstruction, the evaluation metric is Precision.

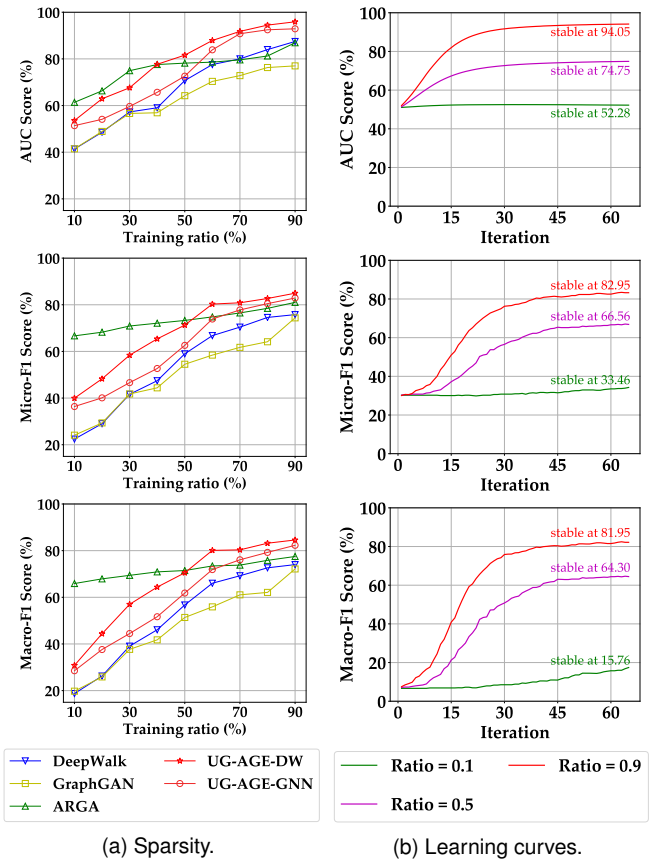
4.3 Performance Evaluation of UG-AGE

For evaluation, we compare it with several methods, including traditional graph embedding methods and GAN-based graph embedding methods. We also propose two versions of UG-AGE implemented by using DeepWalk, node2vec and GCN [40] for network structure retention, named UG-AGE-DW, UG-AGE-NV and UG-AGE-GNN, respectively.

• **Performance Analysis.** We perform two tasks, link prediction and node classification. For link prediction, we predict missing edges given a graph with a fraction of removed edges. Specifically, we remove 20% of edges as positive samples and randomly select node pairs with unconnected edges as negative samples in the test set. Note that we make sure that no node is isolated to avoid meaningless embedding vectors when randomly removing edges. The ratio of training to test data is 8:2. For node classification, we evaluate the proposed UG-AGE and baseline methods on four undirected homogeneous graph datasets *Cora*, *Citseer*, *Facebook* and *Ogbn-products*. Note that we follow OGB [32] default setting and only evaluate the node classification task on the *Ogbn-products*. We use the top 8% nodes of product sales ranking of *Ogbn-products* for training, next 2% nodes for test validation, and the rest for testing. *Ogbn-products* is more challenging for evaluating the scalability and generalization of the model.

We report the results of all models in Table 4. We can notice that compared with DeepWalk and node2vec, UG-AGE-DW and UG-AGE-NV with the adversarial training module can achieve higher AUC scores and F1 scores, which verifies the adversarial training module is considerably beneficial to preserve the graph structure. UG-AGE-DW and UG-AGE-NV both perform better than DeepWalk, node2vec and LINE on the three datasets. The underlying reason is that these baselines generate negative samples by

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944



(a) Sparsity. (b) Learning curves.

Fig. 4. Performance and learning curves of UG-AGE on *Cora* for link prediction and node classification.

randomly sampling from the original graph, which are not strong enough and can be easily identified by the model, while the negative samples of UG-AGE-DW and UG-AGE-NV are generated by implicit node distributions. Compared with GAN-based methods (GraphGAN, ANE) and graph autoencoders (GAE, VGAE, ARGA), our UG-AGE-DW and UG-AGE-NV also show better performance. It shows that the implicit distributions of graphs provide better inductive bias than the common prior distribution (e.g. Gaussian distribution). Moreover, the results demonstrate that our UG-AGE has good extensibility and scalability. For the classification task on the large-scale dataset *Ogbn-products*, UG-AGE-GNN shows better performance. The reason is that the GNN-based encoder fully learns the features of the node neighbors and also shows that our framework has good scalability and generalization.

• **Sparsity and Learning Analysis.** To verify the advantages of our model, we analyze the link prediction and node classification performance of UG-AGE under different conditions on *Cora*. We randomly sample nodes of different ratios (from 10% to 90%) as the training data and randomly sample 10% nodes outside the training set as the test data.

