Knowledge-Based Systems 268 (2023) 110456

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Adversarially regularized graph attention networks for inductive learning on partially labeled graphs

Jiaren Xiao [a], Quanyu Dai [b], Xiaochen Xie [c,d], James Lam [a], Ka-Wai Kwok [a,*]

[a] Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China
[b] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[c] Department of Automation, Harbin Institute of Technology, Shenzhen, China
[d] Guangdong Key Laboratory of Intelligent Morphing Mechanisms and Adaptive Robotics, Shenzhen, China

ARTICLE INFO

Article history:
Received 3 May 2021
Received in revised form 3 March 2023
Accepted 7 March 2023
Available online 13 March 2023

Keywords:
Adversarial regularization
Graph-based semi-supervised learning
Graph neural networks
Attention mechanism
Inductive learning

ABSTRACT

The high cost of data labeling often results in node label shortage in real applications. To improve node classification accuracy, graph-based semi-supervised learning leverages the ample unlabeled nodes to train together with the scarce available labeled nodes. However, most existing methods require the information of all nodes, including those to be predicted, during model training, which is not practical for dynamic graphs with newly added nodes. To address this issue, an adversarially regularized graph attention model is proposed to classify newly added nodes in a partially labeled graph. An attention-based aggregator is designed to generate the representation of a node by aggregating information from its neighboring nodes, thus naturally generalizing to previously unseen nodes. In addition, adversarial training is employed to improve the model's robustness and generalization ability by enforcing node representations to match a prior distribution. Experiments on real-world datasets demonstrate the effectiveness of the proposed method in comparison with the state-of-the-art methods. The code is available at https://github.com/JiarenX/AGAIN.

1. Introduction

Graphs naturally represent the data with complicated relationships and rich information, as seen in social, biological, and citation networks. Since graph-structured data are usually sparse, nonlinear and high-dimensional, the analysis of graph-structured data is challenging. To tackle graph-analytic tasks, a common approach is graph embedding which aims at learning the low-dimensional node representation vectors [1]. The key idea is to encode meaningful information like node features and graph structure into node representations (i.e., embedding vectors). Based on graph embedding, node classification tasks can be performed using classical machine learning techniques like a linear support vector machine (SVM) classifier [2]. Node classification has many practical applications, such as predicting user types in e-commence networks [3], assigning topics to papers in citation networks [4], and classifying protein roles in biological networks [5]. However, in many scenarios, labels are only available for a small subset of nodes due to the high cost and technical difficulty of labeling by human. To lessen the requirement of large amounts of labeled training nodes, a recent surge of research interest can be seen in semi-supervised learning on graphs [4].

Graph-based semi-supervised learning leverages the ample unlabeled nodes to train together with very few labeled nodes, so that the node classification accuracy can be improved. In recent years, substantial research effort has been devoted to designing neural network models that directly operate on graphs, known as graph neural networks (GNNs) [4,6]. In addition, a few GNN models introduce attention mechanisms to graph embedding [7,8]. Attention mechanisms allow the graph embedding model to highlight neighbors that contain more task-relevant information, consequently increasing the model capacity. Moreover, attention mechanisms are likely to help the model to disregard the noisy portions of a graph, thus improving model robustness.

Most existing graph embedding approaches are inherently transductive, since they focus on generating representations for nodes in a fixed graph. These methods optimize the representation of each node based on random walk [9,10] or matrix factorization [11,12]. To predict nodes newly added to the graph (i.e., unseen nodes), the transductive methods need non-trivial modifications and additional training, consequently suffering from expensive computation [5]. Many real applications involve unseen nodes. For example, new members may join social networks in Twitter and Facebook. In addition, there are usually massive amounts of new publications added to citation networks

* Corresponding author.
E-mail address: kwokkw@hku.hk (K.-W. Kwok).

https://doi.org/10.1016/j.knosys.2023.110456

in databases like PubMed and arXiv. An inductive approach [5, 13], that enables node representations to be quickly generated for unseen nodes, is essential for such scenarios. Compared to transductive learning, the inductive learning problem is more challenging, since the model would have already been optimized on the existing nodes before the introduction of new nodes.

Furthermore, noise, perturbations or even attacks are commonly seen in graph-structured data. For instance, scientific papers may have spelling mistakes, missing words, or incorrect expressions; criminals tend to hide or fabricate their personal information in social networks; fraudsters often manipulate the online reviews of their products to attract customers on e-commerce platforms. The general learning objective of existing graph embedding methods is to well preserve graph structure only, or to jointly capture both structural properties and side information like node features. As a result, the noise in structure and features can lead to poor performance of these methods [14–16]. In semi-supervised learning, a common regularization is to drive connected nodes to have the same label based on the homophily assumption [13,17]. As shown in [18], the working mechanism of graph convolution [4] is a special form of Laplacian smoothing which mixes the features of a node and its neighbors. Therefore, the relational effect of graph structures [16] is likely to worsen the model performance, since manipulating one node or edge may misguide the predictions of relational nodes.

To improve the model robustness over noisy graphs, some pioneering research [19–21] employs adversarial training in graph embedding. These studies are largely inspired by recent generative adversarial models [22–24], which are shown to be effective in learning robust representations. Similar to the adversarial autoencoder (AAE) [25], the basic idea is to match the learned node representations with a prior distribution using adversarial training. The underlying motivation is to enforce an additional regularization on the node representations, and to introduce a certain amount of uncertainty in the learning process. This helps improve the model robustness against noisy graphs. Adversarial training also upholds the potential to avoid overfitting and achieve relatively promising generalization performance. However, to our knowledge, none of these prior studies focuses on robust graph embedding under the inductive semi-supervised setting.

In this paper, we propose a novel method named Adversarially regularized Graph Attention networks for INductive learning on partially labeled graphs (AGAIN). On one hand, our method encodes graph structure and node features into node embeddings with an attention-based aggregator. When aggregating the neighborhood information, an attention mechanism is adopted to assign different learnable weights to the sampled neighbors, capturing the importance of each neighbor. At the inference time, the learned aggregator can produce informative representations for previously unseen nodes. On the other hand, adversarial training is employed to learn robust node representations by enforcing the representations to match a prior distribution.

The proposed method is evaluated on four datasets including three citation networks (i.e., Cora, CiteSeer and PubMed) as well as one social network named BlogCatalog. The *main contributions* of this work are summarized as follows.

- The first adversarially regularized GNN model is proposed and designed specifically to address the challenging inductive learning problem on partially labeled graphs.
- Our model is devised to incorporate attention mechanism and adversarial training, effectively generating informative and robust node representations.
- Extensive experiments are conducted with real-world information networks, showing our model is comparable with or even superior to the state-of-the-art methods on the benchmark inductive node classification tasks.

This rest of this paper is organized as follows. The relevant literature is reviewed in Section 2. The proposed method is described in Section 3. The experimental results are reported in Section 4. Finally, the conclusions are summarized in Section 5.

2. Related work

2.1. Graph-based semi-supervised learning

On a partially labeled graph, graph-based semi-supervised learning aims to jointly utilize both the scarce labeled and ample unlabeled nodes to improve node classification accuracy. There exist two learning paradigms: transductive learning and inductive learning. Transductive learning [26,27] only aims at classifying the unlabeled nodes that are observed in training time. Inductive learning algorithms, such as manifold regularization [28] and semi-supervised embedding [29], can generalize to unobserved nodes. Planetoid [13] has both transductive and inductive variants. The inductive algorithm, Planetoid-I, learns a parameterized classifier based on node features to facilitate predictions on nodes unseen during training. Note that, graph-based semi-supervised learning assumes the training and test nodes share the same label space. In contrast, open-set learning [30] and out-of-distribution detection [31] have test data from the classes that are unseen in training data.

Graph embedding is a broader research topic that focuses on mapping the nodes to representation vectors in the low-dimensional space. There are a number of recent approaches that learn low-dimensional embeddings based on random walk (e.g., DeepWalk [9], LINE [10], and node2vec [32]) and matrix factorization (e.g., GraRep [11], HOPE [12], and M-NMF [33]). The learning objective of these methods is to maximally preserve the topological information. Under the assumption that node features are available, some approaches are capable of exploiting both the topological and feature information, such as TADW [34], TriDNR [35], and UPP-SNE [36]. However, these methods are transductive by training embeddings for individual nodes in a fixed graph, and not designed specifically for semi-supervised learning.

Beyond the classical graph embedding methods, increasing research interest can be seen in graph neural networks (GNNs) [6] which can be categorized as spectral and spatial approaches. Spectral-based approaches introduce filters for graph convolutions [4,37,38]. Among them, Kipf and Welling [4] simplified the previous spectral convolutions to be a localized first-order approximation for semi-supervised learning. This algorithm depends on the graph Laplacian and all node features during training, and hence lies within the transductive setting. Imitating the convolutional neural networks on images, the spatial approaches define graph convolution directly based on the spatial relations of a node and its neighborhood [5,8,39]. The well-known inductive method, GraphSAGE [5], computes the node embeddings by sampling a fixed-size neighborhood and then aggregating features. The feature aggregation is based on the elementwise mean of neighborhood (GS-mean), the inductive variant of GCN [4] (GS-GCN), the LSTM architecture (GS-LSTM), and elementwise max-pooling or mean-pooling operation (GS-pool). The performance of GraphSAGE in several large-scale benchmarks is quite impressive.

Furthermore, attention mechanisms have been widely adopted in computer vision [40] and natural language processing [41]. The goal is to attend over important parts of the data, and to improve the performance of a machine learning model. Attention mechanisms have also been introduced to designing GNN models. GAT [8] assigns learnable weights to the entire neighborhood nodes, yielding improved or matched performance in semi-supervised node classification. Although the reported single-graph

experiments are transductive, GAT is capable of supporting inductive learning on one graph. The reason is that GAT only requires access to the local neighborhood of a node, instead of the upfront knowledge about the whole graph. In addition, GAT adds a self-loop to a node and treats the node itself as one of its neighbors, so that the previous node representation can be inherently incorporated in the neighborhood aggregation process. Unlike the traditional multi-head attention, GaAN [42] controls the importance of each attention head with a convolutional subnetwork. HAN [43] proposes a two-level attention (i.e., node level and semantic level) for learning on heterogeneous graphs.

