Learning node representations against perturbations

Xu Chen [a,b], Yuangang Pan [c], Ivor Tsang [b,c], Ya Zhang [a,*]

[a] *Shanghai Jiao Tong University, Shanghai, China*
[b] *University of Technology Sydney, Sydney, Australia*
[c] *Center for Frontier AI Research, Agency for Science Technology, Singapore*

ARTICLE INFO

ABSTRACT

Recent graph neural networks (GNN) has achieved remarkable performance in node representation learning. One key factor of GNN's success is the *smoothness* property on node representations. Despite this, most GNN models are fragile to the perturbations on graph inputs and could learn unreliable node representations. In this paper, we study how to learn node representations against perturbations in GNN. Specifically, we consider that a node representation should remain stable under slight perturbations on the input, and node representations from different structures should be identifiable, which two are termed as the *stability* and *identifiability* on node representations, respectively. To this end, we propose a novel model called Stability-Identifiability GNN Against Perturbations (SIGNNAP) that learns reliable node representations in an unsupervised manner. SIGNNAP formalizes the *stability* and *identifiability* by a contrastive objective and preserves the *smoothness* with existing GNN backbones. The proposed method is a generic framework that can be equipped with many other backbone models (e.g. GCN, GraphSage and GAT). Extensive experiments on six benchmarks under both transductive and inductive learning setups of node classification demonstrate the effectiveness of our method. Codes and data are available online: https://github.com/xuChenSJTU/SIGNNAP-master-online

1. Introduction

Learning node representations for graphs is an important research topic that has great promise in a variety of areas. In recent years, it has been studied by two main methods: the network embedding based methods and the graph neural network (GNN) based methods. The network embedding based methods such as DeepWalk [1] and Node2Vec [2] mainly use the statistical random walks on graphs by a language model. DeepWalk and Node2Vec benefit the learning of node representations on graphs while rely on high-quality random walks. On the other hand, inspired by the graph convolutional theory and deep learning, GNN has emerged as one crucial technique for node representation learning on graphs. The concept of GNN was firstly proposed in [3]. Later, ChebNet [4], introduces a fast localized convolution approach on graphs in spectral domain. Inspired by the idea that high-order convolutions can be built by stacking multiple convolutional layers, graph convolutional network (GCN) [5] simplifies ChebNet with multiple stacked graph convolutional layers where each layer is one-hop convolution. GCN also connects the spectral graph convolution with information propagation, which encourages the following design of GNN models [6,7].

A key characteristic of GNN is smoothing which aggregates the features of a node and its nearby neighbors [8], enforcing *smoothness*

on node representations. However, most GNN models are fragile to the perturbations on the input and learn unreliable node representations [9,10]. Studying how to learn node representations against perturbations in GNN is a promising topic. When learning against perturbations, a node's representation should be stable to the slight perturbations on graph inputs, which can be termed as the *stability* of node representations. Meanwhile, in order to emphasize the graph signals against perturbations, nodes of different structures should have identifiable representations, i.e. the *identifiability* of node representations. In Fig. 1, we provide an example to illustrate the *smoothness*, *identifiability* and *stability* for node representations on graphs. Designing a model that simultaneously enforces the three properties is important for node representation learning on graphs. However, there are several difficulties: (1) The inappropriate way of constructing perturbations may introduce bias in *stability* and *identifiability* estimation. (2) How to formulate the optimization objective of *stability* and *identifiability* determines the estimation quality.

In this paper, we propose a new model named Stability-Identifiability GNN Against Perturbations (SIGNNAP) to learn reliable node representations on graphs. Specifically, the *stability* and *identifiability* are interpreted as high similarity for representations of the same node with
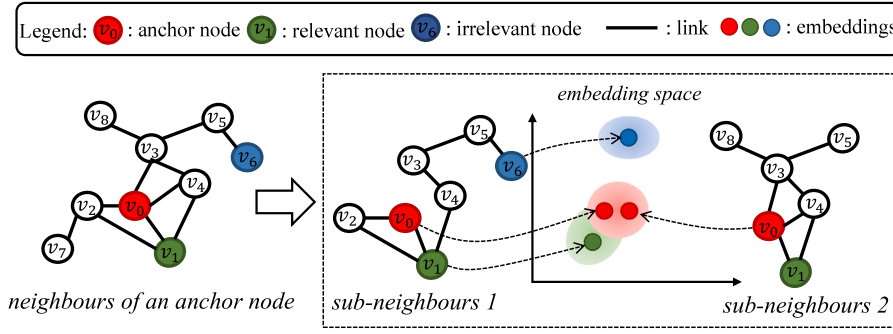
Fig. 1. An example to illustrate the *smoothness*, *stability* and *identifiability* of node representations on graphs. For an anchor node v_0, let v_1 be a relevant node and v_6 be an irrelevant node. Here, different sub-neighbors refer to different variants of the original neighbors under slight perturbations (e.g. drop edge). A reliable GNN model should preserve the *smoothness* (i.e. the embeddings of relevant nodes (v_0 and v_1) are close to each other), the *stability* (i.e. the representations of v_0 is stable to slight perturbations) and the *identifiability* (i.e. the embeddings of irrelevant nodes (v_0 and v_6) are distant from each other).

