Neurocomputing 541 (2023) 126235

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Wasserstein Adversarially Regularized Graph Autoencoder

Huidong Liang *, Junbin Gao

The University of Sydney Business School, The University of Sydney, Camperdown, NSW 2006, Sydney, Australia

ARTICLE INFO

Article history:
Received 5 February 2022
Revised 6 April 2023
Accepted 22 April 2023
Available online 25 April 2023

2010 MSC:
00–01
99–00

Keywords:
Wasserstein distance
Graph neural networks
Variational autoencoder
Adversarial training

ABSTRACT

This paper introduces Wasserstein Adversarially Regularized Graph Autoencoder (WARGA), an implicit generative algorithm that directly regularizes the latent distribution of node embedding to a target distribution via the Wasserstein metric. To ensure the Lipschitz continuity, we propose two approaches: WARGA-WC that uses weight clipping method and WARGA-GP that uses gradient penalty method. The proposed models have been validated by link prediction and node clustering on real-world graphs with visualizations of node embeddings, in which WARGA generally outperforms other state-of-the-art models based on Kullback–Leibler (KL) divergence and typical adversarial framework.

1. Introduction

Graph, consisting of a set of nodes and links, is an essential type of data that captures the topological structure within observations. Typical tasks on graphs involve link prediction, node clustering, node classification, etc., which have received an increasing amount of attention in many fields [1] including social network analysis [2,3], recommendation system [4,5] and bioinformatics [6,7].

Recently, *node embedding*, an approach that provides low-dimensional vector-space representations for graph nodes, has become a paradigm in graph analysis [8]. The learned embedding preserves useful information from the original features and retains the topological structure of the graph in the meantime, such that the algorithm will be both effective and efficient when performing the aforementioned downstream tasks. Some embedding methods assume that only topological structure is given, such as Spectral Clustering [3], DeepWalk [9] and node2vec [10], while others not only consider the graph structure but also exploit the content features, such as Graph Convolutional Networks (GCN) [11], GraphSAGE [12] and Graph Attention Networks (GAT) [13].

Among the embedding models, some take the generative approach, which, instead of learning a fixed vector representation for each node, assume particular probability distributions for the latent representations. For example, Variational Graph Autoencoder (VGAE) [14] takes a standard Gaussian as the prior distribution, and then uses variational inference to estimate the posterior distribution of the latent embedding, with Kullback–Leibler (KL) divergence being the measurement of "distance" between distributions. Adversarially Regularized Graph Autoencoder (ARGA) [15], based on VGAE, further proposes an adversarial framework that mimics Generative Adversarial Nets (GAN) [16] as regularization, which introduces a discriminator to penalize the encoded distribution for deviating from the target distribution, leading to better experimental results. However, KL divergence is not a valid metric by strict definition, since it fails to satisfy symmetry and triangular inequality, and only works for distributions with common supports [17].

Meanwhile, Wasserstein distance, also known as Earth-Mover distance [18], has gained popularity in machine learning research for its effective measurement of the distance between distributions. Compared to the other commonly used similarity measures such as KL divergence and Jensen-Shannon (JS) divergence, Wasserstein distance is suitable for estimating distributions with disjoint supports [17]. The seminal work Wasserstein Generative Adversarial Networks (WGAN) [19] first replaces the discriminator in GAN [16] by Wasserstein metric to tackle the problems of unstable training and mode collapsing. WGAN-GP [20] further proposes a gradient penalty method to replace the weight clipping approach

* Corresponding author.
E-mail addresses: hlia0714@uni.sydney.edu.au (H. Liang), junbin.gao@sydney.edu.au (J. Gao).

https://doi.org/10.1016/j.neucom.2023.126235
0925-2312/

for the Lipschitz constraint in WGAN, and achieves better training speed and picture sample quality. Moreover, Wasserstein Autoencoder (WAE) [21] outperforms Variational Autoencoder (VAE) [22] by enforcing a continuous encoded latent distribution to match the target distribution, as opposed to VAE that individually enforces each observation's latent distribution to match the target distribution.

In this work, we propose Wasserstein Adversarially Regularized Graph Autoencoder (WARGA), which directly regularizes the encoded latent distribution to a target distribution via 1-Wasserstein distance. Our main contributions are summarized below:

- We propose two models: WARGA-WC and WARGA-GP to regularize node embedding via Wasserstein distance, which use weight clipping method and gradient penalty method respectively to ensure Lipschitz continuity.
- Compared to other node embedding regularizers that use KL divergence and adversarial method, WARGA can handle distributions with little common support and provide a more natural explanation via proper distance measures than artificially designed discriminators.
- We validate the proposed methods on real-world citation networks by link prediction, node clustering and embedding visualization, where WARGA generally outperforms the other baselines based on KL divergence and adversarial framework.

The paper is organized as follows. Section 2 is dedicated to reviewing the KL divergence and adversarial regularization framework under graph settings and summarising the formulation of Wasserstein distance. Section 3 introduces our proposed methods based on the 1-Wasserstein distance in Graph Autoencoder architecture. In Section 4, comprehensive experiments on different types of learning tasks are conducted to demonstrate the performance and effectiveness of the proposed models.

2. Background and related work

In this section, we first review graph embedding approaches that use KL divergence and adversarial regularizer, then give an overview of Wasserstein distance with its famous applications in machine learning.

2.1. Variational Graph Autoencoder

Let $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ denote a graph consisting of a set of N nodes $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ with their features $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and an adjacency matrix $\mathbf{A}$ with $\mathbf{A}_{ij} = 1$ if there is a link between node $\mathbf{v}_i$ and $\mathbf{v}_j$, and $\mathbf{A}_{ij} = 0$ otherwise.

VGAE is optimized to perform link prediction task by maximizing the variational lower bound $\mathscr{L}$:

$$\mathscr{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{A},\mathbf{X})}[\log p(\mathbf{A}|\mathbf{X},\mathbf{z})] - \mathbf{KL}[q(\mathbf{z}|\mathbf{A},\mathbf{X}) \,\|\, p(\mathbf{z})], \tag{1}$$

in which $\mathbf{z}$ is the encoded latent embedding from GCN [11] with probability distribution $q(\mathbf{z}|\mathbf{A},\mathbf{X})$, $p(\mathbf{z})$ is a prior distribution of $\mathbf{z}$ (e.g. standard Gaussian), and $p(\mathbf{A}|\mathbf{X},\mathbf{z})$ is the likelihood of reconstructing $\mathbf{A}$ given $\mathbf{z}$ through an inner-product decoder.