Attention mechanism is also explored in this work. However, different from GAT, we sample a fixed size of neighboring nodes before calculating attention coefficients, in order to keep the computational footprint consistent for every node. Additionally, we utilize a skip connection [44] to incorporate the node representation of the previous layer. As introduced in GraphSAGE [5], such skip connection operation has the potential to boost model performance. Moreover, the methods introduced above are mostly unregularized and ignore the data distribution of learned node representations, which may result in poor performance on sparse and noisy graphs in real applications. In this work, we utilize adversarial training to address this issue.

2.2. Graph adversarial attacks and defenses

Many studies on image [45–47] and text [48] have shown that neural networks are vulnerable to deliberate adversarial perturbations in the input. There are two dominant types of adversarial attacks [49,50], namely, poisoning attacks in which the model is trained after the attack, and evasion attacks targeting the test phase in which the learned model is assumed to be fixed. Recently, it is also found that the performance of graph embedding methods including GNNs would drop significantly under malicious manipulations in graph structure or node features [14–16]. Accordingly, some defense models are proposed to improve the robustness of GNNs [51,52]. An additional hinge loss is considered in [52] during the training process to achieve certified robustness under perturbations on the node features. Inherited from the principle of information bottleneck [53,54], GIB [51] learns minimal sufficient node representations that naturally defend against attacks. To evaluate the model robustness, GIB employs adversarial attacks generated using Nettack [16], and simple feature attacks which inject Gaussian noise into the feature vectors. In this work, similar feature attacks are also used for robustness evaluation, due to the generality of Gaussian noise injection.

2.3. Generative adversarial models

The deep generative model, i.e., generative adversarial networks (GANs) [55], builds a minimax adversarial game for two players: the generator and the discriminator. The discriminator is usually a multi-layer perceptron (MLP) which is trained to tell apart whether an input sample comes from the real data distribution or the generator. Simultaneously, the generator is trained to generate samples as close to the real samples as possible to fool the discriminator. Being inspired by GANs, Makhzani et al. [25] employed adversarial training to perform variational inference by matching the representations with a prior distribution. This adversarial autoencoder (AAE) achieves competitive performance in semi-supervised classification on images. Some other generative adversarial models are proposed to learn robust representations for images [22,24] and text [23].

Recently, the adversarial regularization has been applied to graph-structured data in several studies. The first one is ANE [19] which combines an inductive variant of DeepWalk and the adversarial training for learning robust node representations. ARGA [20] and ARVGA [20] further utilize the node features together with topological information in a similar adversarial learning scheme. NetRA [21] circumvents the need of a pre-defined fixed prior, and further employs Wasserstein GANs [56] to overcome the unstable problem during training. However, the inductive semi-supervised learning that this work focuses on is not considered in the prior art.

3. Proposed method

In this section, we first introduce the problem and main notations. Then we present an overview of the model architecture, followed by a detailed description of each component. Finally, the algorithm of our model is provided together with an analysis of the computational complexity.

3.1. Problem definition and notations

An information network can be expressed as an attributed graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$, where $\boldsymbol{V}$ is the set of nodes, $\boldsymbol{E}$ is the set of edges representing the relationships between nodes, and $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ is the feature matrix. N is the number of nodes and D is the feature dimension. $\boldsymbol{x}_v^\top$ is one row in the feature matrix $\boldsymbol{X}$ representing the feature vector of node $v \in \boldsymbol{V}$. The topological structure of unweighted graph $\mathcal{G}$ can be represented as an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ with each element a_{ij} set as 0 or 1, specifying whether an edge exists between two nodes. The degree of the i-th node v is the number of its connected edges, i.e., degree $(v) = \sum_j a_{ij}$. The average degree is further defined as $\langle k\rangle = 2|\boldsymbol{E}|/N$, indicating the density of an undirected graph.

As shown in Fig. 1, in this work, we investigate the classification of nodes that are newly added to a partially labeled attributed graph. The main notations used in this paper are summarized in Table 1. A set of nodes, $\boldsymbol{V}$, consists of labeled nodes $\boldsymbol{V}_{\mathrm{L}}$ and unlabeled nodes $\boldsymbol{V}_{\mathrm{U}}$. Some of the unlabeled nodes (i.e., $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{o}}$) are observed during training, and the rest (i.e., $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{n}}$) are unobserved. Unobserved nodes, $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{n}}$, are added to the original graph during test phase.

Graph embedding aims at mapping a node, $v \in \boldsymbol{V}$, to a low-dimensional embedding vector $\boldsymbol{u}_v$. $\boldsymbol{u}_v^\top$ is one row within representation matrix $\boldsymbol{U} \in \mathbb{R}^{N\times d}$, where d is the embedding dimension. As shown in Fig. 1, the attributed graph is partially labeled, that is, only a small percentage of nodes are with labels. To perform node classification on top of embeddings, the semi-supervised learning is defined as learning a classifier, $f : \boldsymbol{V} \longmapsto \boldsymbol{Y}$, using both labeled nodes (i.e., $\boldsymbol{V}_{\mathrm{L}}$) and observed unlabeled nodes (i.e., $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{o}}$). Label matrix, $\boldsymbol{Y} \in \mathbb{R}^{N\times C}$, contains binary element, Y_{vk}, indicating whether node v is associated with class k. The total number of classes in $\boldsymbol{Y}$ is C. There are two learning paradigms. The transductive learning only aims to predict the observed unlabeled nodes in the graph, that is, $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{o}}$. Inductive learning further seeks to generalize the classification model to nodes that are unseen in the graph during training, that is, $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{n}}$. This work focuses on the inductive semi-supervised learning.

As stated in Section 1, the robustness of a graph embedding model against noisy input is an important issue, since noise and perturbations are commonly seen in graph-structured data. Therefore, in this work, we assume the inputs to be noisy when evaluating robustness. For an attributed graph, these inputs are usually feature matrix and structural information such as adjacency matrix, PPMI matrix [57] and random walk. Being inspired by the feature attacks in GIB [51], we randomly select a percentage of nodes in the graph, and add independent Gaussian noise to each dimension of the node features. The Gaussian noise is

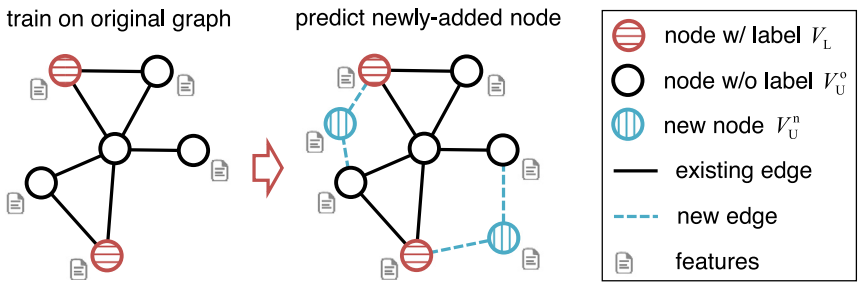

Fig. 1. Inductive learning under semi-supervised setting. The node classification model is trained on the original graph in which only a small percentage of nodes have labels. Then the learned model is directly applied to make predictions on nodes that are newly added and unseen during training.

Table 1
Main notations.

Notation	Description
$\mathcal{G}$	An attributed graph
$\boldsymbol{V}$, $\boldsymbol{E}$, $\boldsymbol{A}$	Node set, edge set, and binary adjacency matrix of $\mathcal{G}$
$\boldsymbol{x}_v$, $\boldsymbol{X}$	Feature vector of node $v \in \boldsymbol{V}$ and feature matrix of $\mathcal{G}$
$\boldsymbol{u}_v$, $\boldsymbol{U}$	Embedding vector of node $v \in \boldsymbol{V}$ and representation matrix of $\mathcal{G}$
$\boldsymbol{Y}$, $\hat{\boldsymbol{Y}}$	Label matrix and prediction score matrix of $\mathcal{G}$
N	Number of nodes in $\mathcal{G}$
n	Number of labeled nodes per class in $\mathcal{G}$
$\lvert\boldsymbol{E}\rvert$, $\langle k\rangle$	Number of edges and average degree in $\mathcal{G}$
D, d	Feature dimension and embedding dimension
C	Number of classes in $\boldsymbol{Y}$
$f_{\boldsymbol{\varphi}}(\cdot)$, $l_{\boldsymbol{\psi}}(\cdot)$, $d_{\boldsymbol{w}}(\cdot)$	GNN encoder, node classifier, and discriminator
$\boldsymbol{\varphi}$, $\boldsymbol{\psi}$, $\boldsymbol{w}$	Sets of parameters in $f_{\boldsymbol{\varphi}}(\cdot)$, $l_{\boldsymbol{\psi}}(\cdot)$ and $d_{\boldsymbol{w}}(\cdot)$
$n_{\max}$	Maximum training epoch
n_{D}	Number of discriminator training per generator iteration
K	Maximum search depth
$\boldsymbol{B}$	A batch of nodes
p_r	Discriminator learning rate
p_c	Weight decay coefficient
s	Neighborhood sample size
σ	Nonlinear activation function
AGG	Aggregator function
α	Attention coefficient
$\boldsymbol{h}$	Latent representation
η	Percentage of nodes with noise
λ	Feature noise ratio
Δ	Performance gap

injected during the test phase in which the model is fixed. As introduced in Section 2, this kind of noise injection belongs to the evasion attacks. As shown in [51], the resilience to feature attacks, or the lack of it, can be reflected by the consequent performance under feature noise.

3.2. Overview of model architecture

Fig. 2 shows the model architecture of the proposed method, i.e., AGAIN. There are two main components, i.e., inductive learning and adversarial training. Specifically, the Graph Attention networks for INductive learning (GAIN) consist of GNN encoder $f_{\boldsymbol{\varphi}}(\cdot)$ and node classifier $l_{\boldsymbol{\psi}}(\cdot)$. GNN encoder encodes the topological information and node features of an input graph into low-dimensional node embedding vectors with the attention-based aggregator. Node embeddings are further transformed by a node classifier, $l_{\boldsymbol{\psi}}(\cdot)$, which is a fully-connected layer followed by a softmax activation, into predictions of node labels. Moreover, the adversarial training imposes a prior distribution on the node embeddings. Discriminator, $d_{\boldsymbol{w}}(\cdot)$, aims at discriminating the prior samples and the embedding vectors. It is a standard multi-layer perceptron (i.e., MLP), in which the output is a single neuron followed by a sigmoid activation, indicating the probability of an input sample to be real. Note that GNN encoder $f_{\boldsymbol{\varphi}}(\cdot)$ also plays the role of generating fake samples (i.e., embedding vectors) in the adversarial training. Hence, the GNN encoder is shared by the inductive learning and adversarial training components. Three sets of parameters, $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$ and $\boldsymbol{w}$, are involved in $f_{\boldsymbol{\varphi}}(\cdot)$, $l_{\boldsymbol{\psi}}(\cdot)$ and $d_{\boldsymbol{w}}(\cdot)$, respectively.

