Journal of
Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

Identity-Preserving Adversarial Training for Robust Network Embedding

Cen Ke–Ting, Shen Hua–Wei, Cao Qi, Xu Bing–Bing, Cheng Xue–Qi

View online: http://doi.org/10.1007/s11390–023–2256–4

JCST Official
WeChat Account

JCST WeChat
Service Account

JCST Homepage: https://jcst.ict.ac.cn
SPRINGER Homepage: https://www.springer.com/journal/11390
E–mail: jcst@ict.ac.cn
Online Submission: https://mc03.manuscriptcentral.com/jcst
Twitter: JCST_Journal
LinkedIn: Journal of Computer Science and Technology

Cen KT, Shen HW, Cao Q *et al.* Identity-preserving adversarial training for robust network embedding. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(1): 177-191 Jan. 2024. DOI: 10.1007/s11390-023-2256-4

Identity-Preserving Adversarial Training for Robust Network Embedding

Ke-Ting Cen[1, 2] (岑科廷), Hua-Wei Shen[1, 2, 3, *] (沈华伟), *Senior Member, CCF, Member, IEEE*
Qi Cao[1] (曹　婍), Bing-Bing Xu[1] (徐冰冰), and Xue-Qi Cheng[2, 4] (程学旗), *Fellow, CCF, Member, IEEE*

[1] *Data Intelligence System Research Center, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

[2] *University of Chinese Academy of Sciences, Beijing 101480, China*

[3] *Beijing Academy of Artificial Intelligence, Beijing 100000, China*

[4] *Chinese Academy of Sciences Key Laboratory of Network Data Science and Technology, Institute of Computing Technology Chinese Academy of Sciences, Beijing 100190, China*

E-mail: cenketing@ict.ac.cn; shenhuawei@ict.ac.cn; caoqi@ict.ac.cn; xubingbing@ict.ac.cn; cxq@ict.ac.cn

Received February 21, 2022; accepted April 17, 2023.

Abstract Network embedding, as an approach to learning low-dimensional representations of nodes, has been proved extremely useful in many applications, e.g., node classification and link prediction. Unfortunately, existing network embedding models are vulnerable to random or adversarial perturbations, which may degrade the performance of network embedding when being applied to downstream tasks. To achieve robust network embedding, researchers introduce adversarial training to regularize the embedding learning process by training on a mixture of adversarial examples and original examples. However, existing methods generate adversarial examples heuristically, failing to guarantee the imperceptibility of generated adversarial examples, and thus limit the power of adversarial training. In this paper, we propose a novel method Identity-Preserving Adversarial Training (IPAT) for network embedding, which generates imperceptible adversarial examples with explicit identity-preserving regularization. We formalize such identity-preserving regularization as a multi-class classification problem where each node represents a class, and we encourage each adversarial example to be discriminated as the class of its original node. Extensive experimental results on real-world datasets demonstrate that our proposed IPAT method significantly improves the robustness of network embedding models and the generalization of the learned node representations on various downstream tasks.

Keywords network embedding, identity-preserving, adversarial training, adversarial the example

1 Introduction

Network embedding aims to learn low-dimensional representation for each node, preserving the network structure or node attributes[1, 2]. The learned representations can benefit various downstream tasks, e.g., node classification[1, 3, 4] and link prediction[5]. Despite their great success, some researchers find that existing network embedding models are vulnerable to random or adversarial perturbations[6], which may degrade the performance of network embedding when being applied to downstream tasks. Then, how to design a robust network embedding model becomes a crucial problem.

A robust network embedding model is desired to capture the intrinsic structural regularities or node properties of the network, thus being capable of filtering out potential noise and generating stable node embedding. As illustrated in Fig.1, for a graph and its

Regular Paper

This work was supported by the National Natural Science Foundation of China under Grant Nos. U21B2046 and 62102402, and the National Key Research and Development Program of China under Grant No. 2020AAA0105200. This work was also sponsored by CCF-Tencent Open Research Fund under Grant No. RAGR20210108. Hua-Wei Shen was also supported by Beijing Academy of Artificial Intelligence (BAAI).

[*] Corresponding Author

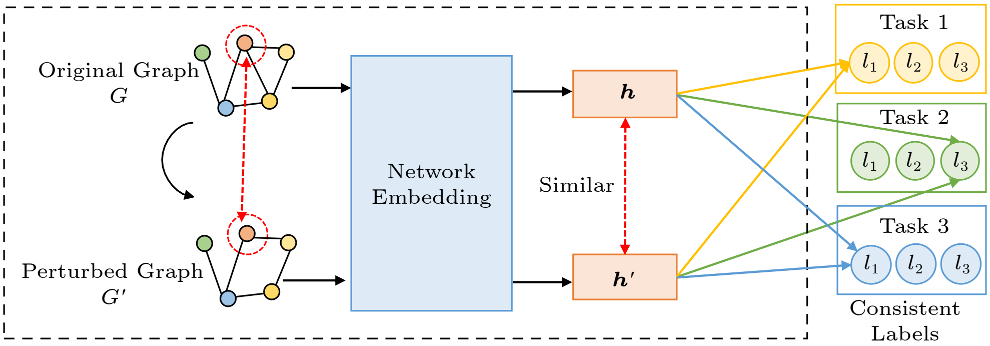


Fig.1. Robust network embedding.

perturbed counterpart, a robust network embedding model should obtain similar node embedding for the same node, thus anticipating that each node and its perturbed counter-part have consistent labels in various downstream tasks.

Recently, adversarial training is leveraged as an effective approach to achieving robust network embedding and improving the generalization of learned representations on downstream tasks[6, 7]. Specifically, adversarial training for network embedding consists of two iterative steps. First, a raw network embedding model is trained based on the original graph. Next, adversarial examples are generated via adding perturbation to nodes' representations obtained by the raw network embedding model, and these adversarial examples are used to improve the robustness of the network embedding model.

For adversarial training, one key is to generate imperceptible adversarial examples according to certain criteria[8], e.g., L_2-norm for image data[9]. Nevertheless, when applying adversarial training to network embedding, we still lack guidelines or guarantees to craft imperceptible adversarial examples. Existing methods generate adversarial examples via heuristically exerting L_2-norm to constrain their divergence from node representations in the latent space. However, such heuristic constraints cannot guarantee the imperceptibility of adversarial examples in the latent space, limiting the power of adversarial training. For network embedding, it is a particularly challenging problem to generate imperceptible adversarial examples in the latent space.

In this paper, we propose the Identity-Preserving Adversarial Training (IPAT) method with identity-preserving regularization to explicitly preserve the imperceptibility of adversarial examples. IPAT regards each node as a class consisting of itself and its adversarial examples, and for each node, IPAT encourages the generated adversarial examples to be classified into the same class together with the node itself. In other words, adversarial examples preserve the identity of the original node. As depicted in Fig.2(b), our method explicitly constrains each adversarial example inside the identity region of the original node, thus achieving the desired imperceptibility of adversarial examples. In contrast, existing methods may generate adversarial examples outside the identity region, resulting in inconsistent identity between the adversarial example and its original node (shown in Fig.2(a)). In brief, our identity-preserving regularization guarantees the imperceptibility of adversarial examples, thus achieving robust network embedding.

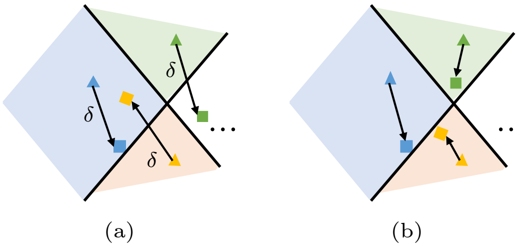


Fig.2. Comparison of (a) adversarial examples with heuristic L_2-norm constraint and (b) adversarial examples with identity-preserving regularization. The triangles represent the original nodes and the squares are adversarial examples. The lines represent the boundaries of node identity.

Our identity-preserving adversarial training is a general method, which can be applied to existing network embedding methods to improve their performance. Applying our IPAT to a representative network embedding method GAE[10], extensive experimental results on real-world datasets demonstrate that our method significantly improves the performance of network embedding on various downstream tasks. Moreover, we also attach our identity-preserving adversarial training on DGI[11] to further show the generality of our method.

The main contributions are summarized as follows.

- We propose a novel identity-preserving adversarial training method for network embedding, guaranteeing the imperceptibility of adversarial examples and achieving robust network embedding.

- Our method is applicable to enhance existing network embedding methods as a flexible module.