Fig. 4.3 illustrates the performance and learning curves of UG-AGE with different training ratios on *Cora*. Fig. 4a shows that UG-AGE outperforms baselines under all training ratios, which indicates that the adversarial training module can significantly improve the performance of the model. We can observe that although the autoencoder (ARGA) has better performance with a small ratio of training, UG-AGE outperforms it rapidly. The reason is the GAN-based method needs enough samples to fit the Gaussian distribution into the data distribution. In addition, we find that the UG-AGE-GNN has lower speedups for different training ratios, which may be due to the fact that node representations are more dependent on the neighborhood aggregation. Fig. 4b shows the test performances of UG-AGE in the learning process. The results show that the performance of UG-AGE improves rapidly with the increase of the training ratio, indicating that it has better generalization ability, especially in node classification.

4.4 Performance Evaluation of DG-AGE

For directed graph, we compare DG-AGE with traditional graph embedding methods, directed graph embedding methods, and GAN-based graph embedding methods. We also construct two variants of DG-AGE to demonstrate the effectiveness and flexibility of our framework. The DG-AGE* uses only one generator G^t to generate target neighborhoods of each node, and the DG-AGE-GNN uses GCN-layer as the node encoder. Note that we do not report the results of GraphGAN on *Twitter* and *Epinions*, since it cannot run on these two large datasets.

• **Link Prediction.** Given a graph with a fraction of removed edges, we predict missing edges. A fraction of edges are removed randomly to serve as test split while the remaining network are utilized for training. Specifically, we remove 50% edges in *Cora*, *Epinions* and *Google*, and 40% edges in *Twitter*. Since we are interested in both the existence and the direction of the edge, we reverse a fraction of positive node pairs to replace the original negative samples if the edges are not bi-directional. The reversed ratio $\gamma \in (0, 1]$ means the fraction of positive edges from the test data reversed as negative examples and 0 corresponds to the classical undirected graph setting where all the negative edges are sampled from random node pairs.

We summarize AUC scores of all methods in Table 5. We can observe that the performances of all undirected graph embedding methods (including GAN-based and autoencoder-based methods) decrease rapidly with the increase of reversed positive edges because they cannot model the asymmetric proximity. The directed graph embedding methods like HOPE and APP show poor performance on *Cora* and *Epinions*. The reason is that these methods treat the source role and target role of one node separately, resulting in less robustness. Moreover, DG-AGE outperforms DG-AGE* as it utilizes two generators mutually updating each other for more robust source and target vectors. DG-AGE-GNN is the runner-up of comprehensive performance, because the neighbors of the reversed edges affect the node aggregation of the GCN encoder. Overall, DG-AGE shows more robustness and outperforms all baselines across datasets for link prediction.

• **Node Classification.** For node classification, we evaluate DG-AGE on two directed graph datasets *Cora* and *CoCit*. Note that for the methods using both source and target embedding matrices, we set the dimension d of each embedding to 64 and concatenate the two embedding vectors into a 128-dim vector to represent each node.

Fig. 4.4 summarizes the experimental results with various training ratios. Our DG-AGE consistently outperforms

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944

TABLE 5
Summary of link prediction AUC scores (%) on directed graphs with various fractions of reversed positive edges.
(Result: average score ± standard deviation; **Bold**: best; Underline: runner-up.)

Method	Cora			Twitter			Epinions			Google		
	0%	50%	100%	0%	50%	100%	0%	50%	100%	0%	50%	100%
DeepWalk [7]	84.9±1.39	68.1±0.43	52.9±0.12	50.4±0.67	50.3±0.21	50.3±0.01	76.6±1.21	67.9±0.54	66.6±0.12	83.6±2.61	72.1±0.65	66.5±0.32
LINE-1 [9]	84.7±0.63	68.0±0.25	52.5±0.06	53.1±0.45	51.5±0.13	50.0±0.01	78.8±0.52	69.8±0.26	68.5±0.05	89.7±0.82	72.7±0.45	65.1±0.21
node2vec [8]	**85.3±1.07**	65.5±0.35	52.1±0.09	50.6±0.75	50.5±0.33	50.3±0.01	89.7±0.31	74.6±0.12	72.6±0.02	84.3±1.13	70.5±0.53	64.3±0.26
GraphGAN [16]	51.6±0.67	51.3±0.31	51.2±0.12	-	-	-	-	-	-	71.3±2.37	61.1±1.59	56.2±1.13
ANE [17]	72.8±0.53	61.4±0.28	51.5±0.07	49.7±0.53	49.8±0.29	50.0±0.02	85.5±2.15	69.2±0.74	66.9±0.24	76.1±1.86	63.7±0.83	57.8±0.53
GAE [38]	83.5±0.73	72.1±0.31	55.3±0.78	59.6±0.87	51.4±0.56	50.1±1.03	84.0±1.19	71.9±1.93	69.5±0.63	74.8±0.51	69.8±0.92	68.2±0.04
VGAE [38]	84.2±0.20	73.0±0.61	58.2±0.59	62.5±0.15	59.9±0.30	59.6±0.10	82.4±0.36	71.9±0.16	68.1±0.64	76.2±0.50	67.5±0.37	66.0±0.17
ARGA [39]	81.3±0.57	73.2±0.18	66.7±0.50	64.1±0.65	61.2±1.73	60.4±0.40	73.1±0.76	64.7±0.79	63.5±0.86	77.3±0.19	69.4±0.11	68.9±0.57
LINE-2 [9]	69.3±0.47	72.1±0.23	74.3±0.05	95.6±0.37	95.7±0.13	95.8±0.01	58.1±0.67	67.1±0.52	68.4±0.41	77.4±0.24	85.2±0.17	89.0±0.13
HOPE [6]	77.6±1.53	74.2±0.65	71.5±0.42	98.0±0.63	97.9±0.42	97.8±0.03	79.6±1.13	71.7±0.57	70.5±0.23	87.5±0.46	85.6±0.32	84.6±0.38
APP [19]	76.6±0.83	76.4±0.41	76.2±0.11	71.6±0.57	70.1±0.36	68.7±0.01	70.5±0.47	71.3±0.23	71.4±0.09	92.1±0.21	86.4±0.15	81.0±0.13
DG-AGE*	83.0±0.91	83.3±0.53	83.5±0.25	99.4±0.27	99.3±0.12	99.2±0.01	92.7±0.85	80.0±0.36	78.2±0.21	91.6±0.63	89.2±0.47	87.7±0.26
DG-AGE	85.1±0.63	**86.7±0.31**	**88.3±0.11**	**99.7±0.15**	**99.7±0.09**	**99.7±0.01**	**96.1±0.51**	**86.4±0.25**	**85.1±0.11**	**92.3±0.52**	**93.4±0.36**	**94.4±0.23**
DG-AGE-GNN	83.18±0.15	85.99±0.52	86.74±0.23	98.32±0.30	99.15±0.24	97.73±0.53	95.72±0.07	85.65±0.37	84.23±0.07	90.50±0.48	92.61±0.10	91.67±0.03