3.3. Inductive learning

As illustrated in Fig. 3, in the neighborhood sampling stage, rather than considering the whole neighborhood of a given target node, a fixed size of neighbors are randomly sampled at each search depth. In case that sample size is larger than the node degree, neighbors are sampled with replacement. The sampling is an outward process in which the far neighborhood is gradually discovered. The maximum search depth is denoted as K. Then the nodes aggregate information from their sampled neighbors. Note that the aggregation is an inward process. As the process iterates, more and more information is gained from far neighborhood by the target node.

When aggregating neighborhood information, we introduce an attention mechanism [8] to assign different learnable weights to the neighbors, indicating their relative importance in assisting the learning of target node. As shown in Fig. 3, at step k, attention

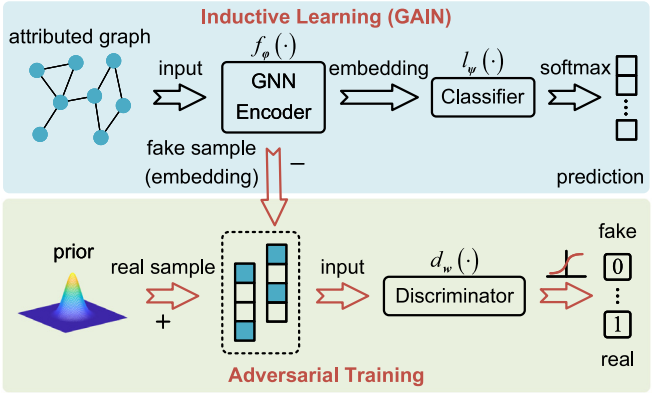

Fig. 2. Model architecture of AGAIN. The upper and lower tiers illustrate inductive learning and adversarial training, respectively. GNN encoder is empowered by the attention-based aggregator.

coefficient α_{vu}^k can be computed as follows.

$$\alpha_{vu}^k = \frac{\exp\left(\sigma_1\left(\left(\boldsymbol{a}^k\right)^\top [\boldsymbol{W}^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}^k \boldsymbol{h}_u^{k-1}]\right)\right)}{\sum_{m \in \boldsymbol{S}_v} \exp\left(\sigma_1\left(\left(\boldsymbol{a}^k\right)^\top [\boldsymbol{W}^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}^k \boldsymbol{h}_m^{k-1}]\right)\right)} \tag{1}$$

where $\boldsymbol{S}_v$ is the set of immediate neighbors of node v; $\boldsymbol{h}_v^{k-1} (v \in \boldsymbol{V})$ and $\boldsymbol{h}_u^{k-1} (u \in \boldsymbol{S}_v)$ are the latent representations of target node v and neighboring node u at the previous step (i.e., $k-1$), respectively; $\boldsymbol{a}^k$ and $\boldsymbol{W}^k$ are the weight vector and matrix for linear transformations, respectively. Note that, at step $k=0$, the latent representation is the node feature vector, that is, $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$. Therefore, the latent representations are initialized with node features and updated step by step. Here, nonlinear activation, σ_1, is a leaky ReLU function, i.e., $\sigma_1(x) = \max(0.2x, x)$.

The latent representation of neighborhood can then be derived as follows.

$$\boldsymbol{h}_S^k = \mathrm{AGG}_k(\boldsymbol{h}_u^{k-1} \mid u \in \boldsymbol{S}_v) = \sum_{u \in \boldsymbol{S}_v} \alpha_{vu}^k \boldsymbol{h}_u^{k-1} \tag{2}$$

in which AGG_k is the aggregator function at step k. Then the latent representation of node v at step k (i.e., $\boldsymbol{h}_v^k$) can be calculated.

$$\boldsymbol{h}_v^k = \sigma_2([\boldsymbol{W}_v^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}_S^k \boldsymbol{h}_S^k]), \tag{3}$$

$$\boldsymbol{h}_v^k = \boldsymbol{h}_v^k / \|\boldsymbol{h}_v^k\|_2, \tag{4}$$

where $\boldsymbol{W}_v^k$ and $\boldsymbol{W}_S^k$ are also weight matrices for linear transformations; nonlinear activation, $\sigma_2(x) = \max(0, x)$, is a ReLU function. Note that we implement a skip connection [44] in Eq. (3) to incorporate the node representation of previous layer. As introduced in GraphSAGE [5], such skip connection operation can potentially boost model performance.

The final representation output at step K is denoted as $\boldsymbol{u}_v$, which is the learned representation (i.e., embedding vector) of node v.

$$\boldsymbol{u}_v = \boldsymbol{h}_v^K = f_{\boldsymbol{\varphi}}(\boldsymbol{x}_v, \boldsymbol{x}_S), v \in \boldsymbol{V}, \tag{5}$$

in which $f_{\boldsymbol{\varphi}}$ is the GNN encoder, $\boldsymbol{x}_v$ is the feature vector of node v, and $\boldsymbol{x}_S$ is the feature matrix of the sampled neighboring nodes. For notational convenience, in the following descriptions, we simply use $f_{\boldsymbol{\varphi}}(\boldsymbol{x}_v)$ to denote $\boldsymbol{u}_v$. Note that embedding vector, $\boldsymbol{u}_v$, is also the fake sample in adversarial training indicated by a sign "−" in Fig. 2.

Finally, prediction score vector, $\hat{\boldsymbol{y}}_v$, can be calculated by feeding embedding vector $\boldsymbol{u}_v$ into node classifier $l_{\boldsymbol{\psi}}(\cdot)$.

$$\hat{\boldsymbol{y}}_v = l_{\boldsymbol{\psi}}(\boldsymbol{u}_v), v \in \boldsymbol{V}. \tag{6}$$

$\hat{\boldsymbol{y}}_v^\top$ is one row in prediction score matrix $\hat{\boldsymbol{Y}} \in \mathbb{R}^{N \times C}$. Under semi-supervised setting, the inductive learning component is trained by minimizing the cross-entropy loss of labeled nodes as follows.

$$\mathcal{L}_{\mathrm{GAIN}} = -\mathop{\mathbb{E}}_{v \in \boldsymbol{B}}\left[\sum_{k=1}^{C} Y_{vk} \log(\hat{Y}_{vk})\right] \tag{7}$$

where $\boldsymbol{B}$ is a sampled batch from the training nodes; binary element Y_{vk} within label matrix $\boldsymbol{Y}$ indicates whether a node $v \in \boldsymbol{B}$ belongs to class k; and $\hat{Y}_{vk}$ is the corresponding element in prediction score matrix $\hat{\boldsymbol{Y}}$.

3.4. Adversarial training

An adversarial training model is employed to regularize the embedding vectors. The learned embeddings can be enforced to match a certain prior distribution. It builds an adversarial training platform for two players, namely, generator $g_{\boldsymbol{\theta}}(\cdot)$ and discriminator $d_{\boldsymbol{w}}(\cdot)$, to play a minimax game. Specifically, generator, $g_{\boldsymbol{\theta}}(\cdot)$, represents a nonlinear transformation from the input graph to embedding vectors. In this work, GNN encoder, $f_{\boldsymbol{\varphi}}(\cdot)$, play the role of $g_{\boldsymbol{\theta}}(\cdot)$. A real sample, $\boldsymbol{z}$, is sampled from prior distribution $P_{\mathrm{g}}(\boldsymbol{z})$, while embedding vector $f_{\boldsymbol{\varphi}}(\boldsymbol{x})$ is treated as the fake sample. Discriminator, $d_{\boldsymbol{w}}(\cdot)$, is a standard multi-layer perceptron. The output of discriminator, which is of one dimension followed by a sigmoid activation, indicates the probability of an input sample to be real. The value function of adversarial training can be expressed as follows [55].

$$\min_{\boldsymbol{\varphi}} \max_{\boldsymbol{w}} \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_{\mathrm{g}}(\boldsymbol{z})}[\log d_{\boldsymbol{w}}(\boldsymbol{z})] + \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_{\mathrm{data}}(\boldsymbol{x})}[\log(1 - d_{\boldsymbol{w}}(f_{\boldsymbol{\varphi}}(\boldsymbol{x})))] \tag{8}$$

in which $P_{\mathrm{data}}(\boldsymbol{x})$ is the feature distribution of nodes.

During training, the discriminator is trained to distinguish prior samples from embedding vectors, while the generator aims to fit node embeddings to the prior distribution, thus misguiding the discriminator. We can separate the training of discriminator and generator. The loss function of discriminator is defined as

$$\mathcal{L}_{\mathrm{DIS}}(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{z}) = -\mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_{\mathrm{g}}(\boldsymbol{z})}[\log d_{\boldsymbol{w}}(\boldsymbol{z})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_{\mathrm{data}}(\boldsymbol{x})}[\log(1 - d_{\boldsymbol{w}}(f_{\boldsymbol{\varphi}}(\boldsymbol{x})))]. \tag{9}$$

The loss function of generator is

$$\mathcal{L}_{\mathrm{GEN}}(\boldsymbol{\varphi}; \boldsymbol{x}) = -\mathop{\mathbb{E}}_{\boldsymbol{x} \sim P_{\mathrm{data}}(\boldsymbol{x})}[\log(d_{\boldsymbol{w}}(f_{\boldsymbol{\varphi}}(\boldsymbol{x})))]. \tag{10}$$

In many practices of previous studies [22,25], the Gaussian or Uniform distribution is chosen as a prior for learning robust representations. Note that, in this work, the prior distribution produces real samples, rather than serving as a source of noise for generating fake samples as in [55].