- Extensive experiments are conducted to evaluate the effectiveness of our method. Besides the benchmark datasets, we construct a new dataset with various downstream tasks to comprehensively evaluate the effectiveness of node representation.

2 Problem Formulation

In this section, we give the problem formulation of network embedding and adversarial training for network embedding.

2.1 Network Embedding

Network embedding aims to embed each node into a low-dimensional space, preserving the network structure or node attributes. Given a network (also known as graph) $G=(V,\boldsymbol{A},\boldsymbol{X})$, with node set $V=\{v_i\}_{i=1}^{N}$, adjacency matrix $\boldsymbol{A}\in\mathbb{R}^{N\times N}$, and attribute matrix $\boldsymbol{X}\in\mathbb{R}^{N\times F}$, network embedding aims to map nodes into low-dimensional node representation matrix $\boldsymbol{H}\in\mathbb{R}^{N\times D}$, where N denotes the number of nodes, F is the dimension of node attributes, and D is the dimension of node representations ($D << F$).

Existing methods learn node representations $\boldsymbol{H}$ by solving a pretext task[1], e.g., structure reconstruction[12, 13] or node attributes prediction[14, 15]. For simplicity, the pretext task in network embedding is generally formulated as an objective function $\mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ represents all the learnable parameters in network embedding. Despite the empirical success of network embedding, existing network embedding methods are vulnerable to random or adversarial perturbations[6, 7].

2.2 Adversarial Training for Network Embedding

Adversarial training is newly leveraged to improve the robustness of network embedding models against adversarial random or adversarial perturbations[6]. Specifically, adversarial training for network embedding consists of two iterative steps. First, a network embedding model is trained by minimizing the objective function of the pretext task $\mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta})$. Then, adversarial training for network embedding generates adversarial examples for each node by adding adversarial perturbations on node representation to maximize the loss function of network embedding, and use the obtained adversarial examples to train the network embedding model. The loss on the adversarial examples can be considered as a regularization term in the embedding learning process[6]. To sum up, adversarial training for network embedding is formulated as:

$$\begin{aligned}&\min_{\Theta}\quad \mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta})+\lambda\sum_{i=1}^{N}\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\text{adv}};\boldsymbol{\Theta}),\\&\text{s.t.}\quad \forall v_i\in V,\ \boldsymbol{\delta}_i^{\text{adv}}=\mathop{\arg\max}_{\boldsymbol{\delta}_i,\|\boldsymbol{\delta}_i\|_2\leqslant r}\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i;\boldsymbol{\Theta}),\end{aligned}\tag{1}$$

where $\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\text{adv}};\boldsymbol{\Theta})$ is the network embedding loss on adversarial examples of node v_i, λ is a hyperparameter to control the importance of the regularization term, $\boldsymbol{h}_i$ is the representation of node v_i (the i-th row of node representation matrix $\boldsymbol{H}$), and $\boldsymbol{\delta}_i^{\text{adv}}$ represents the optimal perturbation on node v_i. (1) indicates that the adversarial perturbation $\boldsymbol{\delta}_i^{\text{adv}}$ of each node v_i is obtained by maximizing the loss function, and r is the L_2-norm constraint.

However, heuristical L_2-norm constraint in existing methods is hard to guarantee the imperceptibility of adversarial examples, thus limiting the power of adversarial training. To solve this problem, we propose an identity-preserving adversarial training method which adopts explicit regularization to guarantee the imperceptibility of adversarial examples, anticipating to achieve robust network embedding.

3 Model

In this section, we firstly introduce an overview of our IPAT method. Then we present the detailed information of each component of our IPAT method.

3.1 Overview

Adversarial training for network embedding leverages the generated adversarial examples to retrain the network embedding model, anticipating to improve its robustness. An important premise behind the success of adversarial training is that adversarial examples should be imperceptible[8], however, which is hard to be achieved by existing methods. In this paper, we propose an identity-preserving adversarial training method IPAT to explicitly guarantee the imperceptibility of adversarial examples. Formally, IPAT works as:

$$\min_{\Theta}\ \mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta}) + \lambda_{\rm id}\sum_{i=1}^{N}\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\rm adv};\boldsymbol{\Theta}), \quad (2)$$

s.t. $\forall v_i \in V$,

$$\boldsymbol{\delta}_i^{\rm adv} = \mathop{\arg\max}_{\boldsymbol{\delta}_i,\, \|\boldsymbol{\delta}_i\|_2 \leqslant r_i,\, f_{\rm id}(\boldsymbol{h}_i+\boldsymbol{\delta}_i) = f_{\rm id}(\boldsymbol{h}_i)} \mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i;\boldsymbol{\Theta}), \quad (3)$$

where $\lambda_{\rm id}$ is a hyperparameter to control the importance of network embedding loss on adversarial examples $\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\rm adv};\boldsymbol{\Theta})$. These equations indicate the two iterative steps in adversarial training. Specifically, (3) indicates the first step that IPAT generates the adversarial examples $\boldsymbol{h}_i'$ by adding optimal perturbation $\boldsymbol{\delta}_i^{\rm adv}$ to the original representations $\boldsymbol{h}_i$, where the optimal perturbation $\boldsymbol{\delta}_i^{\rm adv}$ is learned by maximizing the loss function of network embedding. Instead of only leveraging the L_2-norm constraint as existing methods (shown in (1)), we propose additional identity-preserving regularization $f_{\rm id}(\boldsymbol{h}_i + \boldsymbol{\delta}_i) = f_{\rm id}(\boldsymbol{h}_i)$ to guarantee the imperceptibility of adversarial examples (shown as the bottom left in Fig.3). After obtaining the generated adversarial examples, IPAT leverages a mixture of adversarial examples and original examples to train the network embedding model (shown in (2)), thus improving the robustness of model.

Note that our identity-preserving adversarial training is a general method, which can be applied to enhance existing network embedding methods (shown as the basic model in the top half of Fig.3).

3.2 Identity-Preserving Adversarial Examples

In this subsection, we detailedly introduce how we generate the identity-preserving imperceptible adversarial examples according to (3).

3.2.1 *Adversarial Examples with Maximum Loss*

Generally speaking, adversarial examples are examples that significantly degrade the performance of network embedding models. Taking the loss function of the network embedding model as the indicator of the performance, we first generate the adversarial example $\boldsymbol{h}_i'$ for each node v_i via adding an adversarial perturbation $\boldsymbol{\delta}_i$ to the original representation $\boldsymbol{h}_i$ and maximizing the loss, i.e., $\max_{\boldsymbol{\delta}_i} \mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i;\boldsymbol{\Theta})$ as shown in (3). We adopt the widely used fast gradient descent method[6, 16] to obtain the optimal perturbation $\boldsymbol{\delta}_i^{\rm adv}$ of node v_i as follows:

$$\boldsymbol{\delta}_i^{\rm adv} = r_i \cdot \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2},$$

where $\boldsymbol{d}_i = \nabla_{\boldsymbol{h}_i}\mathcal{L}(G|\boldsymbol{h}_i;\boldsymbol{\Theta})$ is the gradient of the network embedding loss function with respect to the representation $\boldsymbol{h}_i$ of node v_i, and r_i is the norm con-