We can view such formulation from a regularization perspective: maximizing the variational lower bound $\mathscr{L}$ is equivalent to minimizing the reconstruction loss and the KL divergence between q and p:

$$\begin{aligned}\max \mathscr{L} \iff \min \{ & -\mathbb{E}_{q(\mathbf{z}|\mathbf{A},\mathbf{X})}[\log p(\mathbf{A}|\mathbf{X},\mathbf{z})] \\ & + \mathbf{KL}[q(\mathbf{z}|\mathbf{A},\mathbf{X}) \,\|\, p(\mathbf{z})]\}.\end{aligned} \tag{2}$$

Therefore, the KL divergence can be treated as a term that penalizes the encoded distribution q for deviating from the specified prior distribution p.

2.2. Adversarially Regularized Graph Autoencoder

ARGA further introduces an adversarial model $\mathscr{D}$ that discriminates the samples of encoded latent distribution q_z by generator $\mathscr{G}$ from the samples of specified prior p_z. Together with the generator $\mathscr{G}$, ARGA is optimized as a *minimax* problem, where the generator wishes to generate embeddings that baffle the discriminator, while the latter tries to discern the "fake" embeddings. The adversarial objective is defined as:

$$\begin{aligned}\min_{\mathscr{G}} \max_{\mathscr{D}} \quad & \{\mathbb{E}_{\mathbf{z}\sim q_z}[\log(1 - \mathscr{D}(\mathscr{G}(\mathbf{A},\mathbf{X})))]\} \\ & + \mathbb{E}_{\mathbf{z}\sim p_z}[\log \mathscr{D}(\mathbf{z})].\end{aligned} \tag{3}$$

Finally, the generator $\mathscr{G}$ will be iteratively updated by both adversarial objective and variational lower bound $\mathscr{L}$ in Eq. (1) to encode the original feature into a regularized latent embedding.

2.3. Wasserstein distance and its dual form

The 1-Wasserstein distance [23] between two given distributions $\mathbb{P}_r$ and $\mathbb{P}_g$ is defined as:

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \mathscr{P}(r\sim\mathbb{P}_r, z\sim\mathbb{P}_g)} \mathbb{E}[\|r - z\|_2], \tag{4}$$

where $\mathscr{P}(r \sim \mathbb{P}_r, z \sim \mathbb{P}_g)$ is the set of all joint distributions $\gamma(r, g)$ with marginals $\mathbb{P}_r$ and $\mathbb{P}_g$ respectively. As the above form is intractable, the expression can be reformulated by Kantorovich-Rubinstein duality as:

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leqslant 1} \{\mathbb{E}_{r\sim\mathbb{P}_r}[f(r)] - \mathbb{E}_{z\sim\mathbb{P}_g}[f(z)]\}, \tag{5}$$

in which f is any continuous function that satisfies 1-Lipschitz continuity. Typically f is parameterized as a neural network function. To make sure the network is 1-Lipschitz continuous, WGAN [19] introduces a weight clipping approach that clamps the parameters of f into a compact space $\Phi = [-c, c]$ after each iteration during optimization. Nevertheless, WGAN-GP [20] argues that such space is only a subspace of all the functions that satisfies Lipschitz continuity, and hence making the algorithm sensitive to the choice of the hyper-parameter c. The authors then propose a gradient penalty method that directly restricts the gradient norm of f w.r.t. its input in the loss function, which is a soft version of the Lipschitz constraint.

In the following sections, we will see how Wasserstein distance improves the regularization under graph settings with the above two methods for Lipschitz continuity.

3. Proposed method

The general structure of the proposed method is illustrated in Fig. 1. We first use a generator $\mathscr{G}_w$ with parameters w to encode the original graph nodes features into low-dimensional vector representations, then force the encoded embedding's distribution to match a target distribution by minimizing their 1-Wasserstein distance via a regularizer f_ϕ. Finally, we reconstruct the adjacency matrix $\hat{\mathbf{A}}$ and iteratively update the generator $\mathscr{G}_W$ and the regularizer f_ϕ until the algorithm converges.

3.1. Graph autoencoder

We use GCN as the generator $\mathscr{G}_w(\mathbf{A},\mathbf{X})$ to encode the original node features $\mathbf{X} \in \mathbb{R}^{N\times p}$ with the graph's topological structure $\mathbf{A}$

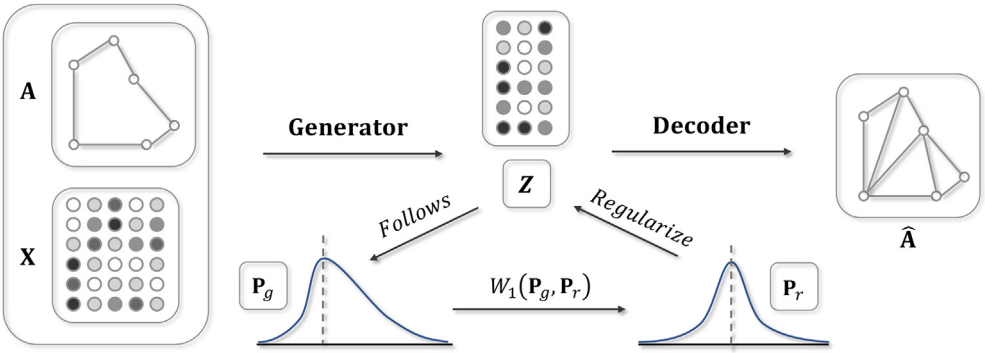

Fig. 1. Overall structure of WARGA.

into a low dimensional representation $\mathbf{Z} \in \mathbb{R}^{N\times d}$, which contains latent embeddings $\mathbf{z}_i$ for each node $\mathbf{v}_i \in \mathbf{V}$ as row-vectors. Each GCN layer can be expressed as:

$$\mathbf{Z}^{(\ell)} = \mathscr{G}_{W^{(\ell)}}\left(\mathbf{A}, \mathbf{Z}^{(\ell-1)}\right) = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(\ell-1)}\mathbf{W}^{(\ell)}\right), \tag{6}$$

where $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is the weighted adjacency matrix for graph convolution, with $\tilde{\mathbf{D}}_{ii} := \sum_j \tilde{\mathbf{A}}_{ij}$ to be the degree matrix of $\tilde{\mathbf{A}} := \mathbf{A} + \mathbf{I}$. $\sigma(\cdot)$ is an activation function for the current layer such as $\mathrm{ReLU}(t) = \max(0, t)$. $\mathbf{W}^{(\ell)}$ is the weight matrix (or parameter matrix) for layer ℓ, and $\mathbf{Z}^{(\ell)}$ is the output matrix of layer ℓ after activation ($\mathbf{Z}^{(0)} = \mathbf{X}$).