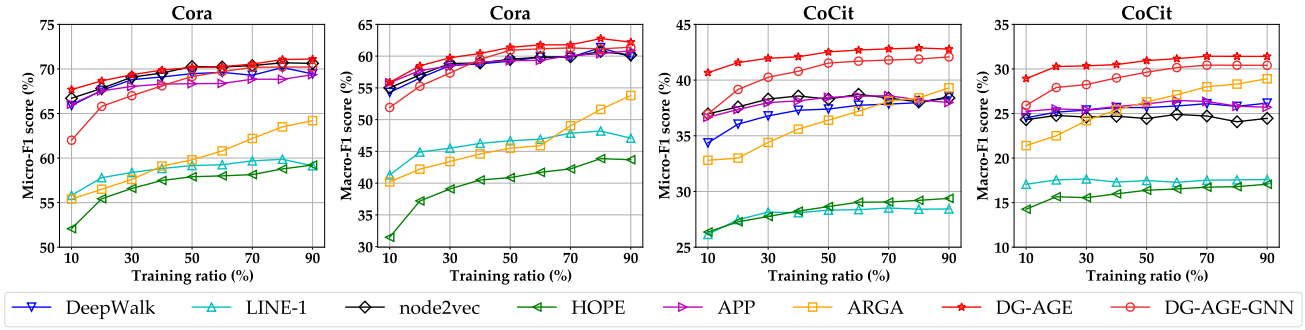


Fig. 5. Node classification performance of DG-AGE on *Cora* and *CoCit*.

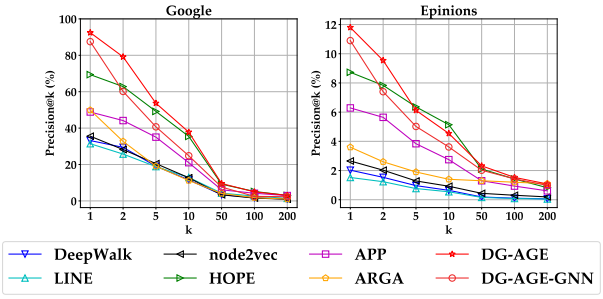
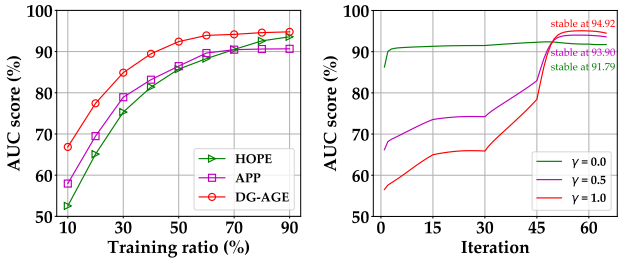


Fig. 6. Graph reconstruction of DG-AGE on *Google* and *Epinions*.



(a) Sparsity. (b) Learning curves.

Fig. 7. Link prediction and learning curves of DG-AGE on *Google*.

all baseline methods across all training ratios on both datasets, which demonstrates that DG-AGE can effectively capture the neighborhood information in a robust manner through adversarial learning. Moreover, although it is difficult to generalize tasks with different semantics with a particular proximity measure, adversarial training still contributes to the generalization of learning. For example, although ARGA is difficult to learn neighborhood information with asymmetric semantics, adversarial training still benefits performance improvement.