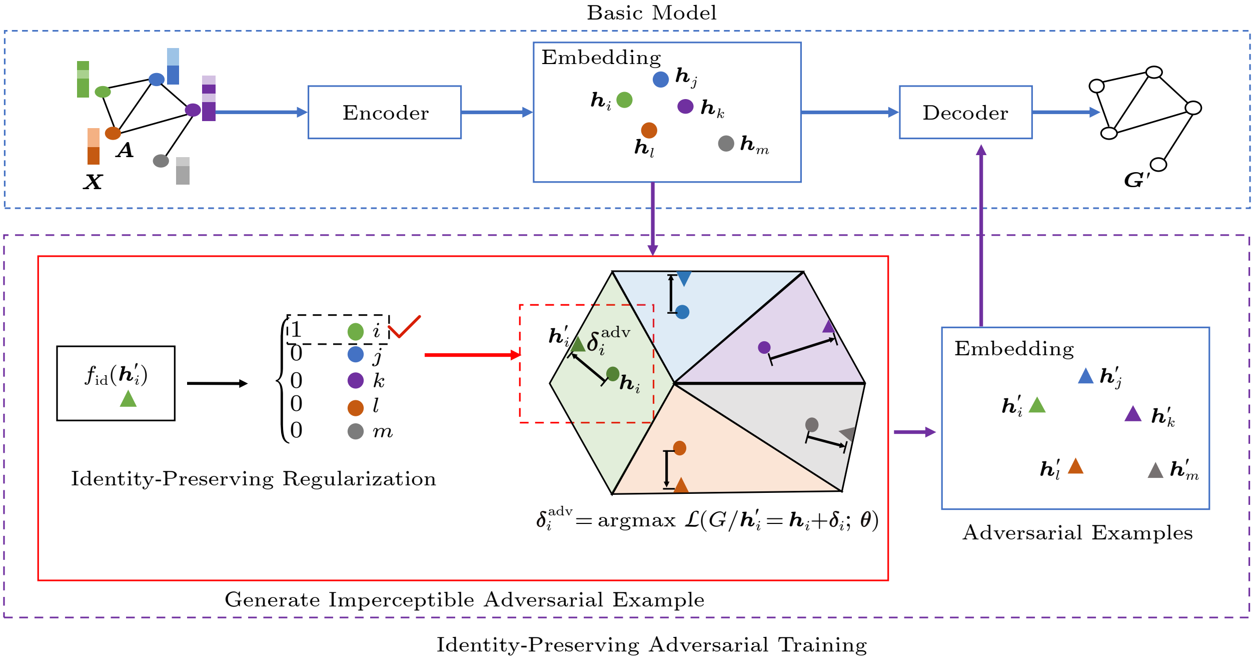


Fig.3. Overall framework of network embedding with the proposed IPAT method.

straint of adversarial perturbation. Accordingly, the generated adversarial example is denoted as:

$$\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\text{adv}} = \boldsymbol{h}_i + r_i \cdot \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2}. \tag{4}$$

Existing methods[6] heuristically adopt a pre-defined norm constraint r for all nodes when generating adversarial examples, which is hard to achieve the imperceptibility of adversarial examples. Different from existing methods, in this paper, we propose novel identity-preserving regularization to learn a node-specific norm constraint r_i for each node v_i to guarantee the imperceptibility of adversarial examples.

3.2.2 Identity-Preserving Regularization

To guarantee the imperceptibility of adversarial examples, we propose a novel perspective that an imperceptible adversarial example should retain the same identity with its original node, and propose identity-preserving regularization to achieve such a constraint. Specifically, we formalize identity-preserving regularization as a multi-class classification problem and treat the identity of each node as a class label. Then we encourage each generated adversarial example to be classified into the same class as its original node (bottom left in Fig.3).

Unfortunately, due to the huge label space of node identities, exactly calculating the probability that an adversarial example belongs to each class is costly. To tackle this challenge, we utilize the negative sampling to efficiently approximate the probability. Specifically, we implement identity-preserving regularization by distinguishing the pair of representations of an adversarial example and its original node representation (positive pairs) from the pair of representations of an adversarial example and the representation of other nodes (negative pairs). The objective function is formalized as follows:

$$\mathcal{L}_{\text{id}} = \sum_{i=1}^{N} \log \sigma(\boldsymbol{h}_i'^{\text{T}} \boldsymbol{h}_i) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim p(v)} \log \sigma(-\boldsymbol{h}_i'^{\text{T}} \boldsymbol{h}_k), \tag{5}$$

where $\boldsymbol{h}_i'$ denotes the representation of the adversarial example of node v_i, and $\boldsymbol{h}_i$ is the original representation of node v_i. For each node, we uniformly sample K other nodes in the graph to build negative pairs, where the probability of a node being sampled is $p(v) = 1/N$, and h_k is the representation of the sampled node v_k. σ is the sigmoid activation function. Substituting (4) into (5), we get the following loss function of identity-preserving regularization:

$$\mathcal{L}_{\text{id}} = \sum_{i=1}^{N} \log \sigma \left(\boldsymbol{h}_i^{\text{T}} \boldsymbol{h}_i + r_i \left(\frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|} \right)^{\text{T}} \boldsymbol{h}_i \right) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim p(v)} \log \sigma \left(-\boldsymbol{h}_i^{\text{T}} \boldsymbol{h}_k - r_i \left(\frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|} \right)^{\text{T}} \boldsymbol{h}_k \right). \tag{6}$$

To achieve the above loss function, we parameterize the norm constraint r_i as a learnable and node specific variable rather than a heuristical pre-defined constant as existing methods. The formula is as follows:

$$r_i = f_r(\boldsymbol{h}_i; \boldsymbol{\Theta}_r),$$

where f_r is a two-layer perceptron with the standard "ReLU" as an activation function, and $\boldsymbol{\Theta}_r$ are all parameters including weight matrixes and bias vectors. That is to say, we can optimize the parameters $\boldsymbol{\Theta}_r$ to find the optimal norm constraint r_i for each node to minimize (6); thus we rewrite (6) as $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$.

Note that, $\boldsymbol{\Theta}_r = \boldsymbol{0}$, i.e., the generated adversarial example is equal to its original node representation with $r_i = 0$, is a trivial solution to minimize identity-preserving regularization $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$, which cannot bring any new information beneficial to the training of the network embedding model. To avoid such trivial solutions, we encourage the norm constraint r_i of each node to be as large as possible, which is formalized as $\mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r) = -\sum_{i \in N} r_i^2$.

To sum up, the overall loss of our identity-preserving regularization is:

$$\mathcal{L}_{\text{ipr}}(\boldsymbol{H}; \boldsymbol{\Theta}_r) = \mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r) + \lambda_n \mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r),$$

where λ_n controls the importance of loss $\mathcal{L}_n$.

Through optimizing the above identity-preserving regularization, i.e.,

$$\boldsymbol{\Theta}_r^{\text{opt}} = \arg\min_{\boldsymbol{\Theta}_r} \mathcal{L}_{\text{ipr}}(\boldsymbol{H}; \boldsymbol{\Theta}_r),$$

we obtain the optimal norm constraint r_i^{opt} for each node, i.e., $r_i^{\text{opt}} = f_r(\boldsymbol{h}_i; \boldsymbol{\Theta}_r^{\text{opt}})$. Accordingly, the finally generated identity-preserving adversarial examples for each node v_i are:

$$\boldsymbol{h}_i' = \boldsymbol{h}_i + f_r(\boldsymbol{h}_i; \boldsymbol{\Theta}_r^{\text{opt}}) \cdot \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2},$$

where $\boldsymbol{d}_i = \nabla_{\boldsymbol{h}_i} \mathcal{L}(G|\boldsymbol{h}_i; \boldsymbol{\Theta})$ is the gradient of the network embedding loss, and $\boldsymbol{\Theta}_r^{\text{opt}}$ is the optimal parameters with respect to our proposed identity-preserving regularization. Note that, the generated adversarial examples can significantly degrade the performance of the network embedding model while preserving the important premise of imperceptibility from the perspective of identity-preserving.

After generating the identity-preserving adversarial examples, we add them back to retrain the model, thus improving the robustness of the network embedding model. The above two steps are iteratively conducted until convergence.

3.3 Training Algorithm and Time Complexity

In this subsection, we give the entire adversarial training procedure of our proposed IPAT method for network embedding (Algorithm 1) and provide an analysis of its time complexity.

Algorithm 1. Identity-Preserving Adversarial Training Algorithm

```
Input: graph$(\boldsymbol{A}, \boldsymbol{X})$, max epochs $E$, hyperparameters of basic model
Output: embedding matrix $\boldsymbol{H}$
1: Initialize network embedding model parameters $\boldsymbol{\Theta}$
   and identity-preserving parameters $\boldsymbol{\Theta}_r$;
2: for i = 1 to E do
3:     $\boldsymbol{d}_i = \nabla_{\boldsymbol{h}_i}\mathcal{L}(G|\boldsymbol{h}_i;\boldsymbol{\Theta})$ for each node $v_i$;
4:     $\boldsymbol{\Theta}_r^{\text{opt}} = \arg\min_{\boldsymbol{\Theta}_r} \mathcal{L}_{\text{ipr}}(\boldsymbol{H};\boldsymbol{\Theta}_r)$;
5:     $r_i = f_r(\boldsymbol{h}_i;\boldsymbol{\Theta}_r^{\text{opt}})$ for each node $v_i$;
6:     $\boldsymbol{h}_i' = \boldsymbol{h}_i + r_i \cdot \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2}$ for each node $v_i$;
7:     Update network embedding model parameters $\boldsymbol{\Theta}$
       with the generated adversarial examples according
       to minimizing $\mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta}) + \lambda\sum_{i=1}^{N}\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + r_i \cdot \frac{\boldsymbol{d}_i}{\|\boldsymbol{d}_i\|_2};\boldsymbol{\Theta})$;
8: end for
9: return $\boldsymbol{H}$;
```