different perturbations and low similarity for representations of different nodes, respectively, which is formulated as a contrastive objective function. In SIGNNAP, we study several general perturbations on edges or node attributes with explicit distributions so that the contrastive objective can marginalize it to avoid biased *stability* and *identifiability* estimation. More importantly, the proposed model is generic and can be equipped with different GNN backbones such as GCN [5], GraphSage [7] and GAT [6]. By optimizing the objective in our model, we show that we can learn more reliable node representations that benefit the downstream node classification task. With the *stability* and *identifiability*, we empirically show that the model has better ability of preventing the over-smoothing and over-fitting problem on graphs. The contributions are summarized as follows:

- To learn node representations against perturbations in GNN, we advocate to guarantee the *stability* and *identifiability* property and develop a novel model named SIGNNAP that learns reliable node representations in an unsupervised manner. The model is generic and can be equipped with many popular GNN backbone models for boosted performance.
- We conduct extensive experiments on six benchmarks under both transductive and inductive learning setups. The results show that SIGNNAP has better ability of preventing the over-smoothing and over-fitting issue and show superior representation learning performance.

2. Related work

Recent progress of node representation learning on graphs is mainly categorized into two groups: network embedding and GNN.

Network Embedding: Network embedding arises as one hot research topic to learn representative node embeddings for a given network. Various methods have been proposed for network embedding. For example, inspired by the skip-gram model for word representation in Natural Language Processing (NLP), LINE [11] defines first-order and second-order proximity to describe the context of a node and trains node embeddings via negative sampling. Node2Vec [2] extends DeepWalk by designing a biased random walk to control the Bread First Search (BFS) and Deep First Search (DFS). These methods rely much on high-quality random walks and are labor-consuming because of carefully designed sampling strategies and fine-tuned hyper-parameters.

Graph Neural Networks (GNN): Inspired by the success of convolution on images in Euclidean space, researchers tried to define the convolution on graphs in non-Euclidean space. In ChebNet [4], a fast and localized convolution filter is defined on graphs in spectral domain. In [12], the authors proposed graph convolutional networks (GCN) which utilizes a localized first-order approximation of the convolution in ChebNet. GCN connects the spectral graph convolution to information propagation, and broadens the way of other GNN models such as

GraphSage [13] and graph attention networks (GAT) [14]. Meanwhile, some works [15] reveal that the key of GCN's success is its smoothing characteristic that aggregates node features from the nearby neighbors. Although the smoothing characteristic has put forward GNN's progress, when stacking too many layers, the GNN model usually faces the over-smoothing problem where all node representations converge to the same subspace and are unidentifiable from each other. Accordingly, ResGCN [16] introduces residual connections in GCN. DropEdge [10] proposes to randomly drop edges with a certain ratio before graph convolutions. These GNN models mainly focus on how to better enforce the *smoothness* property.

Perturbations in GNN: Employing perturbations to learn a robust GNN model has widely appeared in graph attack and defense. For example, both [17,18] explore the perturbations on edges to make attack and defense on graphs. In particular, [17] analyzes the attack from an optimization perspective and proposes a gradient based model that can conduct both edge addition and deletion. [18] explores the attacks by modifying the combinatorial structure of data and further proposes a reinforcement learning based model that learns a generalizable attack policy.

Although SIGNNAP also employs perturbations, it is fundamentally different from the graph attack and defense works. (1). Graph attack aims to attack a target node and change its prediction labels, and graph defense targets to keep the prediction labels unchanged. While SIGNNAP targets to learn reliable node representations in an unsupervised manner. (2). Graph attack involves techniques such as adversarial learning. While SIGNNAP is based on the theory of marginalizing noise and contrastive learning (CL).