• **Graph Reconstruction.** Considering that the direction of edges may directly affect the topology structure of directed graphs, we perform the graph reconstruction task on *Google* and *Epinions* and randomly sample 10% nodes of each dataset as the test data. Then we reconstruct the graph edges based on the k-nearest target neighbors with a given k ranked by reconstructed proximity.

We plot the average precisions corresponding to different values of k in Fig. 4.4. The results show that DG-AGE mostly outperforms all baselines on both datasets. On *Epinions*, HOPE outperforms DG-AGE when k=5 and k=10. It may be because HOPE uses high-order proximity as the weights of directed edges to reconstruct more edges. On *Google*, DG-AGE shows an improvement of around 33% with k=1 over the second best method HOPE. Some methods (e.g., node2vec, ARGA) that focus on undirected graphs exhibit good performance in link prediction but show poor performance in graph reconstruction. This is because graph reconstruction is harder than link prediction and the model needs to distinguish a small number of positive edges from a large number of negative edges. In particular, the precision of ARGA decreases rapidly on Google with k increases. It further proves the benefit of adaptation to semantic rules.

• **Sparsity and Learning Analysis.** For the directed graph, we analyze the performance of models under different graph sparsity levels and the converging performance of DG-AGE on a denser dataset *Google*.

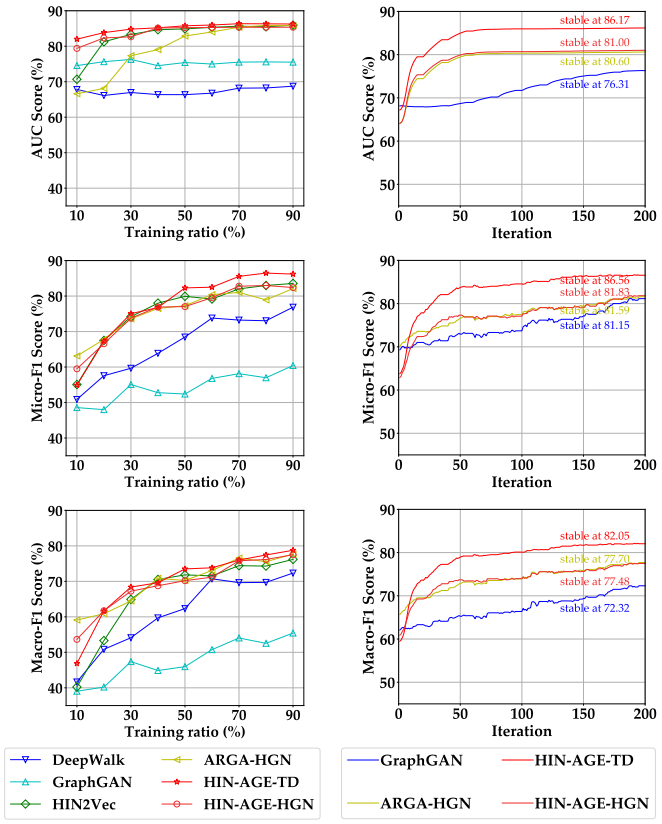
We first investigate how the graph sparsity affects the three directed graph embedding methods HOPE, APP and DG-AGE. These training settings are the same as them in the link prediction task and 50% positive edges of test set are reversed to form negative edges. We randomly delete

TABLE 6
Summary of link prediction AUC scores (%), node classification Accuracy and Macro-F1 scores (%) on heterogeneous information networks. (Result: average score ± standard deviation; **Bold**: best; Underline: runner-up.)