In line 1, we initialize network embedding model parameters $\boldsymbol{\Theta}$ and identity-preserving parameters $\boldsymbol{\Theta}_r$. For each epoch, we first generate the identity-preserving adversarial examples through steps from line 3 to line 6, and then leverage them to update the parameters of network embedding in line 7. For generating the identity-preserving adversarial examples, we first calculate the gradient of the network embedding loss $\boldsymbol{d}_i = \nabla_{\boldsymbol{h}_i}\mathcal{L}(G|\boldsymbol{h}_i;\boldsymbol{\Theta})$ for each node to ensure that the generated adversarial examples maximize the loss function of network embedding in line 3. Then we obtain the optimized $\boldsymbol{\Theta}_r^{\text{opt}}$ by minimizing identity-preserving regularization $\mathcal{L}_{\text{ipr}}(\boldsymbol{H};\boldsymbol{\Theta}_r)$, which is used to guarantee the imperceptibility of adversarial examples in line 4. Line 5 shows that we obtain the suitable norm constraint $r_i = f_r(\boldsymbol{h}_i;\boldsymbol{\Theta}_r^{\text{opt}})$ for each node according to the optimized $\boldsymbol{\Theta}_r^{\text{opt}}$. Then the adversarial examples $\boldsymbol{h}_i' = \boldsymbol{h}_i + r_i \cdot (\boldsymbol{d}_i/\|\boldsymbol{d}_i\|_2)$ are generated for each node v_i in line 6. Finally, all the parameters of the network embedding model $\boldsymbol{\Theta}$ are updated by minimizing the loss function (2) with the generated adversarial examples as shown in line 7. After E rounds of iterations, we output the final node representation matrix $\boldsymbol{H}$.

Compared with the baseline AdvT, our method only introduces an extra time cost in line 4. The time complexity of calculating (5) is $O(NKD^2)$, where N is the number of nodes, K is the number of negative examples used in (5), and D is the dimension of node representations. At the same time, we only update the parameters $\boldsymbol{\Theta}_r$ once according to the gradients in each iteration, where the time complexity is also $O(NKD^2)$. Therefore, the newly increased computational time of our IPAT method is linearly related to the number of nodes.

4 Experiments

In this section, we conduct various experiments to demonstrate the effectiveness of our proposed identity-preserving adversarial training for network embedding.

4.1 Experimental Setup

We first introduce the setup of experiments including datasets, baselines, and implementation details of our IPAT method.

4.1.1 *Benchmark Datasets*

We conduct experiments on four widely used benchmark datasets. The statistics of all the datasets is collected in Table 1. The benchmark datasets include three citation networks[11, 17, 18], i.e., Cora, Citeseer, and Pubmed, where nodes are articles and edges indicate citations between articles, and a web network Wiki[6, 19], where nodes are web pages and edges represent the hyperlinks between web pages.

Table 1. Datasets Description

Dataset	#Nodes	#Edges	#Labels
Cora[11, 17, 18]	2 708	5 429	7
Citeseer[11, 17, 18]	3 327	4 732	6
Pubmed[11, 17, 18]	19 717	44 338	3
Wiki[6, 19]	2 405	17 981	17

Note: #: number of.

4.1.2 Baselines

We compare our method with six representative network embedding methods, i.e., SVD[20], DeepWalk[5], DANE[15], DGI[11], GraphCL[21], and GAE[10], and two adversarial learning regularization methods, i.e., ARGE[22] and AdvT-GAE[6], designed to improve the robustness of the network embedding model. The descriptions of the baselines are as follows.

- SVD [20] learns node representations by decomposing the node attribute matrix. We reduce the dimension of node attributes to 200 via singular value decomposition (SVD), following the settings in [23].
- DeepWalk [5] aims to learn node representations that preserve the cooccurrence in random walks.
- DANE[15] learns node representations by two independent auto-encoders to model attributes and structure information, respectively.
- DGI [11] learns node representations through the pretext task of maximizing mutual information between patch representations and corresponding graph representations.
- GraphCL[21] learns node representations by maximizing mutual information between the hidden representations of two augmentations of the origin graph.
- GAE [10] gets node representations by the graph convolution layer and utilizes network reconstruction as the pretext task.
- ARGE [22] adds adversarially regularization on GAE, which improves the robustness of GAE by imposing a prior distribution on node representations.
- AdvT-GAE[6] represents adversarial training for GAE, which is the current strongest baseline to improve the robustness of the network embedding model. We follow the settings of AdvT[6] to build adversarial examples.

4.1.3 Implementation Details

In this paper, both IPAT and the two adversarial learning regularization baselines adopt the same basic model (GAE) for a fair comparison. We denote our identity-preserving adversarial training for GAE as IPAT-GAE.

To better reproduce the experimental results, we specify the model architecture and loss function of GAE, and give the parameter settings of our IPAT-GAE. GAE stacks two layers of GCN[17] as the encoder to get the node representations $\boldsymbol{H}$. GAE aims to learn node representation by reconstructing the network structure, and the loss function $\mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta})$ is formalized as:

$$\mathcal{L}(G|\boldsymbol{H};\boldsymbol{\Theta}) = \sum_{i\in N}\sum_{j\in N} a_{ij}\log(\sigma(\boldsymbol{h}_i\boldsymbol{h}_j^{\mathrm{T}}))+ \\ (1-a_{ij})\log(1-\sigma(\boldsymbol{h}_i\boldsymbol{h}_j^{\mathrm{T}})),$$

where $a_{ij}=1$, if there is an edge between node v_i and node v_j; otherwise $a_{ij}=0$. After getting the adversarial examples, we add them back to train GAE, and thus $\sum_{i\in N}\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\mathrm{adv}};\boldsymbol{\Theta})$ in (2) is formalized as:

$$\begin{aligned} &\sum_{i\in N}\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\mathrm{adv}};\boldsymbol{\Theta}) \\ =&\sum_{i\in N}\sum_{j\in N} a_{ij}\log(\sigma((\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\mathrm{adv}})\boldsymbol{h}_j^{\mathrm{T}}))+ \\ &(1-a_{ij})\log(1-\sigma((\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\mathrm{adv}})\boldsymbol{h}_j^{\mathrm{T}})). \end{aligned} \quad (7)$$

We set the number of hidden units of GCN in our method IPAT to be 256 and 128, the dropout ratio equals 0.5, and the negative number $K=5$. For the sake of efficiency, we combine all the loss functions together and update the parameters simultaneously, with $\lambda_{\mathrm{id}}=1$, $\lambda_n=1$ as the weights of $\mathcal{L}(G|\boldsymbol{h}'_i=\boldsymbol{h}_i+\boldsymbol{\delta}_i^{\mathrm{adv}};\boldsymbol{\Theta})$ and $\mathcal{L}_n(\boldsymbol{H};\boldsymbol{\Theta}_r)$, respectively. We implement our method in Tensorflow2.3[24]. Moreover, we release all the datasets mentioned in this paper and the source code of our method①, and we also upload a Douban dataset on ScienceDB②.

4.2 Link Prediction

We include link prediction as one of the downstream tasks to evaluate the generalization of the learned node representations. We set the edges in the network as positive instances and split them according to the ratio of 85%, 5%, and 10% for training, validation, and test, respectively. Moreover, we learn the node representations only with the edges in the training set. In the test phase, we randomly select some unconnected node pairs as negative samples to keep the ratio of positive samples and negative samples as 1 : 1. We calculate the link probability via the inner product of node embedding on the test data. Following [10], we adopt area under the curve (AUC) and

①https://github.com/CKTProject/IPAT, Oct. 2022.

②DOI: 10.57760/sciencedb.03252, Oct. 2022.

average precision (AP) as the evaluation metrics.

The experimental results of link prediction are summarized in Table 2, where the best results are bolded, and the second best results are underlined. We observe that adversarial learning regularization baselines ARGE, AdvT-GAE, and our IPAT-GAE are consistently better than GAE, which shows that the generalization of node representation on downstream tasks can be improved by adversarial training for network embedding. Note that IPAT-GAE is consistently better than ARGE and AdvT-GAE in all three datasets. Specifically, IPAT-GAE achieves 4.95%, 4.82%, 1.07%, and 2.35% improvement over AdvT-GAE on AUC in Cora, Citeseer, Pubmed, and Wiki, respectively. The results show that our IPAT method can further improve the robustness and generalization of node embeddings by guaranteeing the imperceptibility of adversarial examples.