Contrastive Learning and GNN: There are some parallel works incorporating contrastive learning with GNN. For instance, DGI [19] maximizes the mutual information between a node representation and the high-level graph representation by contrastive learning. CMV [20] extends DGI and builds contrastive signals between node representations and the graph representation in a multi-view formulation. GMI [21] proposes to estimates the mutual information (MI) in DGI by decomposed parts. By incorporating random walking sampling and contrastive learning, GCC [22] builds contrastive signals between sampled multi-view graphs. GRACE [23] corrupts both the graph structures and node attributes to generate two views and uses all other nodes from two augmentation views as negative pairs in the InfoNCE objective function. GCA [24] designs adaptive augmentation on the graph topology and node attributes to incorporate various priors for topological and semantic aspects of the graph. Some works extends the negative-sampling free CL models from vision domain to graph domain. For example, BGRL [25] extends BYOL [26], a popular self-supervised learning method in computer vision, to graphs. BGRL includes carefully-designed tricks, such as stop gradient, non-symmetric networks and momentum encoders, to avoid degenerate solutions. Following Barlow Twins [27], G-BT [28] utilizes a cross-correlation-based
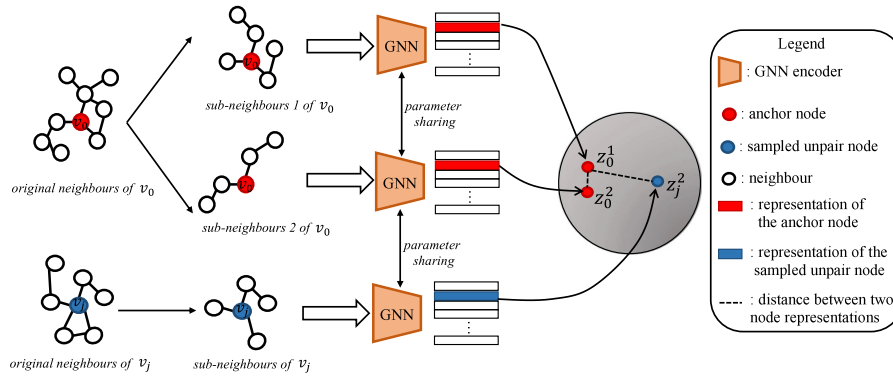
Fig. 2. The framework of our Stability-Identifiability GNN Against Perturbations (SIGNNAP).

loss instead of the non-symmetric network in BGRL. There are some other works that concentrate on learning graph-level representations such as [29,30], which are different to our node-level learning case and we would not introduce more here.

Although these methods also utilize contrastive learning, there are key differences between them and our SIGNNAP. DGI, CMV and GMI formalize the contrastive learning between a node representation and the graph representation. In this case, they try to keep more high-order information of nodes and achieve better *smoothness*. Instead, SIGNNAP contrasts the representations of nodes, aiming to maintain the *stability* and *identifiability* against perturbations. GCA specifically investigates adaptive augmentation strategies to construct positive and negative pairs in graph CL models, which is orthogonal to our work. The negative-sampling free graph CL models (i.e. BGRL and G-BT) usually suffer from degenerate solutions and thus require carefully designed learning tricks. By contrast, the learning process of CL with negatives samples is usually more stable and reliable to reach satisfying performance. Compared to GCC, SIGNNAP has different motivations, methodology and different results, which would be demonstrated later.

3. Method

The problem of node representation learning on graphs is formulated as follows. Given an undirected graph $\mathcal{G} = (\mathcal{V}, A)$, $\mathcal{V} = \{v_i | 1 \leq i \leq N\}$ represents nodes in the graph and $A \in \mathbb{R}^{N \times N}$ denotes the adjacent matrix where $A_{ij} = 1$ indicates node i and node j are connected while $A_{ij} = 0$ means not. When the nodes have attributes, we denote $X \in \mathbb{R}^{N \times F}$ as the attribute matrix, where F is the attribute dimension. The goal of node representation learning on graphs is to encode nodes into representative embeddings. In SIGNNAP, we use popular GNN backbones to guarantee the nice *smoothness* property. The *stability* and *identifiability* are maintained by a contrastive objective function. The general architecture of SIGNNAP is shown in Fig. 2.

3.1. Stability by edge perturbations

To achieve the *stability*, an intuitive way is to enforce a node representation from the slightly changed input to be close with the true one. However, it is non-trivial to obtain the true one. Based on [31], we can achieve the *stability* by marginalizing a perturbation distribution on the original samples.