To reconstruct the adjacency matrix, we adopt an inner-product decoder to form the similarity matrix:

$$p(\mathbf{A}|\mathbf{X}, \mathbf{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\mathbf{A}_{ij}\,|\,\mathbf{z}_i, \mathbf{z}_j) \quad \text{with} p(\mathbf{A}_{ij} = 1\,|\,\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j), \tag{7}$$

where *sigmoid* function $\sigma(t) = 1/(1+\exp(-t))$ is used to restrain the output into range (0,1). As such, the objective of the generator $\mathscr{G}_w$ can be defined as minimizing the reconstruction loss below:

$$\min_w \left\{-\mathbb{E}_{\mathbb{P}_g}[\log p(\mathbf{A}|\mathbf{X}, \mathbf{Z})]\right\}. \tag{8}$$

In our work, instead of expressing the latent distribution explicitly, we take an implicit approach that assumes the latent representation $\mathbf{z}$ follows a target distribution $\mathbb{P}_r$ (e.g. the standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$), and denote the distribution of the embedding generated by $\mathscr{G}_w$ as $\mathbb{P}_g(\mathbf{z}|\mathbf{A}, \mathbf{X})$. The following sections will elaborate on how we use the proposed Wasserstein regularizer to enforce $\mathbb{P}_g$ on $\mathbb{P}_r$.

3.2. Wasserstein regularizer

To match the encoded distribution $\mathbb{P}_g(\mathbf{z}|\mathbf{A}, \mathbf{X})$ with the target distribution $\mathbb{P}_r$, we introduce a Wasserstein regularizer that minimizes the 1-Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_g$ formulated by Kantorovich-Rubinstein dual in Eq (5). To satisfy the 1-Lipschitz continuity, we apply two approaches: weight clipping and gradient penalty.

3.2.1. Weight clipping approach

The weight clipping method clamps the parameters ϕ of $\mathscr{F}$ into a compact space Φ with the following expression:

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \max_{\phi\in\Phi}\left\{\mathbb{E}_{\mathbf{r}\sim\mathbb{P}_r}\left[\mathscr{F}_\phi(\mathbf{r})\right] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_g}\left[\mathscr{F}_\phi(\mathbf{z})\right]\right\}, \tag{9}$$

where $\mathscr{F}$ (f in Eq (5)) is parameterised by a Multilayer Perceptron (MLP) with parameters $\phi \in \Phi$ in a compact space (i.e., $\Phi = [-c, c]$). Each layer in $\mathscr{F}_\phi$ can be expressed as:

$$\mathbf{H}^{(\ell)} = \mathscr{F}_{\phi^{(\ell)}}(\mathbf{H}^{(\ell-1)}) = \sigma(\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)} + \mathbf{b}^{(\ell)}), \tag{10}$$

in which $\mathbf{W}^{(\ell)}$ is the weight (parameter) matrix with bias $\mathbf{b}^{(\ell)}$, $\sigma(\cdot)$ is the activation function, and $\mathbf{H}^{(\ell)}$ is the output matrix for layer ℓ ($\mathbf{H}^{(0)}$ is the input matrix of $\mathbf{z}$ or $\mathbf{r}$). In practice, $\mathbf{z}$ is generated from the implicit latent distribution $\mathbb{P}_g$ by the generator $\mathscr{G}_w$, and $\mathbf{r}$ is generated from the prior distribution $\mathbb{P}_r$.

The generator $\mathscr{G}_w$, on the contrary, wants to minimize such distance w.r.t. w, which leads to an adversarial-like framework with a *minimax* objective:

$$\min_w \max_{\phi\in\Phi}\left\{\mathbb{E}_{\mathbf{r}\sim\mathbb{P}_r}\left[\mathscr{F}_\phi(\mathbf{r})\right] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_g}\left[\mathscr{F}_\phi(\mathbf{z})\right]\right\}. \tag{11}$$

Combining with the objective of reconstructing $\hat{\mathbf{A}}$ for generator $\mathscr{G}_w$ in Section 3.1 (which is independent of $\mathscr{F}_\phi$), we arrive at the final loss function for training WARGA-WC:

$$\min_w \max_{\phi\in\Phi}\left\{-\mathbb{E}_{\mathbb{P}_g}[\log p(\mathbf{A}|\mathbf{X}, \mathbf{Z})] + \mathbb{E}_{\mathbf{r}\sim\mathbb{P}_r}\left[\mathscr{F}_\phi(\mathbf{r})\right] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_g}\left[\mathscr{F}_\phi(\mathbf{z})\right]\right\}. \tag{12}$$

The algorithm for learning the generator $\mathscr{G}_w$ and the regularizer $\mathscr{F}_\phi$ with weight clipping method is demonstrated in Algorithm 1.