4.3 Node Classification

In this subsection, we conduct experiments on node classification to demonstrate the effectiveness of IPAT. After learning the embedding for each node, an SVM[25] with a linear kernel is used to classify nodes. We use the SVM package provided by scikit-learn[26] with default parameters. Moreover, we repeat this process 10 times and show the mean results. In this subsection, we conduct node classification tasks with two different strategies of data partitioning to comprehensively verify the effectiveness of our method.

4.3.1 *Node Classification with Different Label Ratios*

Following the settings in [15, 23], we randomly sample a certain number of nodes with labels as training data and the rest as the test. Moreover, the test data ratios vary in {10%, 30%, 50%}.

Table 3 shows that ARGE, AdvT-GAE, and IPAT-GAE significantly improve the accuracy of the node classification task on Cora, Citeseer, and Pubmed compared with GAE. These results demonstrate that adversarial training for network embedding also improves the generalization ability of node representations on the node classification task. In addition, IPAT-GAE consistently outperforms AdvT-GAE on the four datasets, suggesting that our IPAT method further improves the generalization ability of node representations by generating identity-preserving adversarial examples.

Table 2. Comparison on AUC(%) and AP(%) of Link Prediction Task

Method	Cora		Citeseer		Pubmed		Wiki	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
SVD[20]	79.10	82.46	85.82	88.76	85.98	87.60	83.70	87.98
DeepWalk[5]	80.53	82.79	73.22	76.21	76.88	74.73	81.27	82.39
DANE[15]	88.19	89.56	84.93	84.68	87.76	89.69	91.01	92.45
DGI[11]	78.53	80.50	81.31	82.50	81.66	83.13	78.18	79.30
GraphCL[21]	82.35	83.51	83.72	85.63	88.37	89.02	83.52	85.01
GAE[10]	91.47	92.37	90.52	91.59	95.93	96.25	91.81	92.91
ARGE[22]	92.40	93.20	91.96	92.97	96.80	97.10	92.90	93.80
AdvT-GAE[6]	91.73	92.74	91.47	92.72	96.05	96.42	91.89	92.61
IPAT-GAE	**96.68**	**97.36**	**96.29**	**96.42**	**97.12**	**97.33**	**94.24**	**95.10**

Table 3. Comparison on Micro-$F1$ (%) of Node Classification with Different Label Ratios on Four Benchmark Datasets

Method	Cora			Citeseer			Pubmed			Wiki		
	10%	30%	50%	10%	30%	50%	10%	30%	50%	10%	30%	50%
SVD[20]	47.44	66.70	68.55	61.14	66.95	69.39	64.93	78.05	81.12	65.32	75.59	77.61
DeepWalk[5]	76.53	79.86	81.33	50.15	56.24	56.83	79.87	80.99	81.35	57.95	65.34	67.15
DANE[15]	78.01	81.86	82.50	63.64	68.69	72.30	84.28	85.44	86.56	72.98	77.26	78.30
DGI[11]	82.53	84.35	84.77	66.39	69.37	71.53	84.73	85.36	85.78	70.58	71.86	73.17
GraphCL[21]	81.87	83.76	84.05	64.72	67.18	69.01	82.00	82.83	83.17	68.43	70.31	71.65
GAE[10]	80.39	82.22	82.35	60.52	60.59	62.00	83.91	84.38	84.33	68.82	70.60	72.48
ARGE[22]	81.32	84.87	85.92	63.27	65.26	66.73	84.02	85.64	86.17	70.92	72.56	74.31
AdvT-GAE[6]	82.31	85.70	86.79	67.91	71.91	73.46	85.28	86.21	86.52	72.15	74.21	76.51
IPAT-GAE	**84.85**	**86.72**	**87.92**	**70.65**	**73.57**	**75.05**	**85.36**	**86.42**	**86.73**	**73.54**	76.71	**78.68**

4.3.2 Node Classification with Extremely Limited Labels

In practice, graph data is often profoundly lacking in labels. To illustrate the effectiveness of the representations learned by IPAT-GAE in this scenario, we also conduct the downstream node classification tasks with extremely limited labels. Following the data partition as [17, 18, 27], each class only has 20 labeled data in the training stage in this task.

As shown in Table 4, IPAT-GAE consistently gets higher accuracy than GAE, ARGE, and AdvT-GAE on all the datasets. Specially, IPAT-GAE achieves 2.22%, 2.74%, and 4.72% over AdvT-GAE in Cora, Citeseer, and Pubmed, respectively. Such significant improvement reflects that IPAT greatly improves the generalization of learned node representations in downstream tasks even with limited labels.

Table 4. Comparison on Accuracy(%) of the Node Classification Task with Extremely Limited Labels

Method	Cora	Citeseer	Pubmed
SVD[20]	55.10	46.50	71.40
DeepWalk[5]	67.20	43.20	65.30
DANE[15]	78.20	63.70	75.80
DGI[11]	82.30	71.80	76.80
GraphCL[21]	81.76	71.22	76.15
GAE[10]	80.20	65.80	72.10
ARGAE[22]	80.80	66.80	72.20
AdvT-GAE[6]	81.30	69.82	73.40
IPAT-GAE	83.52	72.56	78.12

4.4 Node Classification on a Large Dataset

In this subsection, we conduct the node classification task on a widely used large dataset ogbn-arxiv[3] with 169 343 nodes and 1 166 243 edges[28]. Each node in ogbn-arxiv is a paper and each edge represents a citation between papers. We follow the data partition as [28], i.e., leveraging papers published until 2017 as the training set, those published in 2018 as the validation set, and those published since 2019 as the test set.

As shown in Table 5, our IPAT-GAE is better than GAE, ARGE, and AdvT-GAE on the ogbn-arxiv dataset. Our IPAT-GAE achieves 0.52% and 0.62% over AdvT-GAE in validation and test, respectively. The results show that our IPAT improves the generalization of learned node representations on the ogbn-arxiv dataset.

Table 5. Comparison on Accuracy (%) of the Node Classification Task on Dataset ogbn-arxiv

Method	Validation	Test
SVD[20]	52.65	50.20
DeepWalk[5]	64.29	63.14
DANE[15]	65.35	65.07
DGI[11]	71.07	70.34
GraphCL[21]	OOM	OOM
GAE[10]	67.46	66.32
ARGAE[22]	67.92	66.58
AdvT-GAE[6]	67.97	67.02
IPAT-GAE	68.49	67.64

Note: OOM: out of memory.

4.5 Multiple Downstream Tasks

Considering the benchmark datasets only have one node classification downstream task, we construct a new dataset Douban[4] with multiple downstream tasks to better verify the generalization ability of node representations on multiple tasks.

4.5.1 Dataset

Our new network dataset is crawled from Douban Movies[5], which is a website providing users comments on movies. Each node in the network represents a movie, and each edge represents that the movies on both ends of it are co-preferenced by audiences, which is provided by Douban. The network contains 31 761 nodes and 179 924 edges. We use the movie profiles to form the attributes of the node. Firstly, we use "jieba"[6], a widely used Chinese word segmentation tool, to segment movie profiles and filter common stop words and words that appear less than three times in the corpus. Then, we build a TF-IDF vector for each movie using scikit-learn[26] and reduce the dimension to 500 via SVD[20].

We build three downstream tasks for this Douban dataset, including movie genres prediction, rating score level prediction, and popularity level prediction. Genre predicting task is a multi-classification task, we directly use the genres of the movie provided by Douban as the label, and each movie has at least one genre.

[3] https://ogb.stanford.edu/, Oct. 2022.

[4] DOI: 10.57760/sciencedb.03252, Oct. 2022.

[5] https://movie.douban.com, Oct. 2022.

[6] https://github.com/fxsjy/jieba, Oct. 2022.

To build the label of the rating score prediction, we rank movies by rating scores, and divide them into 10 classes of the same size. Similarly, we rank all the movies according to the number of comments, and divide them into three classes of the same size. For each task, we randomly sample 70% nodes as the training set, 10% as the validation set, and the rest as the test set.

4.5.2 Experiments on Douban

We compare our IPAT-GAE with the basic network embedding model GAE[10] and two adversarial learning regularization methods, ARGE[22] and AdvT-GAE[6], to focus on verifying the improvements that our IPAT method brings to the network embedding model. We also introduce a supervised GCN model as a baseline to show a possible upper bound on the results of these tasks. The supervised GCN model has the same hidden units as IPAT-GAE.

As shown in Table 6, IPAT-GAE is consistently better than other unsupervised node representation learning baselines on all the tasks, and gets comparable results with the supervised GCN model, which demonstrates that our IPAT method improves the generalization of the learned node representations on different downstream tasks.