Method	DBLP			Yelp			AMiner			Ogbn-mag		
	AUC	Micro-F1	Macro-F1	AUC	Micro-F1	Macro-F1	AUC	Micro-F1	Macro-F1	Accuracy	Micro-F1	Macro-F1
DeepWalk [7]	56.3±0.72	92.0±0.71	92.4±0.59	78.3±0.22	82.6±0.81	75.5±0.67	51.8±0.81	95.2±0.30	94.6±0.17	24.3±0.24	24.3±0.24	11.9±0.16
LINE-1 [9]	72.2±0.30	92.4±0.04	92.1±0.85	79.7±0.95	82.3±0.38	74.4±0.15	64.1±0.58	97.8±0.25	97.1±0.04	26.2±0.53	26.2±0.53	12.8±0.16
LINE-2 [9]	65.0±0.38	91.4±0.20	91.7±0.80	67.5±1.12	75.9±0.57	55.2±0.68	51.1±0.11	94.7±0.19	93.4±0.59	25.8±0.32	25.8±0.32	12.1±0.64
GraphGAN [16]	53.3±0.62	92.0±0.39	92.1±0.11	76.3±0.57	81.0±0.16	72.7±0.70	-	-	-	-	-	-
ANE [17]	54.3±0.47	91.4±0.17	91.5±0.47	73.3±0.39	82.3±0.41	76.2±0.92	52.8±0.16	92.6±0.23	92.0±0.13	28.2±0.89	28.2±0.89	15.5±0.71
GAE-HGN [25], [38]	70.8±0.21	92.9±0.08	91.4±0.37	75.5±0.36	81.2±0.14	77.2±0.11	71.6±0.27	94.2±0.05	93.3±0.92	35.2±0.11	35.2±0.11	18.7±0.24
VGAE-HGN [25], [38]	71.4±0.12	92.1±0.33	91.1±0.44	78.6±0.48	83.0±0.09	78.5±0.08	75.8±0.73	94.9±0.05	93.7±0.31	35.8±0.27	35.8±0.27	18.7±0.39
ARGA-HGN [25], [39]	70.5±0.61	92.8±0.26	91.4±0.47	77.4±0.46	82.1±0.40	77.7±0.53	76.2±0.43	95.6±0.28	94.3±0.12	36.5±0.35	36.5±0.35	19.6±0.13
HIN2vec [36]	79.5±0.39	91.4±0.40	91.2±0.48	79.6±0.58	83.5±0.14	76.1±0.50	78.7±0.13	98.0±0.12	97.8±0.05	34.1±0.14	34.1±0.14	18.8±0.21
Metapath2vec [10]	59.2±0.21	92.9±0.34	93.0±0.41	78.0±0.12	79.5±0.86	78.8±0.73	76.2±0.70	98.5±0.28	98.6±0.09	34.4±0.28	34.4±0.28	19.2±0.34
TransE [22]	76.3±0.07	90.2±0.32	91.2±1.05	77.3±0.57	82.5±0.03	75.4±0.76	75.6±0.21	97.1±0.66	96.4±0.35	31.3±0.73	31.3±0.73	16.3±0.48
HIN-AGE-TE	79.1±0.45	91.3±0.28	92.5±0.13	79.9±0.03	84.2±0.35	79.5±0.30	78.7±0.74	97.7±0.13	97.7±0.10	36.8±0.57	36.8±0.57	19.4±0.43
HIN-AGE-TH	81.3±0.55	94.2±0.58	93.9±0.21	81.3±0.09	86.1±0.17	81.2±0.52	81.5±0.13	98.5±0.08	98.8±0.07	37.0±0.35	37.0±0.35	19.2±0.71
HIN-AGE-TD	**83.2±0.28**	**95.2±0.08**	**94.1±0.16**	**82.1±0.15**	**86.6±0.18**	**82.1±0.84**	**83.1±0.84**	**98.7±0.25**	**98.9±0.01**	37.9±0.24	37.9±0.24	21.0±0.15
HIN-AGE-HGN	76.9±0.65	93.1±0.25	91.9±0.45	77.8±0.27	82.4±0.29	77.5±0.36	78.4±0.40	98.9±0.26	98.3±0.31	**39.2±0.61**	**39.2±0.61**	**22.9±0.49**

different ratios of edges from the original graph to construct graphs with different sparsity levels. Fig. 7a shows the results with respect to the training ratio of edges on *Google*. We can see that DG-AGE consistently and significantly outperforms HOPE and APP across different training ratios. Moreover, DG-AGE still achieves much better performance when the network is very sparse. It demonstrates that the proposed DG-AGE, which is designed to jointly learn a node's source vector and target vector, can significantly improve the representation robustness.

Next, we investigate the effects of the training iterations of the discriminator D. Fig. 7b shows the converging performance of DG-AGE on *Google* with different ratios of reversed positive edges of test set (the results on other datasets show similar trends and are not included here). With the increase of iterations of D, the performance of DG-AGE with γ=0 (i.e., random negative edges in test set) keeps stable first and then slightly increases. Besides, the training curve of DG-AGE with γ=1.0 (i.e., all positive edges except bi-directional edges are reversed to create negative edges in the test set) changes every 15 iterations (i.e., one epoch). The training curve of DG-AGE with γ=1.0 rises gently during second epoch (i.e., from the 16-th iteration to the 30-th iteration) for the generator G which is still been poorly trained at the moment. The trend rises steeply in the following epochs where G is being able to generate close-to-real fake samples.



(a) Sparsity. (b) Learning curves.

Fig. 8. Performance with different graph sparsity and learning curves of HIN-AGE on *Yelp*.

4.5 Performance Evaluation of HIN-AGE

In order to evaluate the performance of HIN-AGE, we compare it with several methods, including traditional graph embedding methods, GAN-based graph embedding methods, HIN embedding methods and unsupervised HGNNs. Note that we present three versions (HIN-AGE-TE, HIN-AGE-TH and HIN-AGE-TD) based on translate model and a version (HIN-AGE-HGN) based on HGNN model. For HGNNs, we mainly consider Simple-HGN [25] as the backbone, which is a simple and stat-of-the-art method based on GAT [28]. Note that we use the autoencoders (GAE-HGN, VGAE-HGN, ARGA-HGN) with Simple-HGN as the backbone and the unsupervised HGNN baselines. The results of GraphGAN on *AMiner* and *Ogbn-mag* are excluded, because they cannot perform on the large dataset.