Algorithm 1: WARGA with Weight Clipping

Require: Graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$; feature matrix $\mathbf{X}$; Number of epochs T; Compact space for weight clipping Φ;Number of iterations for updating Wasserstein regularizer K

for epoch $= 1, 2, \ldots, T$ **do**
  Encode $\mathbf{A}$ and $\mathbf{X}$ into low-dimensional
  representation $\mathbf{Z}$ by $\mathscr{G}_w$
  for iteration $= 1, 2, \ldots, K$ **do**
    Sample $\{\mathbf{r}^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch of priors
    Sample $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim \mathbb{P}_g$ a batch from the 7: encoded embedding
    Update $\mathscr{F}_\phi$ by computing: 9: $\nabla_\phi \frac{1}{m}\sum_{i=1}^m \left(\mathscr{F}_\phi(\mathbf{r}^{(i)}) - \mathscr{F}_\phi(\mathbf{z}^{(i)})\right)$
    Clip ϕ back to $\Phi = [-c, c]$
  end for
  Update $\mathscr{G}_w$ by computing ∇_w in Eq. (12)
end for
return $\mathbf{Z}, \mathscr{G}_\mathbf{w}$ and $\mathscr{F}_\phi$

3.2.2. Gradient penalty approach

If $\mathscr{F}_\phi$ is differentiable and satisfies the 1-Lipschitz constraint, then the norm of $\mathscr{F}_\phi$ w.r.t. its input should not exceed 1. Thus, we can append a penalty term controlled by a weighting hyper-parameter λ to the objective function, which enforces the norm of $\mathscr{F}_\phi$'s gradient on 1. This will lead to a relaxed version of 1-Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_g$:

$$\max_{\phi} \quad \left\{\mathbb{E}_{\mathbf{r}\sim\mathbb{P}_r}\left[\mathscr{F}_\phi(\mathbf{r})\right] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_g}\left[\mathscr{F}_\phi(\mathbf{z})\right] + \lambda\,\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(\|\nabla_{A\hat{\mathbf{x}}}\mathscr{F}_\phi(\hat{\mathbf{x}})\|_2 - 1\right)^2\right]\right\}, \tag{13}$$

where $\hat{\mathbf{x}} = \epsilon\mathbf{r} + (1-\epsilon)\mathbf{z}$ with $\epsilon \sim U[0, 1]$ indicates a random input. Despite this penalty term forces the norm of $\mathscr{F}_w$'s gradient to approach 1 from both directions instead of just staying below 1 from only one direction, WGAN-GP [20] empirically found the two-direction method performs slightly better than the one-direction method, and hence is adopted in this work.

Together with the generator $\mathscr{G}_w$, the final objective for WARGA-GP becomes:

$$\min_{w}\max_{\phi} \quad \left\{-\mathbb{E}_{\mathbb{P}_g}[\log p(\mathbf{A}|\mathbf{X},\mathbf{Z})] + \mathbb{E}_{\mathbf{r}\sim\mathbb{P}_r}\left[\mathscr{F}_\phi(\mathbf{r})\right] - \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_g}\left[\mathscr{F}_\phi(\mathbf{z})\right] + \lambda\,\mathbb{E}_{\hat{\mathbf{x}}\sim\mathbb{P}_{\hat{\mathbf{x}}}}\left[\left(\|\nabla_{\hat{\mathbf{x}}}\mathscr{F}_\phi(\hat{\mathbf{x}})\|_2 - 1\right)^2\right]\right\}. \tag{14}$$

The algorithm for training WARGA with the gradient penalty method is demonstrated in Algorithm 2.

Algorithm 2: WARGA with Gradient Penalty

Require: Graph $\mathbf{G} = (\mathbf{V}, \mathbf{A})$; feature matrix $\mathbf{X}$; Number of epochs T; Hyper-parameter for the gradient penalty λ; Number of iterations for updating Wasserstein regularizer K

for epoch $= 1, 2, \ldots, T$ **do**
  Encode $\mathbf{A}$ and $\mathbf{X}$ into low-dimensional
  representation $\mathbf{Z}$ by $\mathscr{G}_w$
  for iteration $= 1, 2, \ldots, K$ **do**
    Sample $\{\mathbf{r}^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch of priors
    Sample $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim \mathbb{P}_g$ a batch from the
    encoded embedding
    Sample $\epsilon \sim U[0, 1]$ a random number
    Calculate $\hat{\mathbf{x}}^{(i)} = \epsilon\mathbf{r}^{(i)} + (1-\epsilon)\mathbf{z}^{(i)}$
    Update $\mathscr{F}_\phi$ by computing:
    $\nabla_\phi \frac{1}{m}\sum_{i=1}^m \left(\mathscr{F}_\phi(\mathbf{r}^{(i)}) - \mathscr{F}_\phi(\mathbf{z}^{(i)})\right.$
    $\left. + \lambda\left(\|\nabla_{\hat{\mathbf{x}}^{(i)}}\mathscr{F}_\phi(\hat{\mathbf{x}}^{(i)})\|_2 - 1\right)^2\right)$
  end for
  Update $\mathscr{G}_w$ by computing ∇_w in Eq. (14)
end for
return $\mathbf{Z}, \mathscr{G}_\mathbf{w}$ and $\mathscr{F}_\phi$

3.3. Discussion

Compared to WGAN and WGAN-GP, instead of minimizing the Wasserstein distance between distributions of original (real) data and generated (fake) data in the Euclidean space (e.g. pictures), our WARGA utilizes Wasserstein distance from a regularization perspective on the latent embeddings and operate under graph settings with a topological structure among observations (e.g. citation networks). In this scenario, the mode collapse question (e.g., in image generation) is less problematic, since the target distribution in our choice, instead of being a high-dimensional image with intractable distribution, is just a low-dimensional standard Gaussian distribution, and we will elaborate on this in Section 4.5 with visualizations of empirical results. At the same time, the proposed Wasserstein regularizer also generalizes to other graph embedding generators such as GAT [13] under the Autoencoder architecture [24], and applies to other prior choices such as von Mises-Fisher distribution from $\mathscr{S}$-VAE [25] and even to priors with no closed-form density function as long as we can sample from the distribution. In addition, there are some other alternative methods to the weight clipping and gradient penalty methods, such as spectral normalization [26] and sorting [27–29], which we leave for future investigation.

4. Experiments

4.1. Experimental setup

We validate our proposed methods by link prediction, node clustering and embedding visualization on three popular citation networks: Cora, Citeseer and PubMed [30], with their statistics summarized in Table 1. For all tasks, we compare our algorithms against GAE, VGAE, ARGA and ARVGA. The characteristics of the baselines are shown in Table 2. In link prediction, we report AUC score (the area under the curve) and AP score (average precision), while for node clustering, we choose Acc (accuracy), NMI (normalized mutual information) and ARI (adjusted rand index) as metrics to compare our results to other baselines. Similar to the experimental design in VGAE [14], we use 5% links for validation, 10% links for testing, and the rest for training. All experiments are repeated 10 times by different random seeds, with results reported by mean (in percentage) and standard deviation (in decimal). The code for replicating the experiment results can be retrieved from https://github.com/LeonResearch/WARGA.