3.5. Algorithm

3.5.1. AGAIN minibatch training

The minibatch training procedure of AGAIN is shown in Algorithm 1. In the inductive learning phase, GNN encoder $f_{\boldsymbol{\varphi}}(\cdot)$ and node classifier $l_{\boldsymbol{\psi}}(\cdot)$ are updated to minimize the cross-entropy loss of labeled nodes (Steps 2–10). The total labeled nodes of an attributed graph are randomly shuffled first, and then equally divided into several batches which are then processed one by one. Therefore, a batch of labeled nodes can be considered to be

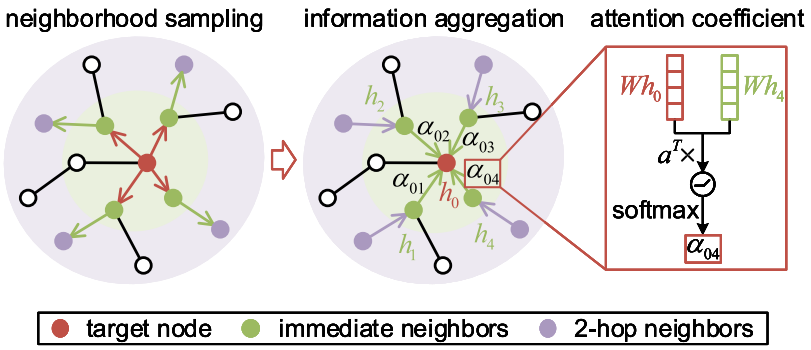

Fig. 3. Illustration of the neighborhood sampling and the subsequent information aggregation process. The sign "×" denotes matrix multiplication. Attention coefficient is activated by the leaky ReLU nonlinearity before softmax operation.

Algorithm 1 AGAIN Minibatch Training

Input: Graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$; maximum training epoch $n_{\max}$; maximum search depth K; number of discriminator training per generator iteration n_{D}; attention-based aggregator function AGG_k (including weight vector $\boldsymbol{a}^k$, weight matrix $\boldsymbol{W}^k$, and nonlinear activation σ_1), $k \in \{1, \ldots, K\}$; weight matrices $\boldsymbol{W}_v^k$ and $\boldsymbol{W}_S^k$, $k \in \{1, \ldots, K\}$; nonlinearity σ_2.

1: **for** epoch $< n_{\max}$ **do**
2: Sample a batch of labeled nodes (i.e., $\boldsymbol{B}$) with initial representations set as $\boldsymbol{h}_v^0 = \boldsymbol{x}_v (v \in \boldsymbol{B})$ and sample the neighboring features $\boldsymbol{x}_S$ (including those of the immediate neighbors, i.e., $\boldsymbol{S}_v$).
3: **for** $k = 1, \ldots, K$ **do**
4: $\boldsymbol{h}_S^k = \mathrm{AGG}_k(\boldsymbol{h}_u^{k-1} \mid u \in \boldsymbol{S}_v)$
5: $\boldsymbol{h}_v^k = \sigma_2([\boldsymbol{W}_v^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}_S^k \boldsymbol{h}_S^k])$
6: $\boldsymbol{h}_v^k = \boldsymbol{h}_v^k / \parallel \boldsymbol{h}_v^k \parallel_2$
7: **end for**
8: $\boldsymbol{u}_v = \boldsymbol{h}_v^K = f_\varphi(\boldsymbol{x}_v, \boldsymbol{x}_S), \hat{\boldsymbol{y}}_v = l_\psi(\boldsymbol{u}_v)$.
9: Compute the cross-entropy loss using Eq. (7)
10: Backpropagate loss and update φ and ψ
11: **for** $n < n_{\mathrm{D}}$ **do**
12: Sample a batch of nodes $\boldsymbol{x}_v (v \in \boldsymbol{B})$ and compute embeddings $\boldsymbol{u}_v$
13: Sample a batch from the prior distribution $\boldsymbol{z}_i \sim P_g(\boldsymbol{z})(i = 1, \ldots, |\boldsymbol{B}|)$
14: Compute $\mathcal{L}_{\mathrm{DIS}}$ using Eq. (9)
15: Backpropagate loss and update $\boldsymbol{w}$
16: **end for**
17: Sample a batch of nodes $\boldsymbol{x}_v (v \in \boldsymbol{B})$ and compute embeddings $\boldsymbol{u}_v$
18: Compute $\mathcal{L}_{\mathrm{GEN}}$ using Eq. (10)
19: Backpropagate loss and update φ
20: **end for**

Algorithm 2 AGAIN Testing

Input: Graph $\mathcal{G}(\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$; test nodes (i.e., $\boldsymbol{V}_{\mathrm{U}}^{\mathrm{n}}$); maximum search depth K; trained GNN encoder $f_\varphi(\cdot)$ (including attention-based aggregator function AGG_k, weight matrices $\boldsymbol{W}_v^k$ and $\boldsymbol{W}_S^k$, and nonlinearity σ_2, $k \in \{1, \ldots, K\}$); trained node classifier $l_\psi(\cdot)$.

1: Set the initial representations as $\boldsymbol{h}_v^0 = \boldsymbol{x}_v (v \in \boldsymbol{V}_{\mathrm{U}}^{\mathrm{n}})$ and sample the neighboring features $\boldsymbol{x}_S$ (including those of the immediate neighbors, i.e., $\boldsymbol{S}_v$).
2: **for** $k = 1, \ldots, K$ **do**
3: $\boldsymbol{h}_S^k = \mathrm{AGG}_k(\boldsymbol{h}_u^{k-1} \mid u \in \boldsymbol{S}_v)$
4: $\boldsymbol{h}_v^k = \sigma_2([\boldsymbol{W}_v^k \boldsymbol{h}_v^{k-1}; \boldsymbol{W}_S^k \boldsymbol{h}_S^k])$
5: $\boldsymbol{h}_v^k = \boldsymbol{h}_v^k / \parallel \boldsymbol{h}_v^k \parallel_2$
6: **end for**
7: $\boldsymbol{u}_v = \boldsymbol{h}_v^K = f_\varphi(\boldsymbol{x}_v, \boldsymbol{x}_S), \hat{\boldsymbol{y}}_v = l_\psi(\boldsymbol{u}_v)$.
8: Calculate the classification accuracy based on label prediction $\hat{\boldsymbol{y}}_v$ and ground truth label.

randomly sampled from the total labeled nodes. Taking one of the selected labeled nodes as a target node, we sample its neighboring nodes and aggregate neighborhood information to compute its embedding vector and label prediction. Cross-entropy loss is calculated based on the predictions and ground-truth labels using Eq. (7). When executing from Steps 2–10, although only one batch of labeled nodes is considered, the whole graph, except for the test data, is accessible in the neighborhood aggregation process. In other words, in addition to the feature vector of a target node, the features of its neighboring nodes, which are sampled from the whole graph, are also involved in the computation of target node embedding vector.

In adversarial training phase, the adversarial networks first update discriminator $d_{\boldsymbol{w}}(\cdot)$ to tell apart real samples (vectors from prior distribution) from fake samples, i.e., embedding vectors (Steps 11–16). In addition, GNN encoder, $f_{\boldsymbol{\varphi}}(\cdot)$, serve as a generator to confuse the trained discriminator and update itself (Steps 17–19). Therefore, the parameters of GNN encoder $f_{\boldsymbol{\varphi}}(\cdot)$ are updated by inductive learning and adversarial training alternatively.

The computational complexity of inductive learning is proportional to the number of parameters $|\boldsymbol{\varphi}|$ and $|\boldsymbol{\psi}|$ in every epoch. Hence, the overall complexity is $O(n_{\max}(|\boldsymbol{\varphi}| + |\boldsymbol{\psi}|))$. Similarly, the computational complexity of generator and discriminator is typically linear with the number of parameters $|\boldsymbol{\varphi}|$ and $|\boldsymbol{w}|$, respectively. Therefore, the complexity of adversarial training is $O(n_{\max}(n_{\mathrm{D}}\,|\boldsymbol{w}| + |\boldsymbol{\varphi}|))$.

3.5.2. AGAIN testing

Algorithm 2 outlines the process of testing. In the testing phase, since GNN encoder $f_{\boldsymbol{\varphi}}(\cdot)$ and node classifier $l_{\boldsymbol{\psi}}(\cdot)$ have been trained, their learnable parameters are fixed. Then the features of test nodes and their sampled neighboring features are fed into the trained model to obtain node embeddings and label predictions (Steps 1–7). Finally, the classification accuracy is calculated based on the label predictions and ground truth labels.

It can be seen in Fig. 1 that, in the test phase, the local neighborhoods of existing nodes would change, and the local structures of new nodes are newly formed. The proposed inductive learning model can handle both situations and compute the representations for all nodes in the graph. GNN encoder, $f_{\boldsymbol{\varphi}}(\cdot)$, generates node representations by aggregating the neighborhood information (see Eqs. (2)–(4)). Once the GNN encoder is trained based on

Table 2
Summary of datasets.

Dataset	#Nodes[a] N	#Edges $\|\boldsymbol{E}\|$	Average degree $\langle k \rangle$	#Labels C	#Features D
Cora	2708	5429	4.0	7	1433
CiteSeer	3327	4732	2.8	6	3,703
PubMed	19,717	44,338	4.5	3	500
BlogCatalog	5196	171,743	66.1	6	8189

[a]"#Nodes" means the number of nodes. The rest can be deduced by analogy.

the available information of the original graph, its parameters are fixed in the test phase. Although the graph topology has changed during testing, the GNN encoder can still compute node representations. In Eq. (2), as long as the updated or new neighborhood information is provided, the representation of neighborhood can be computed, enabling subsequent calculation.

4. Experiments

In this section, we aim to answer the following research questions (RQs) by extensive experiments.

- RQ1: How does AGAIN perform on the inductive node classification tasks compared with the state-of-the-art baselines?
- RQ2: What are the benefits of learning strategies, including information aggregation, attention mechanism, and adversarial training?
- RQ3: How is the performance of AGAIN model affected by the relevant hyperparameters?

4.1. Experimental setup

Datasets. We conduct experiments on four real-world datasets as described in Table 2. The three citation graphs [58] (i.e., Cora, CiteSeer and PubMed) have nodes and edges representing publications and citation links, respectively. These publications are categorized based on their corresponding research topics. For example, Cora consists of machine learning papers which belong to one of the seven classes named as "case based", "genetic algorithms", "neural networks", "probabilistic methods", "reinforcement learning", "rule learning", and "theory". For Cora and CiteSeer, each paper is described by a feature vector with binary values indicating whether each word from a dictionary is present. The publications in PubMed have features described by Term Frequency–Inverse Document Frequency (TF–IDF) vectors drawn from a dictionary consisting of 500 unique words. Therefore, for each citation network, the feature dimension of a node, D, is determined by the corresponding dictionary size.