4.6 Adversarial Examples Visualization

To intuitively evaluate the imperceptibility of adversarial examples, we visualize the representations of the adversarial examples. Firstly, we randomly choose five nodes (the squares in Fig.4), and obtain their adversarial examples (the triangles in Fig.4). Then, we transfer their representations into two 2D spaces using t-SNE[29], and different colors represent different nodes. The results of AdvT-GAE and IPAT-GAE on Cora are shown in Fig.4(a) and Fig.4(b), respectively.

In Fig.4(a), we find that the distance of an adversarial example from its corresponding node is significantly greater than its distance from some other nodes on the graph, e.g., the distance of the red triangle to the red square is significantly larger than its distance to the blue square. This phenomenon reflects that the adversarial examples learned by AdvT-GAE are not imperceptible, and adding these adversarial examples to retrain the network embedding may hurt its performance. While in Fig.4(b), the adversarial examples generated by IPAT-GAE are closer to their original nodes than to other nodes on the graph. The results show that our IPAT with explicit identity-preserving regularization better guarantees the imperceptibility of generated adversarial examples.

4.7 Robustness of Network Embedding Model

In this subsection, we set up an experiment to examine the robustness of network embedding models against randomly adding noisy edges. Firstly, we randomly link some unconnected node pairs as the noisy edges. Specifically, the number of noisy edges is r% of existing edges, and r% varies from 10% to 100%. Then we learn the node representations from these noisy graphs. Lastly, we evaluate the effectiveness of the learned node representations on the above downstream tasks. We show the results of the task mentioned in Subsection 4.3.2 on the Cora and Citeseer datasets as examples.

We notice that IPAT-GAE shows larger improvements than GAE as the noise edge increases on Cora (Fig.5) and Citeseer (Fig.6), which demonstrates that our IPAT method improves the robustness of the network embedding model. Fig.5 and Fig.6 show that our IPAT-GAE consistently outperforms AdvT-GAE. Such a phenomenon proves that maintaining the imperceptibility of adversarial examples in adversarial training is critical for improving the robustness of network embedding model.

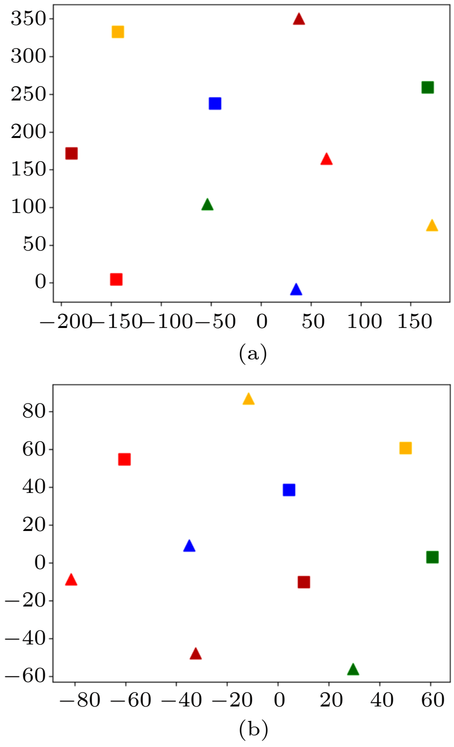
4.8 Ablation Study

In this subsection, we evaluate the effectiveness of components in our IPAT method through ablation

Table 6. Comparison on Accuracy (%) of Three Downstream Tasks on Our Douban Dataset

Method Description	Method	Movie Genre	Rating Score Level	Popularity Level
Supervised	GCN[17]	**73.94** ± 0.3	**33.83** ± 0.6	**67.66** ± 0.5
Unsupervised	GAE[10]	66.76 ± 0.4	21.90 ± 0.4	57.12 ± 0.4
	ARGE[22]	67.36 ± 0.3	22.27 ± 0.4	58.32 ± 0.3
	AdvT-GAE[6]	69.29 ± 0.3	24.14 ± 0.2	59.87 ± 0.2
	IPAT-GAE	**72.07** ± 0.3	**33.14** ± 0.5	**67.22** ± 0.4

Note: The optimal results are in bold.



(a)

(b)

Fig.4. 2D visualization of five randomly chosen nodes and their adversarial examples. (a) Adversarial examples generated by AdvT-GAE. (b) Adversarial examples generated by IPAT-GAE.

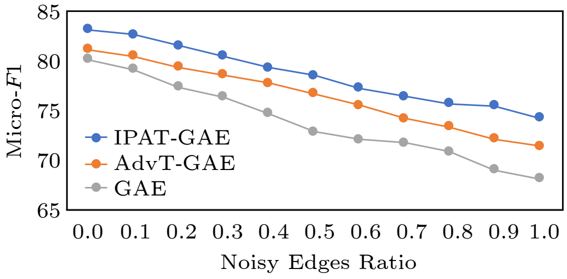


Fig.5. Robustness of the network embedding against randomly adding noisy edges on Cora.

studies.

$\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\text{adv}}; \boldsymbol{\Theta})$ encourages IPAT to leverage the generated identity-preserving adversarial examples to retrain GAE. As shown in Table 7, without identity-preserving regularization (IPAT-GAE w/o $\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\text{adv}}; \boldsymbol{\Theta})$ in Table 7) the results drop obviously, and become only comparable to the results of our basic model GAE, which demonstrates that adding the generated identity-preserving adversarial examples to retrain the network embedding, improves the performance of the learned node representations.

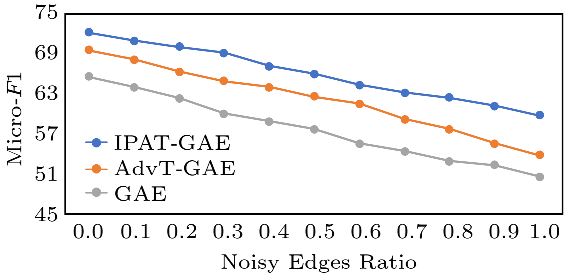


Fig.6. Robustness of the network embedding against randomly adding noisy edges on Citeseer.

Identity-preserving regularization loss $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$ encourages the adversarial examples to be classified into the same identity categories as their origin node examples, which encourages adversarial examples to be imperceptible. To verify the effectiveness of identity-preserving regularization loss $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$, we remove this loss (as IPAT-GAE w/o $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$ in Table 7) and compare it with IPAT-GAE. As shown in Table 7, without identity-preserving regularization, the performance of the learned node representations drops obviously, which demonstrates that our identity-preserving regularization is crucial to improving the effectiveness of adversarial training.

$\mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r)$ prevents the model from converging to a trivial solution $r_i = 0$. To test its effectiveness, we remove $\mathcal{L}_n$ (IPAT-GAE w/o $\mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r)$ in Table 7) and compare it with IPAT-GAE. We find that removing this loss degrades the performance of node embeddings, which demonstrates the effectiveness of our $\mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r)$.

4.9 Hyperparameter Analysis

In this subsection, we evaluate the hyperparameters of our model.

Table 7. Ablation Study: Comparing the Accuracy (%) of Node Classification Task in IPAT

Method	Cora	Citeseer	Pubmed
GAE[10]	80.20 ± 0.4	65.80 ± 1.5	72.10 ± 0.5
IPAT-GAE	$\mathbf{83.56} \pm 0.2$	$\mathbf{72.56} \pm 0.4$	$\mathbf{78.10} \pm 0.5$
IPAT-GAE w/o $\mathcal{L}(G\|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\text{adv}}; \boldsymbol{\Theta})$	80.26 ± 0.3	66.82 ± 0.4	73.19 ± 0.5
IPAT-GAE w/o $\mathcal{L}_{\text{id}}(\boldsymbol{H}; \boldsymbol{\Theta}_r)$	79.38 ± 0.4	68.63 ± 0.5	71.09 ± 0.6
IPAT-GAE w/o $\mathcal{L}_n(\boldsymbol{H}; \boldsymbol{\Theta}_r)$	83.17 ± 0.2	71.69 ± 0.5	77.55 ± 0.4

Note: The optimal results are in bold.

We first investigate the influence of weight λ_{id} of adversarial training loss $\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\mathrm{adv}}; \boldsymbol{\Theta})$, and the results are shown in Fig.7. We find that the performance on the downstream task is better as the weight λ_{id} increases at first, suggesting that adding our identity-preserving adversarial examples to retrain the network embedding model helps improve the robustness of the network embedding model, thus improving the generalization of node representations. Moreover, as λ_{id} becomes larger, the adversarial sample plays a larger weight than the original sample, resulting in a decrease in the performance on the downstream task, which shows that the most reasonable approach is to treat the adversarial sample and the original sample equally.