• **Performance Analysis.** For link prediction, we predict user-review links in *Yelp* and author-paper links in *DBLP* and *AMiner*. For positive samples, we randomly keep 20% connected node pairs in *Yelp*, *DBLP* and *AMiner* as test set and the remaining 80% as training set. For node classification, we evaluate the proposed HIN-AGE and other baseline methods on *DBLP*, *Yelp* and *AMiner*. Similar to link prediction, we sample 80% labeled nodes as the training data and predict the labels of the other 20% labeled nodes. In addition, as same as the setting of Section 4.3, we also evaluate the scalability and generalization of HIN-AGE on a large-scale heterogeneous information network *Ogbn-mag*

This article has been accepted for publication in IEEE Transactions on Knowledge and Data Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2023.3235944

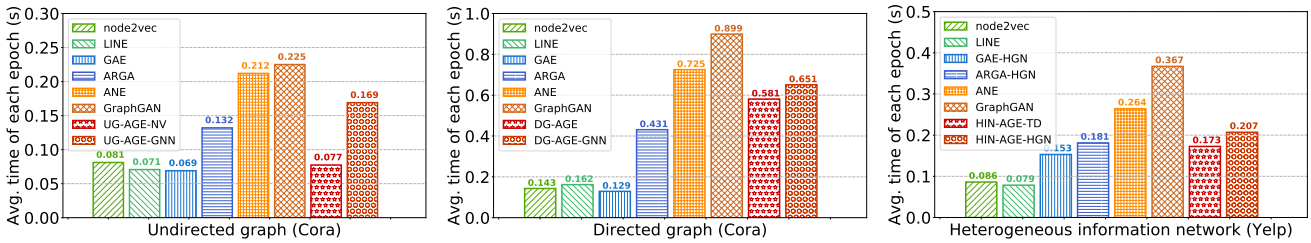


Fig. 9. The average run-time per epoch of link prediction on *Cora* (undirected and directed graph) and *Yelp* (heterogeneous information network).

of the OGB [32]. Note that we add node feature information to the learning of all unsupervised HGNNs methods (GAE-HGN, VGAE-HGN, ARGA-HGN and HIN-AGE-HGN).

As shown in Table 6, HIN-AGE outperforms all baselines. We can observe that the semantic heterogeneity seriously reduces the performance of homogeneous graph neural networks, including traditional, GAN-based, and autoencoder-based graph learning methods. It indicates HIN-AGE can better preserve the heterogeneous semantics. In addition, we observe that HIN-AGE-TD can achieve the best performance compared with other variants of HIN-AGE on *DBLP*, *Yelp* and *AMiner*. It may be because that HIN-AGE-TD uses one vector to represent semantics and another to construct a mapping matrix, which can better preserve the semantic information of multi-classes nodes and multi-relations in heterogeneous networks. For unsupervised HGNNs, we compare our HIN-AGE-HGN with the autoencoder-based HGNNs methods (GAE-HGN, VGAE-HGN, ARGA-HGN). We find that HIN-AGE-HGN has better performance than the autoencoder-based methods on the large-scale dataset, especially on *Ogbn-mag*. The reason is that node representations may depend on the quality of heterogeneous neighbor feature aggregation, which is especially important in large-scale heterogeneous graphs.

- **Sparsity and Learning Analysis.** We further evaluate the generalizability of HIN-AGE with different graph sparse conditions on *Yelp*. We randomly sample 10% to 90% from the original training set as the training data and randomly sample 10% of the remaining nodes as the test data. Fig. 8a shows the results with different training ratios of link prediction and node classification tasks on *Yelp*. It can be observed that HIN-AGE consistently outperforms all baselines for both tasks, even when the training ratio is small. In addition, we can observe that the learning curves of the four variants of HIN-AGE are similar, indicating that the framework is stable in the training process. Compared to other structure-aware adversarial training frameworks (GraphGAN), our framework is stable in the learning process regardless of the any encoders.

4.6 Model Efficiency Analysis.

We conduct experiments with our three models and all baselines for model efficiency analysis. Figure 4.5 illustrates the training time of UG-AGE, DG-AGE, HIN-AGE and baselines on *Cora* and *Yelp* for link prediction. It can be observed that our three models have the best computational efficiency in the GAN-based method (ANE, GraphGAN). In general, our framework can significantly improve the computational efficiency and scalability of network embedding models, which supports our complexity analysis in Section 3.4.

5 Related Work

In this section, we first briefly review the graph representation learning methods. Then we review the graph embedding based on generative adversarial network specifically.

5.1 Graph Representation Learning

Graph representation learning methods fall into three categories: matrix factorization based models, random walk based models and deep learning based models. The matrix factorization based models (e.g., GraRep [4] and M-NMF [5]) first preprocess the adjacency matrix which preserves the graph structure, and then decompose the preprocessed matrix to obtain graph representations. The random walk based models (e.g., DeepWalk [7], LINE [9], PTE [41] and node2vec [8]) sample node sequences to put into Skip-gram model [42] by random walk on the graph and can be unified into the matrix factorization framework with closed forms [43]. In addition, Graph Neural Networks [14], [28], [40] have been widely studied and applied because of their powerful representation capability. However, most of them ignore data noise. The negative samples used are not strong enough, leading to poor robustness.