BlogCatalog [59] is an online community in which bloggers follow each other. It is modeled as a social network, with nodes and edges representing bloggers and their following relationships, respectively. The feature vector of each blogger is obtained according to the corresponding blog description. The bloggers are categorized into one of the six predefined categories based on their interests.

The goal of node classification in this work is to classify one publication into a certain research topic, or predict the interest of a blogger. Note that we treat all networks here as undirected graphs. In the performance study (Section 4.2), the labeled nodes of each citation graph are the same as the designated ones in the Planetoid paper [13] for a fair comparison. In the remaining experiments of Section 4, the labeled nodes are randomly selected from the training data. Specifically, we randomly choose the same number of labeled nodes for each class in the training nodes. In [13], the number of labeled nodes per class (i.e., n) is fixed as 20. However, in this work, labeled number, n, varies from 20 to 100 for a more thorough investigation. The remaining training nodes are unlabeled. Under inductive setting, the test nodes are unobserved during training. Following the setting of [13], the number of test nodes in each graph is fixed as 1000.

Baselines. Three groups of baselines are introduced as follows.

- LR, DeepWalk [9], and DeepWalk+: They are unsupervised baselines followed by the logistic regression classifier. LR is directly trained on the node features. DeepWalk generates embedding vector for each node using the graph structure only. In DeepWalk+, node embeddings generated by DeepWalk are further concatenated with node features.
- ManiReg [28], SemiEmb [29], and Planetoid-I [13]: They are graph semi-supervised learning methods. Graph Laplacian regularization is employed in these methods to impose penalty, if nearby nodes are predicted to have different labels. They are inductive baselines which can naturally handle unseen nodes.
- GAT [8] and GraphSAGE [5]: They are GNN models for inductive learning on graphs. GAT devises an attention mechanism to assign learnable weights for the entire neighborhood nodes. The GraphSAGE variants (including GS-GCN, GS-mean, GS-LSTM and GS-pool) employ various aggregator functions to aggregate information from the sampled neighborhood. Among them, GS-GCN is the inductive variant of the GCN model [4]. GS-mean improves on GS-GCN by concatenating the output of previous layer with a skip connection. Such skip connection can also be found in GS-LSTM and GS-pool.

Implementation details. For baselines using logistic regression (i.e., LR, DeepWalk and DeepWalk+), we use the logistic SGDClassifier in the scikit-learn Python package [60] with default settings. For DeepWalk, we follow what is done in GraphSAGE [5]. While fixing the embeddings of already trained nodes, before making predictions, a new round of SGD optimization is performed to update the embeddings of new test nodes. For Planetoid-I, we use the public source code[1] provided by the authors with default settings, and sweep learning rate in the set $\{0.1, 0.01, 0.001\}$.

The PyTorch implementation[2] of GAT model is originally transductive. We adapt this implementation to calculate the GAT results under inductive scenario. The original GraphSAGE variants only have unsupervised and fully-supervised versions. We adapt the fully-supervised version to be semi-supervised, which only has a few labeled nodes during training. Since the graph structure has already been incorporated in the neighborhood sampling process, similar to GCN [4], GraphSAGE variants are directly trained on the supervised loss of labeled nodes, without having to consider the Laplacian regularization.

Fig. 4 shows the layer structure of AGAIN. GNN encoder, $f_{\boldsymbol{\varphi}}(\cdot)$, is a two-layer graph attention model. The dimension of weighting vector, $\boldsymbol{a}$, is selected in the set $\{64, 128, 256, 512, 1024, 2048\}$. The output dimension of each layer (i.e., $\left|\boldsymbol{h}_v^k\right|$ in Algorithm 1) is set as 256. Node classifier, $l_{\boldsymbol{\psi}}(\cdot)$, is a fully-connected single-layer neural network (i.e., FC layer) followed by a softmax activation. Its input and output dimensions are embedding dimension (i.e., $d = 256$) and the number of classes in each dataset (i.e., C), respectively. Discriminator, $d_{\boldsymbol{w}}(\cdot)$, is a four-layer neural network

[1] https://github.com/kimiyoung/planetoid
[2] https://github.com/Diego999/pyGAT

Table 3
Main hyperparameters for the AGAIN model.

Dataset	n	n_{max}	n_D	Batchsize	Learning rate		Weight decay	d	p	Dropout	K	$\boldsymbol{s}^*$
					φ, ψ	w						
BlogCatalog	20	200	5	256	0.001	0.0002	0.005	256	0	0.5	2	{25, 10}
	60											
	100											
Cora	20		1			0.0001	0.05		−4			
	60					0.001						
	100					0.0001						
CiteSeer	20					0.001						
	60											
	100					0.0001						
PubMed	20					0.001						
	60											
	100											

∗ Neighborhood sample size, $\boldsymbol{s}$, is denoted as a set $\{s_1, \ldots, s_K\}$ containing sample size s_k in each search depth k.

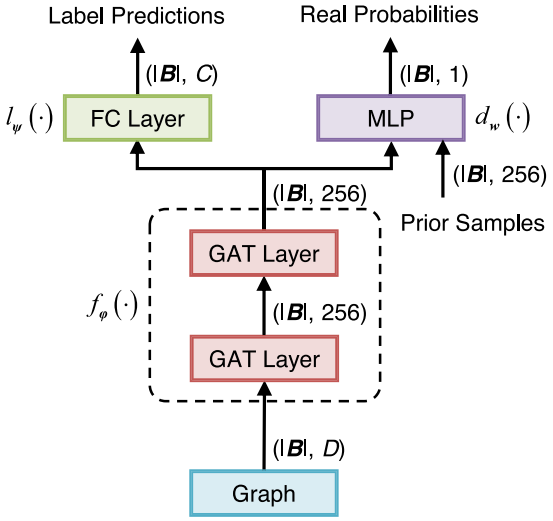

Fig. 4. Layer structure of AGAIN. The input and output dimensions of each block are shown in parentheses, where $|\boldsymbol{B}|$ is the batchsize, D is the feature dimension of a graph, and C is the number of classes.

(i.e., MLP), with the dimensions of three hidden layers set as 1024, 1024 and 256 in sequence. The output of discriminator is of one dimension, indicating the probability of an input sample to be real. We use the leaky ReLU activation (i.e., $\sigma(x) = \max(0.2x, x)$) in the first three layers, and employ a sigmoid activation in the output layer. The default prior of AGAIN is a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, 10^p \boldsymbol{I})$. The dimension of a prior sample is the same as embedding dimension d. Power exponent, p, is swept in the set $\{-4, -2, 0, 2, 4\}$.

AGAIN is implemented using PyTorch [61]. In Table 3, we provide the main hyperparameters selected for each dataset, with n denoting the number of labeled training nodes per class. We train the model for 200 epochs (i.e., $n_{max} = 200$) using the Adam optimizer. The batch size (i.e., $|\boldsymbol{B}|$) is 256. In order to prevent overfitting, L_2 regularization is enforced in the loss function with the weight decay term selected in $\{5e-5, 5e-4, 5e-3, 5e-2\}$. All weights are initialized by default. Note that we set the maximum search depth as $K = 2$, in which neighborhood sample sizes are $s_1 = 25$ and $s_2 = 10$, respectively. As mentioned in [5], increasing K beyond 2 leads to marginal accuracy improvement, while large increment can be seen in the corresponding runtime. The influence of sample size (i.e., s) on classification accuracy is discussed in Section 4.4.

For fair comparison, the above methods have the same embedding dimension, i.e., $d = 256$. All results are averaged by ten runs with different random seeds. We run the experiments using a computer with one NVIDIA GeForce GTX 1080Ti GPU (11 GB of RAM), an Intel(R) Core(TM) i7-8700K CPU (6 cores, 3.70 GHz), and 32 GB of RAM.

4.2. Performance study (RQ1)

In this section, experiments are first conducted on the inductive benchmark task to verify the proposed method. Then the t-SNE visualization of node representation is provided. Finally, the proposed methods are further evaluated under the transductive setting.

4.2.1. Inductive node classification

Following the dataset split of the Planetoid paper [13], in the training nodes, there are only 20 labeled nodes for each class, i.e., $n = 20$. 1000 nodes are selected as the test data. In Table 4, for the citation datasets, the accuracies of ManiReg, SemiEmb, and Planetoid-I are taken from [13]. Since the results of these methods on BlogCatalog are not reported in [13], the corresponding cells of ManiReg and SemiEmb are left empty. As stated in Section 4.1, the performance of Planetoid-I in BlogCatalog is obtained by executing the source code. In the subsequent discussions, AGAIN is the full proposed model shown in Fig. 2. GAIN is our simplified model without adversarial training. That is, GAIN only consists of the GNN encoder and the node classifier. It can be seen in Table 4 that, logistic regression classifier (LR) produces the largest standard deviation values, possibly due to its simplicity. When testing on the same graph, the standard deviation values of other methods are generally close and of the same order. Therefore, the following discussions are based on the mean accuracy.

In the first group of baselines (i.e., LR, DeepWalk, and DeepWalk+), LR obtains much higher accuracies than DeepWalk. This indicates that, for attributed graphs, node features can be more informative than graph structure in learning node embeddings. Note that, although DeepWalk is far more competitive in transductive learning, it has poor performance on inductive tasks. Furthermore, DeepWalk performs worst in CiteSeer, which is probably attributed to the low average degree (see Table 2). The importance of feature information is further validated by the performance lift of DeepWalk+ compared to DeepWalk, after concatenating node features with the learned embeddings. It is found in CiteSeer and PubMed that, although utilizing both structure and feature information, DeepWalk+ cannot surpass LR. Hence, it is not always workable by simply concatenating structural embeddings and node features. In other words, graph structure and feature information need to be incorporated in a systematic manner.

Compared with the first group of baselines, superior performance is observed in the graph-based semi-supervised learning methods (i.e., ManiReg, SemiEmb, and Planetoid-I), which is

Table 4
Mean classification accuracy on test data under inductive setting (in percent). For each dataset, the highest mean accuracy is highlighted in bold and the top two are underlined. The standard deviations are given in parentheses.