4.2. Link prediction

For a fair comparison purpose, we build our encoder identical to other baselines with 32 neurons in the first hidden layer and 16 neurons in the second embedding layer. The Wasserstein regularizer is also constructed similar to the discriminator in ARGA with 2 hidden layers (16-neuron and 64-neuron respectively). For Citeseer and Cora, we train our proposed model for 200 epochs and 400 epochs respectively via Adam optimizer [31] with Xavier [32] initialization, and choose a learning rate of 0.001 for both encoder and Wasserstein regularizer. While for the PubMed dataset, as it is relatively large (around 20k nodes with 44k links) compared to the other graphs, we iterate 1500 epochs for sufficient optimization with a learning rate of 0.005. In WARGA-WC, the hyper-parameter c for weight clipping is set to 0.01 (i.e. clamp the parameters into [-0.01, 0.01] after each iteration); and in WARGA-GP, the

Table 1
Dataset statistics.

Dataset	# Nodes	# Links	# Features	# Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

Table 2
Characteristics of baselines.

Characteristics	GAE	VGAE	ARGA	ARVGA
KL Divergence		✓		✓
Adversarial Method			✓	✓

Table 3
Link prediction results (mean % ± std).

Method	Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP
SC	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE	91.0 ± 0.02	92.6 ± 0.01	89.5 ± 0.04	89.9 ± 0.05	96.4 ± 0.00	96.5 ± 0.00
VGAE	91.4 ± 0.01	92.6 ± 0.01	90.8 ± 0.02	92.0 ± 0.02	94.4 ± 0.02	94.7 ± 0.02
ARGA	92.4 ± 0.003	93.2 ± 0.003	91.9 ± 0.003	93.0 ± 0.003	**96.8 ± 0.001**	**97.1 ± 0.001**
ARVGA	92.4 ± 0.004	92.6 ± 0.004	92.4 ± 0.003	93.0 ± 0.003	96.5 ± 0.001	96.8 ± 0.001
WARGA-WC	93.0 ± 0.004	**94.3 ± 0.003**	**92.9 ± 0.004**	**93.6 ± 0.004**	96.5 ± 0.001	97.0 ± 0.001
WARGA-GP	**93.1 ± 0.003**	**94.3 ± 0.003**	**92.9 ± 0.004**	**93.6 ± 0.004**	96.4 ± 0.002	96.9 ± 0.002

hyper-parameter λ for gradient penalty is set to 10 for all experiments.

The performances of WARGA and the other baselines are summarized in Table 3. The results suggest that by incorporating a Wasserstein regularizer, WARGA outperforms all four baselines on Cora and Citeseer with an increase in AUC score and AP score by over 0.5% compared to the leading baselines under the same model size. While on PubMed dataset, although ARGA achieves

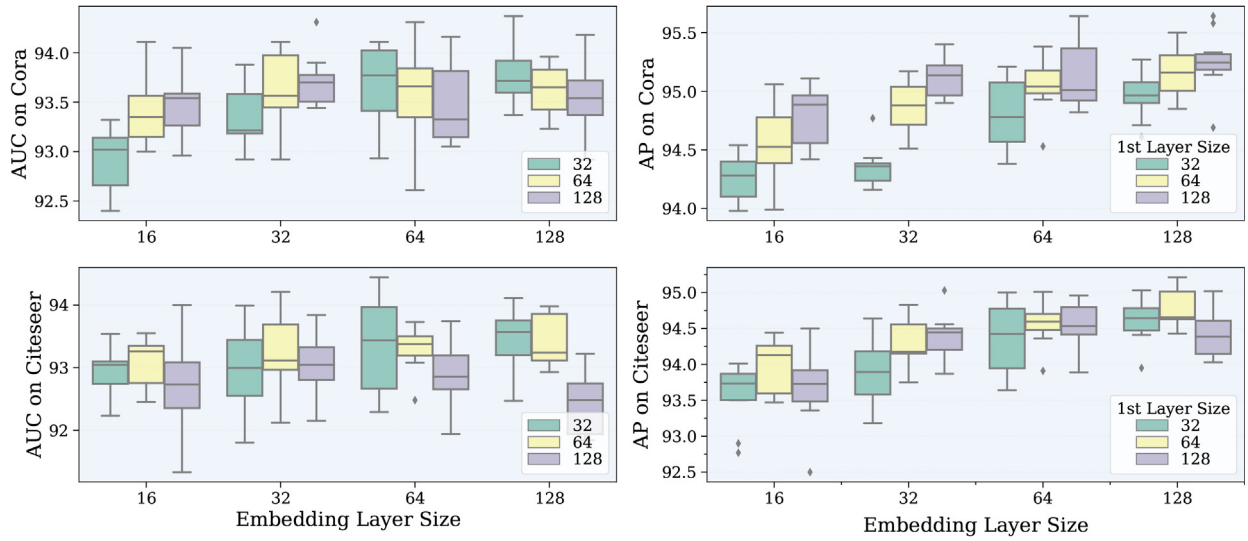

Fig. 2. Performance comparison on Cora and Citeseer under different layer sizes.