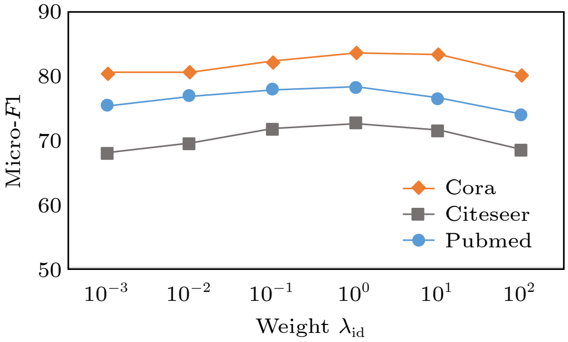


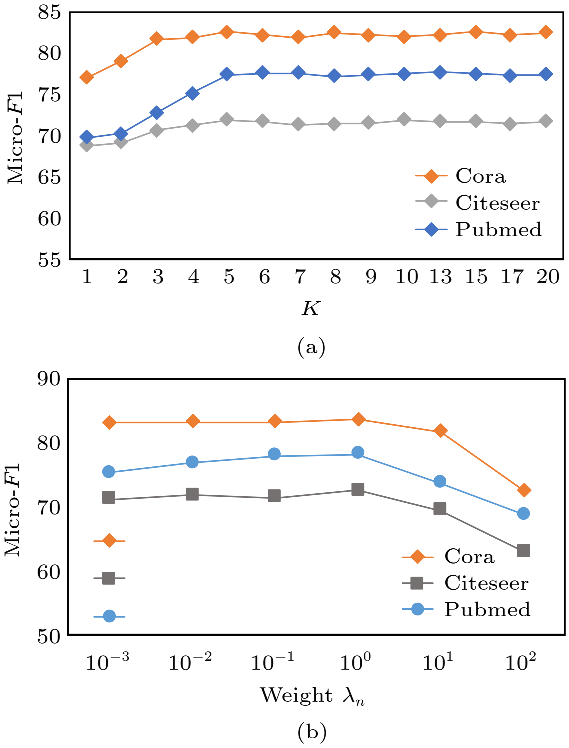
Fig.7. Impact of weight λ_{id} of adversarial training loss $\mathcal{L}(G|\boldsymbol{h}_i' = \boldsymbol{h}_i + \boldsymbol{\delta}_i^{\mathrm{adv}}; \boldsymbol{\Theta})$ on node classification.

For a comprehensive analysis of our proposed identity-preserving regularization loss, we evaluate the influence of two hyperparameters of it, including the number of negative samples K and weight λ_n of $\mathcal{L}_n(\boldsymbol{H};\boldsymbol{\Theta}_r)$. Fig.8(a) illustrates how K affects the node classification performances, where K varies from 1 to 20, and the corresponding micro-$F1$ values are plotted. We find that with the increment of negative examples, the result increases first, and then stabilizes around $K = 5$. Therefore, for a new dataset, the optimal solution can be found around $K = 5$.

The impact of λ_n is found in Fig.8(b). We find too large weight of λ_n will decrease the performance of our model. The reason can be that too large r causes the generated adversarial examples to be too far from the original samples, violating the premise that the adversarial examples should be imperceptible, resulting in hurting the model performance.

4.10 Flexibility of Identity-Preserving Adversarial Training

To verify the flexibility of our identity-preserving adversarial training, we also take DGI a recently state-of-the-art node representation method as the basic model. IPAT-DGI denotes adding our identity-preserving adversarial training method on DGI. AdvT-DGI represents adding AdvT regularization[6] on the DGI model as the baseline. We keep the same architecture and hyper-parameters of DGI as in [11].



(a)

(b)

Fig.8. Impact of identity-preserving regularization loss. (a) Impact of the number of negative samples K in identity-preserving loss $\mathcal{L}_{\mathrm{id}}(\boldsymbol{H};\boldsymbol{\Theta}_r)$. (b) Impact of weight λ_n of loss $\mathcal{L}_n(\boldsymbol{H};\boldsymbol{\Theta}_r)$ on node classification.

The results on node classification tasks with extremity-limited labels are shown in Table 8. IPAT-DGI outperforms DGI and AdvT-DGI, which demonstrates the flexibility of our identity-preserving adversarial training method.

Table 8. Comparing the Accuracy (%) of Node Classification Task using DGI as the Basic Model

Method	Cora	Citeseer	Pubmed
DGI[11]	82.3 ± 0.6	71.80 ± 0.7	76.8 ± 0.6
AdvT-DGI	83.0 ± 0.6	70.22 ± 1.0	75.3 ± 0.2
IPAT-DGI	83.7 ± 0.2	72.60 ± 0.4	77.6 ± 0.5

5 Related Work

In this section, we briefly review the related work on network embedding and adversarial training.

5.1 Network Embedding

Most conventional network embedding methods learn node representations by encouraging them to preserve structure proximity[12, 13, 15, 30], and various other properties such as node attributes[4, 14]. GAE[10] keeps the "first-order" proximity by reconstructing the network structure using hidden representations encoded by GCN. GPT-GNN[4] learns node representations by predicting masked node attributes. Recently, learning graph representations under a contrastive framework[31] has drawn intensive attention[21, 32, 33]. DGI[11] learns node representation by contrasting local node representations with global graph representations. CMVG[33] expands this idea and learns node representations and graph representations by contrasting multi-structural views of graphs.

However, previous methods are vulnerable to random or adversarial perturbations[6, 7]. To overcome the above issues, existing methods introduce either generative adversarial networks (GANs)[34, 35] or adversarial training to regularize the embedding learning process[6]. The former methods, e.g., ANE[7], ARGE[22], and NetRA[36], try to improve the generalization ability via generative adversarial networks (GANs) based regularization.

5.2 Adversarial Training

Adversarial training aims to find worst-case perturbations that maximize the current model loss, and encourages the model to correctly classify both unmodified examples and adversarial examples. Such a paradigm improves the robustness of the model and achieves better generalization performance on learning tasks. Moreover, such a paradigm has achieved success in many different scenarios, e.g., computer vision[37], natural language processing[38], and recommendation[22].

Recently, some studies have explored to apply adversarial training on graph data[39, 40]. Different from conventional adversarial training methods, which treat each example as independent of others, some adversarial training methods on graph data leverage neighboring nodes to model the impact[39, 40]. However, these methods rely on supervised labels and cannot be directly used for network embedding.

To solve this problem, Dai *et al.*[6] adapted the adversarial training (AdvT)[9, 16] for unsupervised node representation learning to improve robustness of network embedding and achieve better generalization performance. They proposed an adversarial training DeepWalk model (Dwns_AdvT), which generates adversarial examples in the embedding space and obtains adversarial perturbation with the fast gradient method[6, 16, 38]. Unlike our model, it ignores the guarantee of the imperceptibility of the adversarial examples, thus limiting the performances of adversarial training.

6 Conclusions

We proposed a novel identity-preserving adversarial training method IPAT for network embedding, guaranteeing the imperceptibility of adversarial examples and achieving a robust network embedding model. Our method is applicable to enhance any existing network embedding methods as a flexible module. The results on datasets Cora, Citeseer, Pubmed, and Wiki showed that our IPAT outperforms the state-of-the-art algorithms on link prediction and node classification tasks. Meanwhile, the experiments proved that our IPAT improves the robustness of network embedding models.

For future work, we plan to explore generating identity-preserving adversarial examples in the raw network data space.

Conflict of Interest Hua-Wei Shen and Xue-Qi Cheng are editorial board members for Journal of Computer Science and Technology, and were not involved in the editorial review of this article. All authors declare that there are no other competing interests.

References

[1] Cui P, Wang X, Pei J, Zhu W W. A survey on network embedding. *IEEE Trans. Knowledge and Data Engineering*, 2019, 31(5): 833–852. DOI: 10.1109/TKDE.2018.2849727.

[2] Qu L, Zhu H S, Zheng R Q, Shi Y H, Yin H Z. ImGAGN: Imbalanced network embedding via generative adversarial graph networks. In *Proc. the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Aug. 2021, pp.1390–1398. DOI: 10.1145/3447548.3467334.

[3] Ruan C Y, Wang Y, Ma J G, Zhang Y C, Chen X T. Adversarial heterogeneous network embedding with metapath attention mechanism. *Journal of Computer Science and Technology*, 2019, 34(6): 1217–1229. DOI: 10.1007/s11390-019-1971-3.