Method	BlogCatalog	Cora	CiteSeer	PubMed
LR	66.4 (3.6)	51.6 (2.6)	51.0 (1.5)	71.4 (3.5)
DeepWalk [9]	25.4 (1.3)	29.4 (1.3)	22.9 (1.1)	48.2 (2.2)
DeepWalk+	66.5 (2.5)	55.9 (0.9)	49.0 (0.6)	67.2 (0.8)
ManiReg [28]	–	59.5	60.1	70.7
SemiEmb [29]	–	59.0	59.6	71.1
Planetoid-I [13]	73.2 (2.0)	61.2	64.7	77.2
GAT [8]	63.7 (2.6)	**80.6** (0.4)	67.7 (1.0)	**77.8** (0.7)
GS-GCN [5]	59.2 (2.7)	77.6 (1.2)	67.4 (0.5)	76.0 (0.7)
GS-mean [5]	77.1 (2.3)	79.8 (0.5)	68.8 (0.5)	76.9 (0.6)
GS-LSTM [5]	74.5 (1.9)	78.4 (0.4)	67.2 (1.1)	76.0 (0.7)
GS-pool [5]	73.9 (2.0)	80.2 (0.7)	68.1 (0.7)	77.1 (0.5)
GAIN [ours]	79.3 (1.9)	80.0 (0.7)	69.2 (0.6)	77.1 (0.6)
AGAIN [ours]	**80.1** (1.7)	79.9 (0.4)	**70.0** (0.8)	77.5 (0.7)

Table 5
Silhouette score of the clusters in a 2D projected space.

Dataset	Planetoid-I	GS-pool	GAIN	AGAIN
Cora	0.034	0.298	0.280	0.325
BlogCatalog	0.231	0.230	0.284	0.341

yielded by jointly incorporating the information of features, structure, and labels in an attributed graph. Among them, Planetoid-I is the most competitive one. Further improvements can be seen in the GraphSAGE variants.

On Cora and PubMed, we observe several inductive GNN models yield close performance, including GAT, GS-mean, GS-pool, GAIN, and AGAIN. On CiteSeer and BlogCatalog, AGAIN has clear performance gains over other GNN models. Specifically, with the help of attention mechanism and skip connection, GAIN outperforms GraphSAGE variants and GAT. Then, AGAIN further improves on GAIN by adversarial training which increases the generalization ability. Compared with those of Cora and PubMed, node feature vectors have larger dimensions in BlogCatalog and CiteSeer (see Table 2). Thus, the above observations reveal the strength of our methods in performing inductive learning on feature-rich graphs. In particular, AGAIN outperforms GAT by 6.4% on BlogCatalog and 2.3% on CiteSeer. On BlogCatalog, GAT surpasses the inductive variant of GCN (i.e., GS-GCN), but underperforms other GraphSAGE variants which employ skip connection and advanced aggregator. Note that, through concatenating the output of previous layer, GS-mean outperforms GS-GCN in all cases, showing the benefit of skip connection.

4.2.2. Visualization of embedding vectors

Fig. 5 visualizes the embedding vectors of the test nodes in Cora and BlogCatalog using t-SNE [62]. We select Planetoid-I and GS-pool as representative baselines, and neglect methods in the first group due to their low accuracies. As shown in Table 5, we further calculate the corresponding Silhouette score [63] for the clusters in a 2D projected space. Embedding vectors generated by AGAIN have the most preferable visualization. Specifically, the clusters are separated more clearly, yielding the highest Silhouette score.

4.2.3. Transductive node classification

Though this work aims at inductive learning, to make the evaluation more comprehensive, we further conduct experiments under the transductive settings. In Table 6, the results are presented and compared with the classical transductive GNN model, i.e., GCN [4]. We reuse the GCN results reported in [4] for Cora, CiteSeer, and PubMed. The GCN performance in BlogCatalog is evaluated by adapting and executing the source codes[3] provided by the authors. Since the test set information is originally assumed to be unavailable when designing the inductive approaches, such information may not be well exploited by the inductive approaches, consequently leading to their underperformance in some cases.

Table 6
Mean classification accuracy on test data under transductive setting (in percent). For each dataset, the highest mean accuracy is highlighted in bold and the top two are underlined. The standard deviations are given in parentheses.

Method	BlogCatalog	Cora	CiteSeer	PubMed
GCN [4]	65.0 (2.3)	**81.5**	70.3	**79.0**
GS-GCN [5]	59.5 (2.0)	78.4 (1.1)	67.2 (0.7)	76.5 (0.9)
GS-mean [5]	76.2 (2.8)	80.0 (0.6)	69.2 (0.7)	76.6 (0.5)
GS-LSTM [5]	73.7 (2.0)	79.4 (0.7)	67.4 (1.4)	75.6 (0.5)
GS-pool [5]	73.3 (2.1)	80.3 (0.5)	68.6 (0.4)	77.4 (0.7)
GAIN [ours]	79.2 (2.4)	80.4 (0.4)	69.6 (0.7)	76.6 (0.7)
AGAIN [ours]	**79.8** (2.2)	80.3 (0.6)	**70.5** (0.8)	77.6 (0.6)

It is found that, in Cora, CiteSeer and PubMed, the transductive accuracies in Table 6 are generally higher than the corresponding inductive ones in Table 4. The reason is that, unlike inductive learning, the information of test data is accessible in transductive learning during training. However, in BlogCatalog, the transductive accuracies are mostly lower than those in inductive cases. It might be attributed to the high average degree of BlogCatalog (i.e., 66.1, see Table 2). The sample sizes are 25 and 10 in the immediate neighbors and the 2-hop neighbors (see Table 3 and Fig. 3), respectively. Therefore, under the inductive setting, there will already be rich neighborhood information to exploit. In transductive setting, although test set information is available when sampling neighboring nodes, more abundant neighborhood information can sometimes introduce certain noise, which would lead to underperformance.

Although AGAIN is designed for inductive learning, compared with the classical GCN model [4], its results are still matched in CiteSeer and even higher in BlogCatalog. GCN surpasses its inductive variant GS-GCN in BlogCatalog. However, with advanced aggregator and skip connection employed, the remaining GNN models manage to yield large performance gains over GCN. Note that, in all cases, one of the proposed methods (i.e., GAIN or AGAIN) is able to outperform the GraphSAGE variants.

4.3. Ablation study (RQ2)

In this section, we investigate the influences of information aggregation, attention mechanism, and adversarial training on

[3] https://github.com/tkipf/pygcn

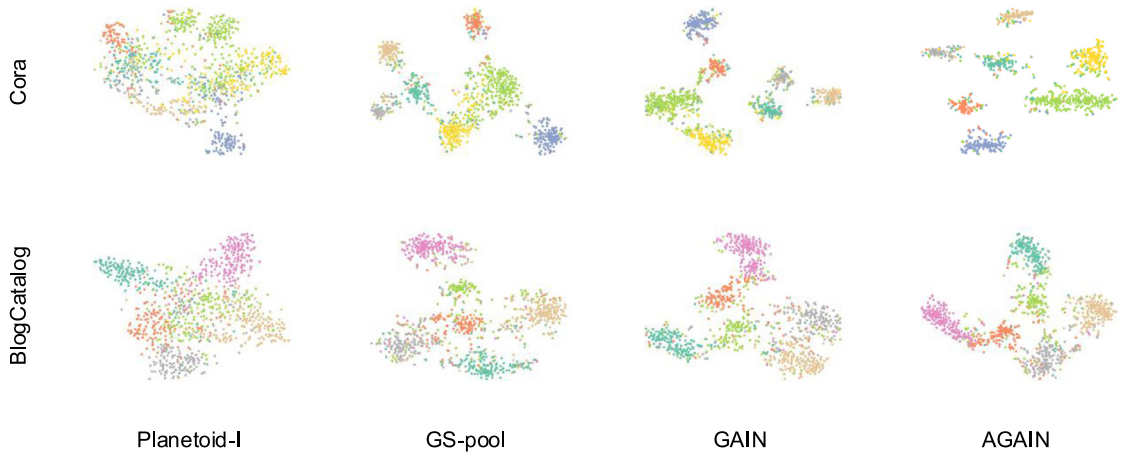

Fig. 5. Visualization of the embedding vectors of Cora and BlogCatalog in the 2D space using t-SNE (best viewed in color). For Cora, each point corresponds to one paper. Seven colors distinguish different paper classes. For BlogCatalog, each point represents one blogger. Six colors denote different interests.

Table 7
Summary of mean classification accuracy (in percent). The number of labeled nodes per class, n, varies from 20 to 100. Bold font denotes the top model. The standard deviations are given in parentheses.

Dataset	BlogCatalog			Cora			CiteSeer			PubMed		
n	20	60	100	20	60	100	20	60	100	20	60	100
MLP	73.1 (1.6)	84.1 (0.9)	87.6 (0.5)	59.3 (1.2)	67.9 (1.2)	71.5 (1.3)	56.0 (1.2)	66.6 (1.8)	69.5 (2.0)	73.4 (0.5)	73.9 (1.9)	76.3 (1.6)
GS-mean	77.1 (2.3)	86.1 (1.7)	89.2 (0.8)	79.8 (0.5)	80.9 (0.8)	83.2 (0.4)	68.8 (0.5)	73.0 (1.1)	74.2 (1.0)	76.9 (0.6)	78.3 (1.7)	81.4 (1.0)
GAIN	79.3 (1.9)	89.1 (1.4)	91.4 (0.7)	**80.0** (0.7)	81.1 (1.2)	**83.4** (1.0)	69.2 (0.6)	**73.6** (1.3)	74.6 (0.9)	77.1 (0.6)	78.9 (1.6)	81.2 (1.1)
AGAIN	**80.1** (1.7)	**89.2** (1.0)	**91.5** (0.8)	79.9 (0.4)	**82.2** (1.4)	83.1 (1.0)	**70.0** (0.8)	73.1 (1.0)	**74.9** (0.5)	**77.5** (0.7)	**79.2** (1.8)	**81.8** (1.4)

learning node embeddings step by step. We construct a two-layer MLP, which only uses node features as input, without having to consider graph structure, and outputs predictions. The first layer of MLP is similar to GNN encoder $f_\varphi(\cdot)$ in Fig. 2. The second layer can be treated as node classifier $l_\psi(\cdot)$. Therefore, the hidden dimension of MLP is set as embedding dimension d. Referring to the neighborhood representation obtained using Eq. (2), GS-mean takes the average of the representations of neighbors with equal weights. GAIN further assigns different learnable weights (i.e., attention coefficients) to these neighboring nodes. Then AGAIN combines adversarial training with GAIN, constraining the learned embeddings to match a prior distribution. Note that the subsequent experiments are all conducted under inductive setting.