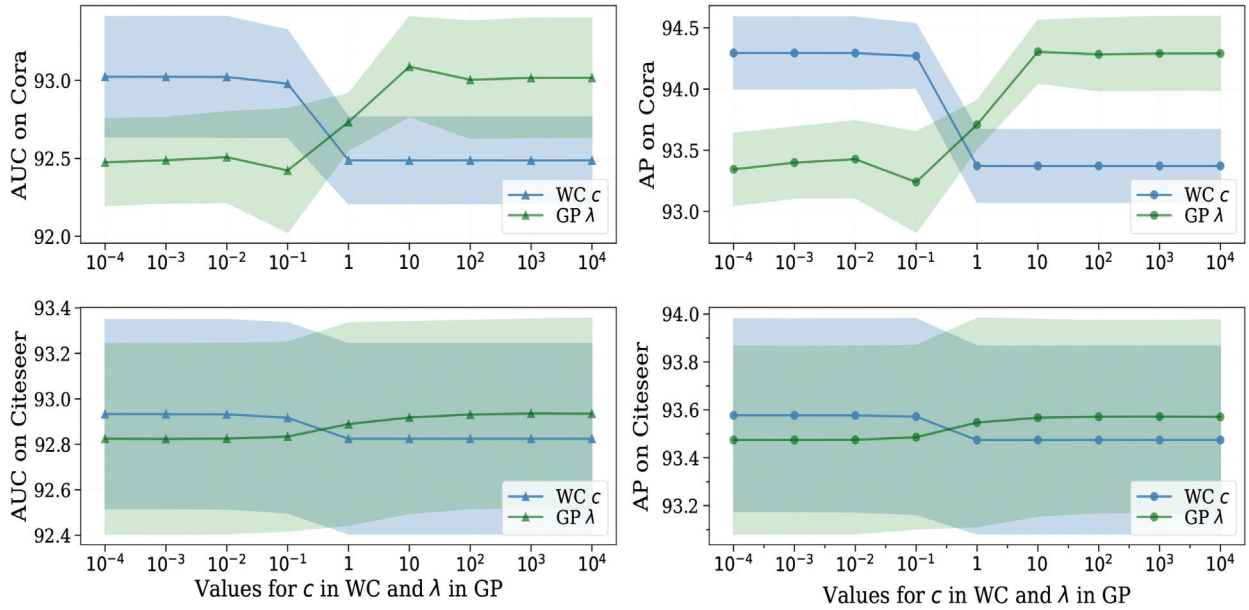

Fig. 3. Performance comparison on Cora and Citeseer given different weight-clipping and gradient penalty hyperparameters.

the best performance with around 97% in both AUC score and AP score, both WARGA-WC and WARGA-GP still generate very competitive results compared to ARGA (only 0.3% and 0.1% lower under AUC and AP respectively). The results also imply that there is no apparent difference in performance between weight clipping approach and gradient penalty approach.

Table 4
Node clustering results on Cora.

Cora	Acc	NMI	ARI
GAE	55.6 ± 0.05	41.2 ± 0.03	33.2 ± 0.05
VGAE	58.6 ± 0.05	40.1 ± 0.03	34.2 ± 0.03
ARGA	59.3 ± 0.04	42.2 ± 0.03	31.6 ± 0.05
ARVGA	63.4 ± 0.01	45.3 ± 0.00	39.2 ± 0.02
WARGA-WC	**66.0± 0.03**	**49.0 ± 0.02**	**43.8 ± 0.02**
WARGA-GP	65.7 ± 0.03	48.9 ± 0.02	43.5 ± 0.02

Table 5
Node clustering results on Citeseer.

Citeseer	Acc	NMI	ARI
GAE	42.5 ± 0.05	19.9 ± 0.03	13.7 ± 0.06
VGAE	50.3 ± 0.02	23.6 ± 0.02	22.1 ± 0.02
ARGA	36.6 ± 0.08	28.4 ± 0.04	16.1 ± 0.08
ARVGA	51.5 ± 0.03	26.3 ± 0.01	22.7 ± 0.02
WARGA-WC	56.2 ± 0.03	30.1 ± 0.02	28.5 ± 0.02
WARGA-GP	**56.3 ± 0.03**	**30.2 ± 0.02**	**28.6 ± 0.03**

Table 6
Node clustering results on PubMed.

PubMed	Acc	NMI	ARI
GAE	63.7 ± 0.01	23.3 ± 0.01	22.7 ± 0.02
VGAE	**68.9 ± 0.01**	28.3 ± 0.01	**30.6 ± 0.01**
ARGA	68.0 ± 0.00	29.4 ± 0.02	29.3 ± 0.00
ARVGA	63.4 ± 0.00	23.1 ± 0.00	22.4 ± 0.00
WARGA-WC	67.4 ± 0.01	29.4 ± 0.02	28.5 ± 0.02
WARGA-GP	68.2 ± 0.02	**30.9 ± 0.03**	30.1 ± 0.03

4.3. Hyper-parameter analysis

We further conduct a hyper-parameter analysis to explore the changes in WARGA's performance when given different encoding layers and demonstrate our findings with WARGA-WC on both Cora and Citeseer datasets. Specifically, we explore [32] neurons for the first layer and [16,32] neurons for the second (embedding) layer, and then run each combination 10 times with different random seeds. The results are illustrated in Fig. 2, which reveal that adding more neurons to the second embedding layer when given a 32-neuron first encoding layer will lead to conspicuously better performance, but such benefit is diminishing as the number of neurons in the first encoding layer increases to 64 and 128. On the other hand, models with a 64-neuron first layer generally outperform models with other first-layer choices under both AUC and AP, given any number of neurons in the embedding layer.

In addition, we test the sensitivity of our model's performance to the choice of hyperparameters for Lipschitz continuity, ranging from 10^{-4} to 10^{4} by a factor of 10 for both WARGA-WC (c) and WARGA-GP (λ), as shown in Fig. 3. We can see that on both datasets, the performance remains unchanged when $c \leqslant 10^{-1}$ for WARGA-WC and when $\lambda \geqslant 10$ for WARGA-GP, and their top performances are extremely similar, indicating that our models are relatively insensitive to the choice of these hyperparameters after certain thresholds.