Some works focus on directed graphs [6], [19], [44], [45], which learn source and target embedding for each node. HOPE [6] derives node-similarity matrix by approximating high-order proximity measures and then decomposes the node-similarity matrix to obtain node embeddings. APP [19] preserves the asymmetric proximity via random walk with restart, which implicitly preserves the Rooted PageRank score for node pairs. NERD [46] generates role-specific node neighbors with a plain alternating random walk strategy and learns node representations in their related source/target nodes. ATP [47] incorporates graph hierarchy and reachability to construct the asymmetric matrix. For directed graph, most methods fail to capture the highly non-linear property in graphs.

The graph representation learning models for homogeneous graphs are not suitable for heterogeneous information network (HIN) [48]. Recent research in HIN embedding can be divided into three categories: random walk based models, knowledge graph embedding models, and heterogeneous graph neural networks. The random walk based methods model structural and semantic correlations in HIN simultaneously, such as metapath2vec [10] and HIN2vec [36]. These methods design meta-path based or specific random walk strategies to obtain the neighborhood of nodes. HERec [49] designs a meta-path based random walk strategy and further integrates node embeddings

into an extended matrix factorization model. The knowledge graph representation learning methods learn low-dimensional embeddings of entities and relations while capture relative semantic meanings [50]. Heterogeneous Graph Neural Network [51], [52], [53] is a powerful graph representation learning method which focuses on aggregating multi-relational information on HINs. However, these methods always need domain knowledge to design meta-paths or walk strategies, which is difficult to apply to complex and large-scale HINs.

5.2 GAN-based Graph Embedding

Recently, Generative Adversarial Network (GAN) [54] attracts increasing attention among researchers due to its impressing performance on the unsupervised tasks. GAN can be considered as playing a game-theoretical min-max game between the generator and the discriminator. Several methods [16], [17], [18], [39], [55], [56], [57], [58] have been proposed to apply GAN for graph embedding to achieve the robustness and generalization of models. GraphGAN [16] samples negative nodes in the sampling distribution. ANE [17] regularizes graph embedding learning, which contains a structure preserving component and an adversarial learning component for obtaining structural properties and robust representations, respectively. NetRA [18] and ARGA [39] adopt adversarially regularized auto-encoders to learn smooth embeddings. ProGAN [55] employs triplets of nodes for discovering the complicated latent proximity. DANE [57] employs GCN [40] to get transferable node embeddings on different networks. However, the above methods generate the samples from the original graph, and it cannot learn the unseen information of the graph and is difficult to extend to the large-scale network.

6 CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a novel robust and generalized framework called AGE for adversarial graph embedding. Specifically, we design the generator(s) and the discriminator(s) that can preserve complex semantic information of the graph by using the continuous implicit distribution of nodes and the semantic information of the graph. The computational complexity of the proposed framework is linearly related to the number of edges in the graph, and can be generalized well to various graphs. We design three models for three typical graphs by simple modifications, demonstrating the flexibility and generalization of the proposed framework. The extensive experimental results on the real-world graph datasets demonstrate that our models consistently and significantly outperform the state-of-the-art methods in the link prediction, node classification, and graph reconstruction tasks.

In the future, we plan to explore the proposed methods for graphs with more types of semantics (e.g., attribute graphs). Another interesting direction is to fuse our framework with other graph embedding methods deeply for better graph representation capability.

ACKNOWLEDGEMENT

The corresponding author is Jianxin Li. This work is supported by the NSFC under grant No. U20B2053. Philip S. Yu is supported by NSF under grants III-1763325, III-1909323, and SaTC-1930941.

REFERENCES

[1] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *JASIST*, vol. 58, no. 7, pp. 1019–1031, 2007.
[2] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, "Verse: Versatile graph embeddings from similarity measures," in *WWW*, 2018, pp. 539–548.
[3] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*, 2011, pp. 115–148.
[4] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *CIKM*, 2015, pp. 891–900.
[5] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *AAAI*, 2017.
[6] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *KDD*, 2016, pp. 1105–1114.
[7] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.
[8] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.
[9] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[10] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *KDD*, 2017, pp. 135–144.
[11] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *KDD*, 2016, pp. 1225–1234.
[12] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE TNNLS*, 2020.
[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[14] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[15] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
[16] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *AAAI*, 2018, pp. 2508–2515.
[17] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," in *AAAI*, 2018, pp. 2167–2174.
[18] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *KDD*, 2018, pp. 2663–2671.
[19] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity," in *AAAI*, 2017, pp. 2942–2948.
[20] Y. Shi, H. Gui, Q. Zhu, L. Kaplan, and J. Han, "Aspem: Embedding learning by aspects in heterogeneous information networks," in *SIAM*, 2018, pp. 144–152.
[21] S. Zhu, J. Li, H. Peng, S. Wang, and L. He, "Adversarial directed graph embedding," in *AAAI*, 2021.
[22] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NeurIPS*, 2013, pp. 1–9.
[23] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, vol. 28, no. 1, 2014.
[24] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL*, 2015, pp. 687–696.
[25] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *KDD*, 2021, pp. 1150–1160.
[26] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *IEEE TKDE*, vol. 29, no. 1, pp. 17–37, 2017.
[27] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE TKDE*, vol. 29, no. 12, pp. 2724–2743, 2017.