In Table 7, the methods based on information aggregation (i.e., GS-mean, GAIN, and AGAIN) are superior to MLP which solely exploits node features. Compared with GS-mean, GAIN obtains higher accuracies in most cases. Large margins can be seen on BlogCatalog, where GAIN achieves on average 2.93% relative gain in accuracy over GS-mean. However, the margins are small on citation graphs which are relatively sparse. This indicates that the attention mechanism can be more powerful on a dense graph containing rich features. In terms of the mean accuracy, AGAIN outperforms GAIN in 9 out of 12 cases, showing AGAIN has a slightly improved generalization ability when evaluated on unseen test nodes.

To further investigate the effects of attention mechanism and adversarial training on improving the robustness of embeddings, we corrupt the node features in test phase after the models are trained. Referring to GIB [51], we randomly choose a percentage of nodes (denoted as η), and add independent Gaussian noise $(\lambda \cdot r \cdot \epsilon)$ to each dimension of their feature vectors, with increasing amplitude. Random number, ϵ, is from standard normal distribution $\mathcal{N}(0, 1)$. Feature noise ratio, λ, is selected in the set $\{0, 0.5, 1.0, 1.5\}$. When λ equals 0, it is the case without noise. To incorporate the graph property during noise injection, reference amplitude, r, is obtained by taking the average of the maximum value in each node's feature vector. As stated in Section 2, the noise here is similar to the evasion attacks.

In real applications, it would be a frequently encountered situation that a small fraction of nodes are noisy. Therefore, the percentage of nodes with noise, η, is first fixed as 10%. The experimental results are reported in Table 8. GAIN outperforms GS-mean in most cases, which indicates the potential of attention mechanism on improving robustness. Moreover, the accuracy of AGAIN is higher than that of GAIN, except those on CiteSeer. This reveals that adversarial training contributes to generating robust embeddings in some degree. For comprehensive evaluation, we also present the results of Planetoid-I and GS-pool, which are found to be mostly inferior to those of our methods. There is one exception on BlogCatalog, where Planetoid-I performs best when the noise amplitude is high (i.e., $\lambda = 1.5$). The likely reason is that the methods relying on information aggregation are influenced by the relational effect of graph structure [16]. Referring to Fig. 3, the noise added on one node might misguide the predictions of other nodes with structural relations, worsening the model performance, especially when the high-intensity noise is injected into a dense graph.

Further investigations are conducted by varying the percentage of nodes with noise (i.e., η). For the sake of clarity, performance gap, Δ, is obtained through subtracting the accuracy of GAIN from that of GS-mean or AGAIN. As shown in Fig. 6,

Table 8
Mean classification accuracy in percent for the trained models with increasing additive feature noise ($n = 60$, $\eta = 10\%$). Bold font denotes the top model.

Dataset	BlogCatalog				Cora				CiteSeer				PubMed			
λ	0	0.5	1.0	1.5	0	0.5	1.0	1.5	0	0.5	1.0	1.5	0	0.5	1.0	1.5
Planetoid-I	83.0	77.6	76.9	**76.7**	69.2	66.1	65.0	64.7	69.3	66.7	65.4	65.0	74.7	71.6	71.0	70.6
GS-pool	84.8	74.2	63.0	54.6	80.8	76.3	67.5	61.2	72.5	68.4	62.3	58.4	78.0	72.7	69.5	67.6
GS-mean	86.1	80.4	77.7	74.9	80.9	77.6	72.4	68.9	73.0	70.5	67.3	64.7	78.3	74.4	71.5	69.7
GAIN	89.1	82.0	75.3	68.4	81.1	77.9	72.9	69.7	**73.6**	**72.0**	**68.6**	**65.9**	78.9	74.8	71.8	69.7
AGAIN	**89.2**	**82.2**	**78.2**	72.5	**82.2**	**78.8**	**74.7**	**71.1**	73.1	71.1	67.7	65.3	**79.2**	**75.4**	**72.7**	**71.4**

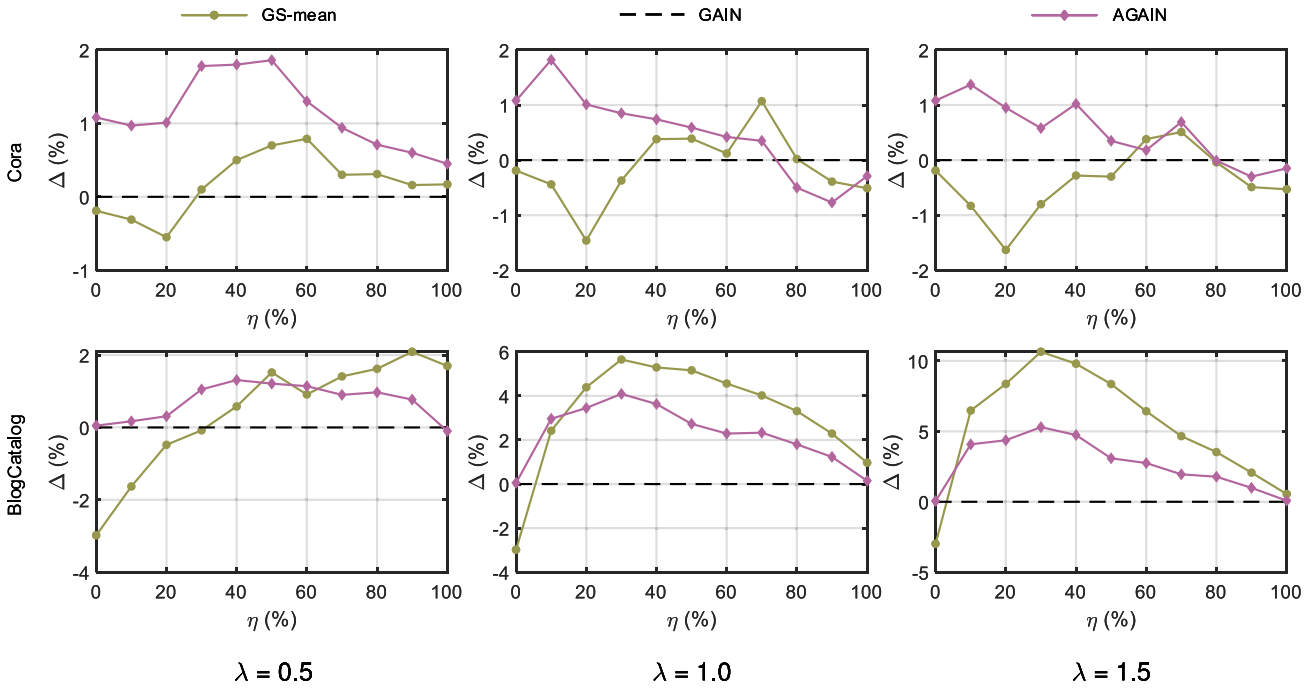

Fig. 6. Performance gap (i.e., Δ) on Cora and BlogCatalog ($n = 60$). η is the percentage of nodes in a graph corrupted with additive feature noise. λ is the feature noise ratio.

the performance of attention mechanism and adversarial training varies with node percentage and graph property. In general, as the percentage of nodes with noise increases, the attention mechanism brings an improvement first but loses effects gradually. With adversarial training employed, AGAIN outperforms GAIN by clear margins in most cases, revealing adversarial training is able to improve model robustness. However, the performance gain yielded by adversarial training shrinks or even becomes negative when more nodes are corrupted with noise.

4.4. Hyperparameter sensitivity study (RQ3)

In this section, we analyze the classification accuracy of AGAIN with regard to four relevant hyperparameters, i.e., embedding dimension d, neighborhood sample size s, discriminator learning rate p_r, and weight decay coefficient p_c. When one hyperparameter is investigated, the remaining hyperparameters are set as the default values introduced in Section 4.1. Fig. 7 displays the classification accuracies on the four graphs.

Embedding dimension, d, is the dimension of node representation vector learned by the AGAIN model. The prediction accuracy increases with the embedding dimension first and then becomes stable. Similar trends can be seen on Cora and PubMed, when increasing the number of sampled neighbors (i.e., s). However, there are little variations on CiteSeer due to its low average degree (see Table 2). In contrast, when the test is applied on BlogCatalog which has relatively high density, the accuracy increases steadily with the sample size. Note that, when investigating the sample size, we select the same number of neighbors in each search depth, i.e., $s_1 = s_2 = s$ (see Table 3). In the case that the learning rate of discriminator (i.e., 1e-2) is much larger than that of the GNN encoder (i.e., 1e-3), a clear performance drop is observed on each graph. When evaluated on Cora and PubMed, the model is more sensitive to the discriminator learning rate. On BlogCatalog, the best accuracy is obtained with a weight decay coefficient of 5e-3. On the three citation graphs (i.e., Cora, CiteSeer, and PubMed), the classification accuracy reaches its peak value when $p_c =$ 5e-2.