4.4. Node clustering

In this section, we perform node clustering using K-means algorithm [33] based on the embeddings learned from the link prediction task above to cluster similar nodes into the same groups. We retain the experiment results for the four baselines from Mrabah et al. [34] together with ours, and summarize them in Table 4, Table 5 and Table 6. Similar to link prediction results, our proposed method outperforms all baselines on Cora and Citeseer datasets in

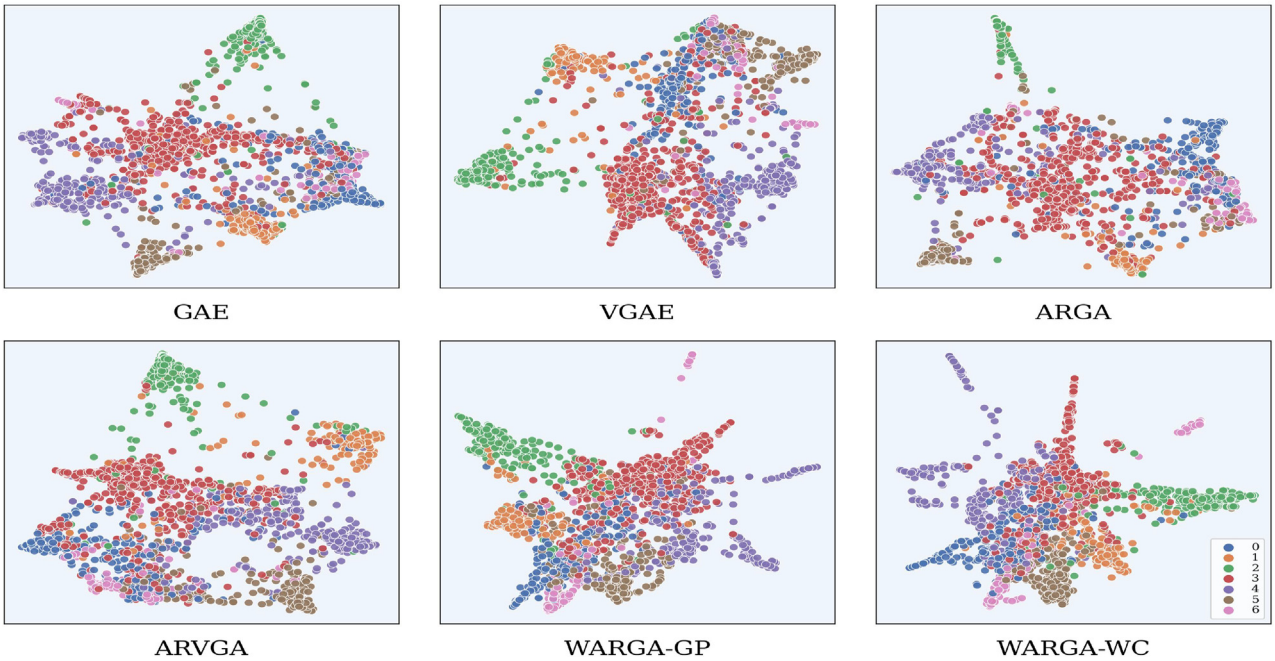

Fig. 4. Embeddings' t-SNE visualization on Cora.

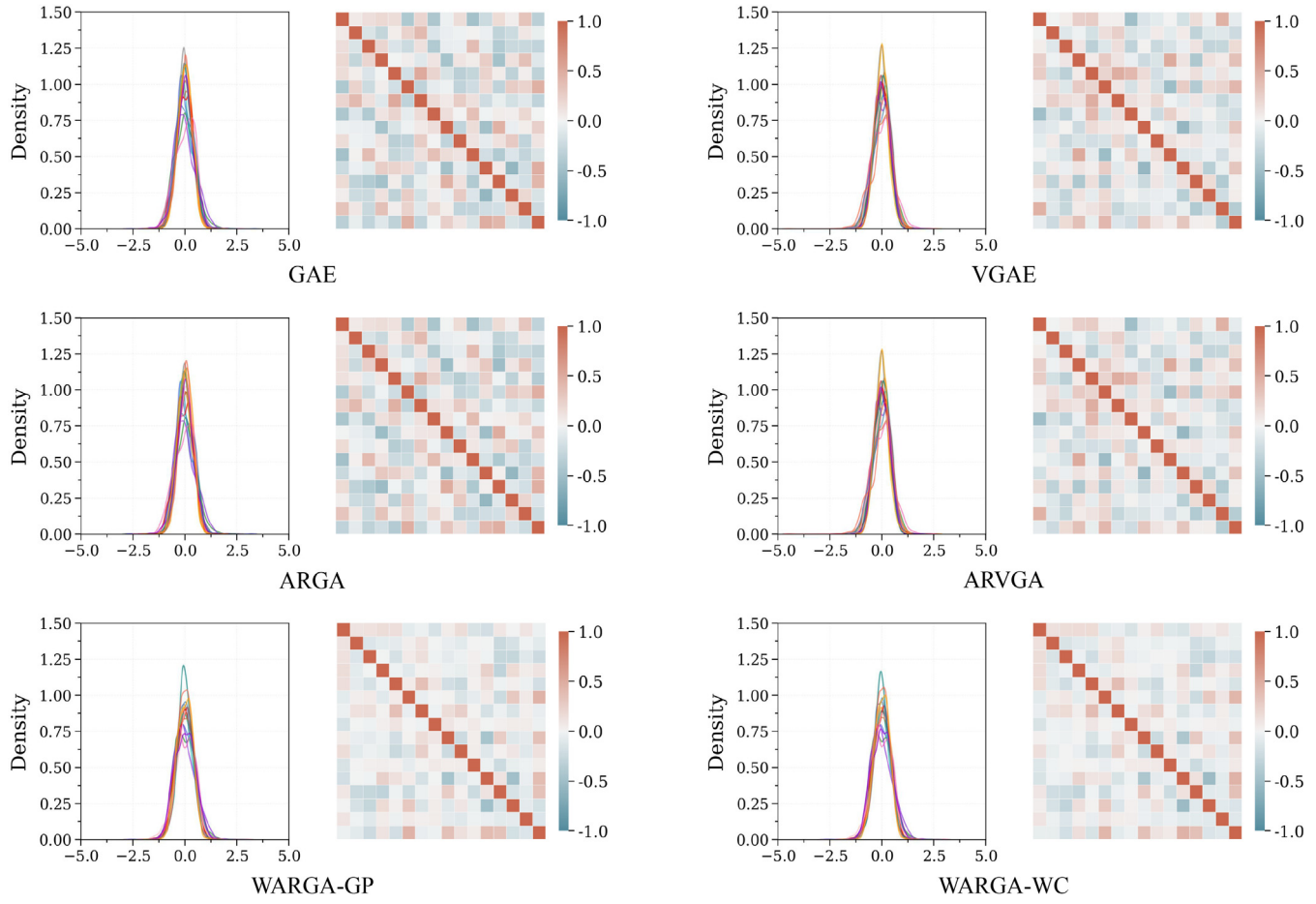

Fig. 5. Embedding's KDE visualizations and correlation heatmaps on Cora.

every metric by a decent margin of around 3% to 5%. For the PubMed dataset, however, VGAE shows the best results under Acc and ARI metrics of 68.9 and 30.6 respectively, while our WARGA-GP achieves the best NMI score of 30.9, with slightly lower Acc and ARI scores of 68.2 and 30.1 compared to the best baseline.