[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[29] L. Šubelj and M. Bajec, "Model of complex networks based on citation dynamics," in *WWW*, 2013, pp. 527–530.

[30] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[31] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, vol. 29, no. 1, 2015.

[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *NeurIPS*, 2020.

[33] M. D. Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher, "How does the data sampling strategy impact the discovery of information diffusion in social media?" in *ICWSM*, 2010, pp. 34–41.

[34] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *ISWC*, 2003, pp. 351–368.

[35] G. Palla, I. J. Farkas, P. Pollner, I. Derényi, and T. Vicsek, "Directed network modules," *New J. Phys.*, vol. 9, no. 6, p. 186, 2007.

[36] T.-y. Fu, W.-C. Lee, and Z. Lei, "Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning," in *CIKM*, 2017, pp. 1797–1806.

[37] B. Hu, Y. Fang, and C. Shi, "Adversarial learning on heterogeneous information networks," in *KDD*, 2019, p. 120–129.

[38] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NeurIPS*, 2016.

[39] S. Pan, R. Hu, S. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," *IEEE TC*, vol. 50, no. 6, pp. 2475–2487, 2020.

[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[41] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," in *KDD*, 2015, pp. 1165–1174.

[42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NeurIPS*, 2013, pp. 3111–3119.

[43] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *WSDM*, 2018, pp. 459–467.

[44] Z. Tong, Y. Liang, C. Sun, X. Li, D. Rosenblum, and A. Lim, "Digraph inception convolutional networks," *NeurIPS*, vol. 33, pp. 17 907–17 918, 2020.

[45] Z. Tong, Y. Liang, H. Ding, Y. Dai, X. Li, and C. Wang, "Directed graph contrastive learning," in *NeurIPS*, vol. 34, 2021.

[46] M. Khosla, J. Leonhardt, W. Nejdl, and A. Anand, "Node representation learning for directed graphs," in *ECML-PKDD*, vol. 11906, 2019, pp. 395–411.

[47] J. Sun, B. Bandyopadhyay, A. Bashizade, J. Liang, P. Sadayappan, and S. Parthasarathy, "Atp: Directed graph embedding with asymmetric transitivity preservation," in *AAAI*, 2019, pp. 265–272.

[48] Y. Sun and J. Han, "Mining heterogeneous information networks: principles and methodologies," *Synthesis DMK*, vol. 3, no. 2, pp. 1–159, 2012.

[49] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE TKDE*, vol. 31, no. 2, pp. 357–370, 2018.

[50] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE TNNLS*, 2021.

[51] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.

[52] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *SIGKDD*, 2019, pp. 793–803.

[53] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *WWW*, 2020, pp. 2704–2710.

[54] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014, pp. 2672–2680.

[55] H. Gao, J. Pei, and H. Huang, "Progan: Network embedding via proximity generative adversarial network," in *KDD*, 2019, pp. 1308–1316.

[56] L. Sang, M. Xu, S. Qian, and X. Wu, "Aaane: Attention-based adversarial autoencoder for multi-scale network embedding," in *PAKDD*, 2019, pp. 3–14.

[57] Y. Zhang, G. Song, L. Du, S. Yang, and Y. Jin, "Dane: Domain adaptive network embedding," in *IJCAI*, 2019.

[58] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang, "Adversarial training methods for network embedding," in *WWW*, 2019, pp. 329–339.

Jianxin Li is currently a Professor with the Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His current research interests include machine learning, big data, and trustworthy computing. Dr. Li has published research papers in top-tier journals and conferences, including the IEEE TKDE, TDSC, KDD, NeurIPS, AAAI, and WWW.

Xingcheng Fu is currently a Ph.D. candidate at the Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His research interests include graph representation learning, complex network and social network analysis. He has published several papers on WWW, AAAI, ICDM, CIKM, etc.

Shijie Zhu is currently a M.S. candidate at the Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His research interests include graph representation learning, graph adversarial learning.

Hao Peng is currently an Associate Professor at the Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. His research interests include representation learning, data mining, and reinforcement learning. Dr. Peng has published papers in top-tier journals and conferences, including the IEEE TPAMI, TKDE, TNNLS, and WWW.

Senzhang Wang is currently a Professor with the School of Computer Science and Engineering, Central South University, Changsha. His current research interests include data mining, urban computing and social network analysis.

Qingyun Sun is currently an Assistant Professor at the Beijing Advanced Innovation Center for Big Data and Brain Computing in Beihang University. Her research interests include data mining and graph representation learning. Her research interests include machine learning and graph mining. She has published several papers on WWW, AAAI, ICDM, CIKM, etc.

Philip S. Yu is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. He is a Fellow of the ACM and IEEE. Dr. Yu was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011-2017) and IEEE Transactions on Knowledge and Data Engineering (2001-2004).

Lifang He is currently an Assistant Professor in the Department of Computer Science and Engineering at Lehigh University. Before her current position, Dr. He worked as a postdoctoral researcher in the Department of Biostatistics and Epidemiology at the University of Pennsylvania. Her current research interests include machine learning, data mining, tensor analysis, with major applications in biomedical data and neuroscience.