5. Conclusion

An adversarially regularized GNN model, AGAIN, has been proposed to address the inductive node classification problem

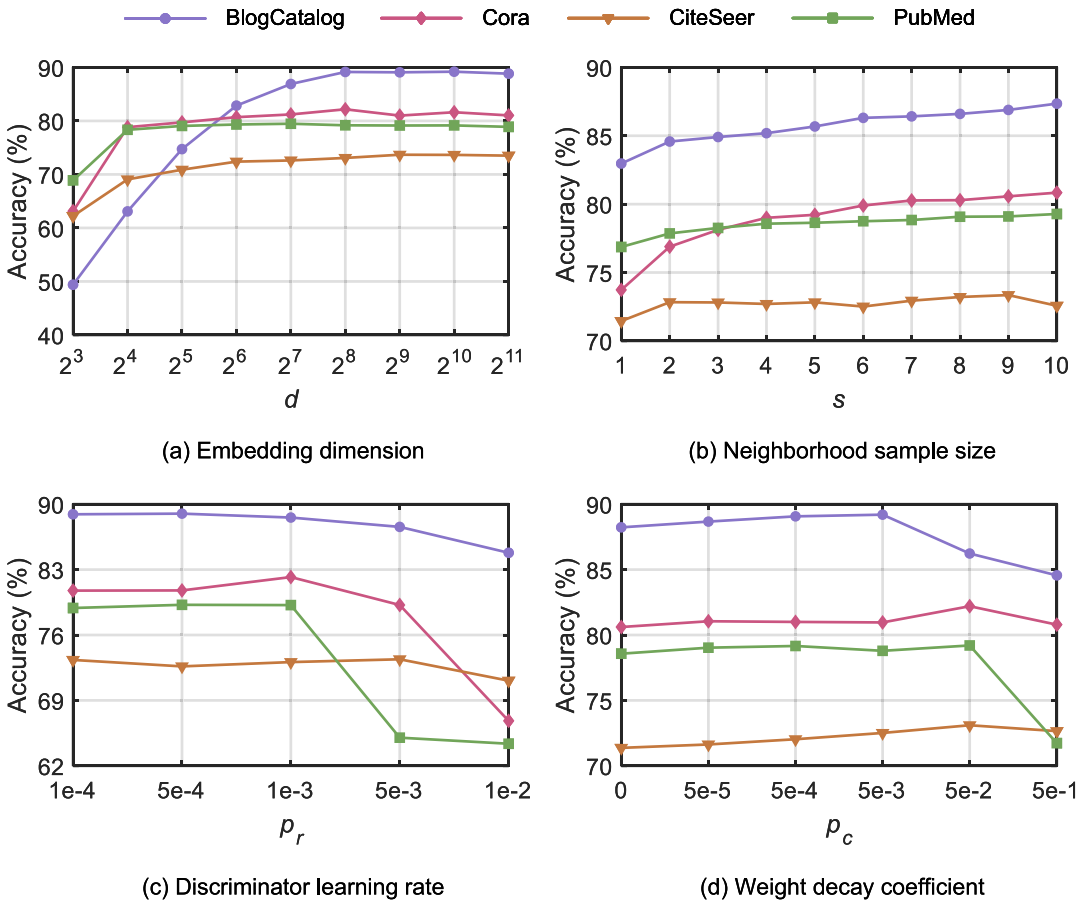

Fig. 7. Accuracy varied with four different hyperparameters individually ($n = 60$).

on partially labeled graphs. AGAIN generates an informative representation vector for an unseen node with an attention-based aggregator that aggregates information from its neighbors. Adversarial training is employed to improve model robustness and generalization ability by matching node representations with a prior distribution. Experimental results on inductive node classification tasks show that our method achieves matched or even more favorable performance compared with the state-of-the-art methods.

CRediT authorship contribution statement

Jiaren Xiao: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Quanyu Dai:** Conceptualization, Methodology, Software. **Xiaochen Xie:** Conceptualization, Project administration, Writing – review & editing. **James Lam:** Conceptualization, Supervision, Funding acquisition. **Ka-Wai Kwok:** Conceptualization, Supervision, Funding acquisition.

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.

Data availability

The dataset link is in the manuscript, data file is not required.

Acknowledgments

This work was supported in part by the Research Grants Council (RGC) of Hong Kong (17201820, 17207020, 17205919), as well as the Innovation and Technology Commission (ITC), Hong Kong (MRP/029/20X), and Centre for Transformative Garment Production (TransGP) funded by ITC, Hong Kong.

References

[1] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowl.-Based Syst. (ISSN: 0950-7051) 151 (2018) 78–94.
[2] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Trans. Knowl. Data Eng. 31 (5) (2018) 833–852.
[3] D. Eswaran, S. Günnemann, C. Faloutsos, D. Makhija, M. Kumar, Zoobp: Belief propagation for heterogeneous networks, In Proc. of the VLDB Endow 10 (5) (2017) 625–636.
[4] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of International Conference on Learning Representations, 2017.
[5] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. (ISSN: 2162-2388) (2020) 1–21.
[7] J.B. Lee, R.A. Rossi, S. Kim, N.K. Ahmed, E. Koh, Attention models in graphs: A survey, ACM Trans. Knowl. Discov. Data (ISSN: 1556-4681) 13 (6) (2019) 62:1–62:25.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: Proceedings of International Conference on Learning Representations, 2018.
[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, ISBN: 978-1-4503-2956-9, 2014, pp. 701–710.
[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1067–1077.
[11] S. Cao, W. Lu, Q. Xu, GraRep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, ISBN: 978-1-4503-3794-6, 2015, pp. 891–900.
[12] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1105–1114.
[13] Z. Yang, W.W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, in: Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 40–48.
[14] A. Bojchevski, S. Günnemann, Adversarial attacks on node embeddings via graph poisoning, in: Proceedings of International Conference on Machine Learning, 2019, pp. 695–704.
[15] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song, Adversarial attack on graph structured data, in: Proceedings of International Conference on Machine Learning, 2018, pp. 1115–1124.
[16] D. Zügner, A. Akbarnejad, S. Günnemann, Adversarial attacks on neural networks for graph data, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 2847–2856.
[17] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: Homophily in social networks, Annu. Rev. Sociol. 27 (1) (2001) 415–444.
[18] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[19] Q. Dai, Q. Li, J. Tang, D. Wang, Adversarial network embedding, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[20] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, C. Zhang, Adversarially regularized graph autoencoder for graph embedding, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, ISBN: 978-0-9992411-2-7, 2018, pp. 2609–2615.
[21] W. Yu, C. Zheng, W. Cheng, C.C. Aggarwal, D. Song, B. Zong, H. Chen, W. Wang, Learning deep network representations with adversarially regularized autoencoders, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 2663–2671.
[22] J. Donahue, P. Krähenbühl, T. Darrell, Adversarial feature learning, in: Proceedings of International Conference on Learning Representations, 2017.
[23] J. Glover, Modeling documents with generative adversarial networks, 2016, ArXiv:1612.09122.
[24] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, 2016, ArXiv:1511.06434.
[25] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, 2016, ArXiv:1511.05644.
[26] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Proceedings of Advances in Neural Information Processing Systems, 2004, pp. 321–328.
[27] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 912–919.
[28] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (Nov) (2006) 2399–2434.
[29] J. Weston, F. Ratle, R. Collobert, Deep learning via semi-supervised embedding, in: Proceedings of the 25th International Conference on Machine Learning, ISBN: 978-1-60558-205-4, 2008, pp. 1168–1175.
[30] Z. Fang, J. Lu, A. Liu, F. Liu, G. Zhang, Learning bounds for open-set learning, in: Proceedings of the 38th International Conference on Machine Learning, 2021, pp. 3122–3132.
[31] Z. Fang, Y. Li, J. Lu, J. Dong, B. Han, F. Liu, Is out-of-distribution detection learnable? in: Proceedings of Advances in Neural Information Processing Systems, 2022.
[32] 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.
[33] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[34] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Chang, Network representation learning with rich text information, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[35] S. Pan, J. Wu, X. Zhu, C. Zhang, Y. Wang, Tri-party deep network representation, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI '16, ISBN: 978-1-57735-770-4, 2016, pp. 1895–1901.
[36] D. Zhang, J. Yin, X. Zhu, C. Zhang, User profile preserving social network embedding, in: Proceedings of International Joint Conference on Artificial Intelligence, 2017, pp. 3378–3384.
[37] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2014, ArXiv:1312.6203.
[38] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[39] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1263–1272.
[40] V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu, Recurrent models of visual attention, in: Proceedings of Advances in Neural Information Processing Systems, 27, 2014, pp. 2204–2212.
[41] D. Bahdanau, K.H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Proceedings of International Conference on Learning Representations, 2015.
[42] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.-Y. Yeung, GaAN: Gated attention networks for learning on large and spatiotemporal graphs, 2018, ArXiv:1803.07294.
[43] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: Proceedings of the World Wide Web Conference, 2019, pp. 2022–2032.
[44] K. He, X. Zhang, S. Ren, J. Sun, Identity Mappings in Deep Residual Networks, in: Proceedings of the European Conference on Computer Vision, 9908, Amsterdam, The Netherlands, 2016, pp. 630–645.
[45] A.N. Bhagoji, The role of data geometry in adversarial machine learning (Ph.D. thesis), Princeton University, United States, 2020.
[46] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Proceedings of International Conference on Learning Representations, 2015.
[47] Y. Zhang, X. Tian, Y. Li, X. Wang, D. Tao, Principal component adversarial example, IEEE Trans. Image Process. 29 (2020) 4804–4815.
[48] R. Jia, P. Liang, Adversarial examples for evaluating reading comprehension systems, 2017, ArXiv:1707.07328.
[49] B. Biggio, G. Fumera, F. Roli, Security evaluation of pattern classifiers under attack, IEEE Trans. Knowl. Data Eng. (ISSN: 1558-2191) 26 (4) (2014) 984–996.
[50] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: IEEE European Symposium on Security and Privacy, 2016, pp. 372–387.
[51] T. Wu, H. Ren, P. Li, J. Leskovec, Graph information bottleneck, in: Proceedings of Advances in Neural Information Processing Systems, 33, 2020.
[52] D. Zügner, S. Günnemann, Certifiable robustness and robust training for graph convolutional networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '19, ISBN: 978-1-4503-6201-6, 2019, pp. 246–256.
[53] N. Tishby, F.C. Pereira, W. Bialek, The information bottleneck method, 2000, ArXiv:Physics/0004057.
[54] N. Tishby, N. Zaslavsky, Deep learning and the information bottleneck principle, in: IEEE Information Theory Workshop, 2015, pp. 1–5.
[55] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[56] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of International Conference on Machine Learning, 2017, pp. 214–223.
[57] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Proceedings of Advances in Neural Information Processing Systems, 27, 2014, pp. 2177–2185.

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

[59] X. Huang, J. Li, X. Hu, Label informed attributed network embedding, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 731–739.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. (ISSN: 1532-4435) 12 (2011) 2825–2830.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019, pp. 8026–8037.

[62] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.

[63] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. (ISSN: 0377-0427) 20 (1987) 53–65.