[4] Hu Z N, Dong Y X, Wang K S, Chang K W, Sun Y Z. GPT-GNN: Generative pre-training of graph neural networks. In *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, pp.1857–1867. DOI: 10.1145/3394486.3403237.

[5] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: Online learning of social representations. In *Proc. the 20th ACM SIGKDD International Conference on Knowledge Dis-*

covery and Data Mining, Aug. 2014, pp.701–710. DOI: 10.1145/2623330.2623732.

[6] Dai Q Y, Shen X, Zhang L, Li Q, Wang D. Adversarial training methods for network embedding. In *Proc. the World Wide Web Conference*, May 2019, pp.329–339. DOI: 10.1145/3308558.3313445.

[7] Dai Q Y, Li Q, Tang J, Wang D. Adversarial network embedding. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, Feb. 2018, pp.2167–2174. DOI: 10.1609/aaai.v32i1.11865.

[8] Qin Y, Carlini N, Cottrell G W, Goodfellow I J, Raffel C. Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. In *Proc. the 36th International Conference on Machine Learning*, Jun. 2019, pp.5231–5240.

[9] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I J, Fergus R. Intriguing properties of neural networks. In *Proc. the 2nd International Conference on Learning Representations*, Apr. 2014.

[10] Kipf T N, Welling M. Variational graph autoencoders. In *Proc. the NIPS Workshop on Bayesian Deep Learning*, Dec. 2016.

[11] Veličković P, Fedus W, Hamilton W L, Liò P, Bengio Y, Hjelm R D. Deep graph infomax. In *Proc. the 7th International Conference on Learning Representations*, May 2019.

[12] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In *Proc. the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp.855–864. DOI: 10.1145/2939672.2939754.

[13] Wang D X, Cui P, Zhu W W. Structural deep network embedding. In *Proc. the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp.1225–1234. DOI: 10.1145/2939672.2939753.

[14] Liu J, He Z C, Wei L, Huang Y L. Content to node: Self-translation network embedding. In *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2018, pp.1794–1802. DOI: 10.1145/3219819.3219988.

[15] Gao H C, Huang H. Deep attributed network embedding. In *Proc. the 27th International Joint Conference on Artificial Intelligence*, Jul. 2018, pp.3364–3370. DOI: 10.24963/ijcai.2018/467.

[16] Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[17] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In *Proc. the 5th International Conference on Learning Representations*, Apr. 2017.

[18] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In *Proc. the 6th International Conference on Learning Representations*, Apr. 2018.

[19] Sen P, Namata G, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T. Collective classification in network data. *AI Magazine*, 2008, 29(3): 93–106. DOI: 10.1609/aimag.v29i3.2157.

[20] Golub G H, Reinsch C. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 1970, 14(5): 403–420. DOI: 10.1007/BF02163027.

[21] You Y N, Chen T L, Sui Y D, Chen T, Wang Z Y, Shen Y. Graph contrastive learning with augmentations. In *Proc. the 34th International Conference on Neural Information Processing Systems*, Dec. 2020.

[22] Pan S R, Hu R Q, Long G D, Jiang J, Yao L N, Zhang C Q. Adversarially regularized graph autoencoder for graph embedding. In *Proc. the 27th International Joint Conference on Artificial Intelligence*, Jul. 2018, pp.2609–2615. DOI: 10.24963/ijcai.2018/362.

[23] Yang C, Liu Z Y, Zhao D L, Sun M S, Chang E Y. Network representation learning with rich text information. In *Proc. the 24th International Conference on Artificial Intelligence*, Jun. 2015, pp.2111–2117. DOI: 10.5555/2832415.2832542.

[24] Abadi M, Barham P, Chen J M, Chen Z F, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray D G, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X Q. Tensorflow: A system for large-scale machine learning. In *Proc. the 12th USENIX Conference on Operating Systems Design and Implementation*, Jun. 2016, pp.265–283. DOI: 10.5555/3026877.3026899.

[25] Noble W S. What is a support vector machine? *Nature Biotechnology*, 2006, 24(12): 1565–1567. DOI: 10.1038/nbt1206-1565.

[26] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, VanderPlas J T, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 2011, 12: 2825–2830. DOI: 10.5555/1953048.2078195.

[27] Yang Z L, Cohen W W, Salakhutdinov R. Revisiting semi-supervised learning with graph embeddings. In *Proc. the 33rd International Conference on Machine Learning*, Jun. 2016, pp.40–48.

[28] Hu W H, Fey M, Zitnik M, Dong Y X, Ren H Y, Liu B W, Catasta M, Leskovec J. Open graph benchmark: Datasets for machine learning on graphs. In *Proc. the 34th International Conference on Neural Information Processing Systems*, Dec. 2020. DOI: 10.5555/2999792.2999959.

[29] Van Der Maaten L, Hinton G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008, 9(86): 2579–2605.

[30] Xu B B, Shen H W, Cao Q, Qiu Y Q, Cheng X Q. Graph wavelet neural network. In *Proc. the 7th International Conference on Learning Representations*, Apr. 2019.

[31] Chen T, Kornblith S, Norouzi M, Hinton G E. A simple framework for contrastive learning of visual representations. In *Proc. the 37th International Conference on Machine Learning*, Jul. 2020, Article No. 149.

[32] Qiu J Z, Chen Q B, Dong Y X, Zhang J, Yang H X, Ding M, Wang K S, Tang J. GCC: Graph contrastive coding

for graph neural network pre-training. In *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, pp.1150–1160. DOI: 10.1145/3394486.3403168.

[33] Hassani K, Khasahmadi A H. Contrastive multi-view representation learning on graphs. In *Proc. the 37th International Conference on Machine Learning*, Jul. 2020, pp.4116–4126. DOI: 10.5555/3524938.3525323.

[34] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In *Proc. the 27th International Conference on Neural Information Processing Systems*, Dec. 2014, pp.2672–2680.

[35] Wang H W, Wang J, Wang J L, Zhao M, Zhang W N, Zhang F Z, Xie X, Guo M Y. GraphGAN: Graph representation learning with generative adversarial nets. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, Feb. 2018, pp.2508–2515. DOI: 10.1609/aaai.v32i1.11872.

[36] Yu W C, Zheng C, Cheng W, Aggarwal C C, Song D L, Zong B, Chen H F, Wang W. Learning deep network representations with adversarially regularized autoencoders. In *Proc. the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2018, pp.2663–2671. DOI: 10.1145/3219819.3220000.

[37] Qin C L, Martens J, Gowal S, Krishnan D, Dvijotham K, Fawzi A, De S, Stanforth R, Kohli P. Adversarial robustness through local linearization. In *Proc. the 33rd International Conference on Neural Information Processing Systems*, Dec. 2019, pp.13824–13833.

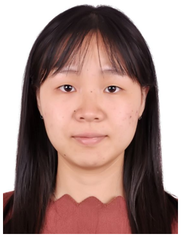
[38] Miyato T, Dai A M, Goodfellow I. Adversarial training methods for semi-supervised text classification. In *Proc. the 5th International Conference on Learning Representations*, Apr. 2017.

[39] Zhu D Y, Zhang Z W, Cui P, Zhu W W. Robust graph convolutional networks against adversarial attacks. In *Proc. the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2019, pp.1399–1407. DOI: 10.1145/3292500.3330851.

[40] Hu W B, Chen C, Chang Y M, Zheng Z B, Du Y F. Robust graph convolutional networks with directional graph adversarial training. *Applied Intelligence*, 2021, 51(11): 7812–7826. DOI: 10.1007/s10489-021-02272-y.

Ke-Ting Cen received his Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, in 2022. His research interests include network embedding and graph neural networks.

Hua-Wei Shen is a professor in the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences, Beijing. He received his Ph.D. degree in computer science and technology from ICT, CAS, Beijing, in 2010. His major research interests include network science, social media analytics, and recommendation.



Qi Cao is an assistant professor in the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing. She received her Ph.D. degree in computer science and technology from ICT, CAS, Beijing, in 2020. Her research interests include social media computing, influence modeling, information diffusion prediction, and trustworthy graph learning.

Bing-Bing Xu is an assistant professor in the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing. She received her Ph.D. degree in computer science from ICT, CAS, Beijing, in 2021. Her research interests include geometric deep learning, graph-based data mining, and graph convolution.

Xue-Qi Cheng is a professor in the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), and the University of Chinese Academy of Sciences, Beijing, and the director of the CAS Key Laboratory of Network Data Science and Technology, Beijing. His main research interests include network science, web search and data mining, big data processing, and distributed computing architecture.