4.5. Embedding visualization

To intuitively show the advantages of Wasserstein regularization, we provide node embeddings' visualizations generated from the link prediction task on Cora. This section consists of two parts: a 2-dimensional visualization with t-SNE algorithm [35] and a kernel density estimation (KDE) on the latent embeddings with their corresponding correlation heatmaps.

We first use the t-SNE algorithm to reduce the dimensionality of node embeddings into 2-D, and illustrate different labels (not used in training) with different colours, as shown in Fig. 4. We can see that WARGA-WC and WARGA-GP generally separate different classes well, especially on blue and pink labels compared to other baselines. Note that the embeddings are generated from link prediction task rather than node classification task, where the latent embeddings are learned with no information about node classes. Then, we visualize the embeddings' KDE and plot their correlation heat maps in Fig. 5, which shows that although embeddings from GAE/VGAE and ARGA/ARVGA are Gaussian-like marginally, WARGA is able to generate less-correlated latent features under comparable performances. It implies that by employing Wasserstein distance for regularization, WARGA exhibits superior ability in producing node embedding that captures more distinctive information than the other baselines based on KL divergence and standard adversarial methods.

5. Conclusion

In this work, we introduced Wasserstein Adversarially Regularized Graph Autoencoder, which enforces graph node's latent representation to follow a target distribution by measuring the Wasserstein distance between latent distributions. To ensure the Lipschitz continuity in the Kantorovich-Rubinstein dual, we proposed two approaches: WARGA-WC with weight clipping method and WARGA-GP with the gradient penalty method. Empirical results suggest that our methods generally outperform models based on KL divergence and typical adversarial regularizer in link prediction and node clustering, meanwhile, the embedding KDE visualization intuitively shows WARGA's superiority of regularizing node latent distributions over the other baselines.

Data availability

Data will be made available on request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.

[2] L. Backstrom, J. Leskovec, Supervised random walks: predicting and recommending links in social networks, in: Proceedings of the 4th ACM International Conference on Web Search and Data Mining, 2011, pp. 635–644.

[3] L. Tang, H. Liu, Leveraging social media networks for classification, Data Min. Knowl. Disc. 23 (3) (2011) 447–478.

[4] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[5] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, 2019, pp. 417–426.

[6] M. Agrawal, M. Zitnik, J. Leskovec, Large-scale analysis of disease pathways in the human interactome, in: Proceedings of the Pacific Symposium on Biocomputing, World Scientific, 2018, pp. 111–122.

[7] X. Lin, Z. Quan, Z.-J. Wang, T. Ma, X. Zeng, Kgnn: Knowledge graph neural network for drug-drug interaction prediction., in: IJCAI, Vol. 380, 2020, pp. 2739–2745.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (1) (2020) 4–24.

[9] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.

[10] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.

[11] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[12] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.

[14] T.N. Kipf, M. Welling, Variational graph auto-encoders, arXiv:1611.07308 (2016).

[15] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: International Joint Conference on Artificial Intelligence, 2018.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inform. Process. Syst. 27 (2014).

[17] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, in: International Conference on Learning Representations, 2017.

[18] Y. Rubner, C. Tomasi, L.J. Guibas, A metric for distributions with applications to image databases, in: International Conference on Computer Vision, IEEE, 1998, pp. 59–66.

[19] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 214–223.

[20] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein gans, in: International Conference on Neural Information Processing Systems, 2017, pp. 5769–5779.

[21] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf, Wasserstein auto-encoders, in: International Conference on Learning Representations, 2018.

[22] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: International Conference on Learning Representations, 2014.

[23] C. Villani, Topics in Optimal Transportation, American Mathematical Society, 2003.

[24] A. Salehi, H. Davulcu, Graph attention auto-encoders, in: IEEE International Conference on Tools with Artificial Intelligence, IEEE, 2020, pp. 989–996.

[25] T.R. Davidson, L. Falorsi, N. De Cao, T. Kipf, J.M. Tomczak, Hyperspherical variational auto-encoders, in: Conference on Uncertainty in Artificial Intelligence, Association For Uncertainty in Artificial Intelligence (AUAI), 2018, pp. 856–865.

[26] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: International Conference on Learning Representations, 2018.

[27] C. Anil, J. Lucas, R. Grosse, Sorting out lipschitz function approximation, in: International Conference on Machine Learning, PMLR, 2019, pp. 291–301.

[28] J.E. Cohen, T. Huster, R. Cohen, Universal lipschitz approximation in bounded depth neural networks, arXiv preprint arXiv:1904.04861 (2019).

[29] U. Tanielian, G. Biau, Approximating lipschitz continuous functions with groupsort neural networks, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 442–450.

[30] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (3) (2008), 93–93.

[31] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2014.

[32] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[33] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, Oakland, CA, USA, 1967, pp. 281–297.

[34] N. Mrabah, M. Bouguessa, M.F. Touati, R. Ksantini, Rethinking graph autoencoder models for attributed graph clustering arXiv:2107.08562 (2021).

[35] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (86) (2008) 2579–2605.

Huidong Liang graduated from The University of Sydney Business School in 2022, majoring in business analytics [Hons] and finance, and is currently pursuing a Master's degree in statistics at University of Oxford. His main research interests include graph neural networks and their applications in finance.

Junbin Gao graduated from Huazhong University of Science and Technology (HUST), China in 1982 with a BSc in Computational Mathematics and obtained his PhD from Dalian University of Technology, China in 1991. He is Professor of Big Data Analytics in the University of Sydney Business School at the University of Sydney and was a Professor in Computer Science in the School of Computing and Mathematics at Charles Sturt University, Australia. He was a lecturer, then a senior lecturer in Computer Science from 2001 to 2005 at the University of New England, Australia. From 1982 to 2001 he was an associate lecturer, lecturer, associate professor, and professor in Department of Mathematics at HUST. His main research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.