To clarify our method, given an arbitrary anchor node v_0 and a perturbation distribution $p(\epsilon)$, we can obtain the node representation z_0 by reparameterizing its true posterior $q(z_0|v_0)$ with a differentiable transformation $f_\theta(\epsilon, v_0)$ of a perturbation variable ϵ:

$$z_0 = f_\theta(\epsilon, v_0) \; with \; \epsilon \sim p(\epsilon) \tag{1}$$

In Eq. (1), enforcing the *stability* among multiple z_0 with different ϵ is computational-inefficient in each step. Instead, we employ an equivalent manner which enforces the *stability* between every two random

samples z_0^1, z_0^2 from $f_\theta(\epsilon, v_0)$ by sampling two ϵ independently. Since learning on graphs cares more about structures, here we use Bernoulli distribution on edges as $p(\epsilon)$. In particular, if we denote the original neighbors of v_0 as $\mathcal{N}_0$, we randomly drop edges of $\mathcal{N}_0$ with ratio ρ to have different sub-neighbors of v_0. In this case, $p(\epsilon)$ is represented as $p(\epsilon; \rho)$. Assuming f_θ is a GNN backbone that ensures the *smoothness* property, we have:

$$z_0^1 = f_\theta(\epsilon_1, \mathcal{N}_0), z_0^2 = f_\theta(\epsilon_2, \mathcal{N}_0) \; with \; \epsilon_1, \epsilon_2 \sim p(\epsilon; \rho) \tag{2}$$

where z_0^1, z_0^2 are the anchor node's representations from two different sub-neighbors by corrupting $\mathcal{N}_0$ independently. Other perturbation distributions may have different effects. The effects of other common perturbations including perturbations on node attributes X are studied in the experiments. Note that we mainly study perturbations with explicit distributions here to explore the general case of learning node embeddings against perturbations. More complicated types such as task-specific perturbations or generated perturbations are beyond the scope of this work and could be explored in future.

3.2. Combating perturbations by stability and identifiability

In order to combat the perturbations for learning node representations, a typical way is to make sure a node representation remains stable under slight perturbations, i.e. the *stability*. Besides, we propose to emphasize the original graph signals against perturbations by enforcing nodes with different structures are identifiable in latent space, i.e. the *identifiability*. In our model, the *stability* and *identifiability* are interpreted as high similarity for representations of the same node with different perturbations and low similarity for representations of different nodes, respectively. Since contrastive learning has been widely proved as a good way to achieve the pair-similarity optimization, we propose to use a contrastive loss to formalize the two properties. Specifically, we denote (z_0^1, z_0^2) as a positive pair. If z_j^2 is a node representation not belonging to v_0 (i.e. $j \neq 0$), then (z_0^1, z_j^2) is a negative one. Then the contrastive objective is shown as:

$$\mathcal{L}_1 = -\mathbb{E}_{\{z_0^1, z_0^2, z_j^2\}_{j=1}^K} \left[\log \frac{h_\phi(z_0^1, z_0^2)}{\sum_{j=0}^K h_\phi(z_0^1, z_j^2)} \right] \tag{3}$$

where h_ϕ is a score function that is high for positive pairs and low for negative pairs. K indicates the sampling size of negative pairs. Note that the expectation on $\{z_0^1, z_0^2, z_j^2|_{j=1}^K\}$ in Eq. (3) marginalizes the perturbation distribution in Eq. (2), which facilitates the unbiased *stability* estimation. Through the contrastive objective in Eq. (3), we push the positive pairs z_0^1 and z_0^2 are closer than the negative pairs z_0^1 and z_j^2 in the embedding space. We implement the score function as the Gaussian potential kernel (also known as the Radial Basis Function (RBF) kernel).

$$h_\phi(z_0^1, z_0^2) = e^{z_0^{1^T} z_0^2 / \tau} \tag{4}$$

Table 1
The statistics of six benchmarks.

	#nodes	#edges	#density	#classes	#features	#label rate	Train/Val/Test
Pubmed	19,717	44,324	0.01%	3	500	0.30%	60/500/1000
Facebook	22,470	170,823	0.03%	4	4714	0.35%	80/120/rest
Coauthor-CS	18,333	81,894	0.02%	15	6805	1.60%	300/450/rest
Amazon-Com	13,752	245,861	0.13%	10	767	1.45%	200/300/rest
Amazon-Pho	7650	119,081	0.20%	8	745	2.09%	160/240/rest
Coauthor-Phy	34,493	247,962	0.02%	5	8415	57.98%	20,000/5000/rest

where τ is a temperature to control the distribution of h_ϕ. Among a general class of kernels, the RBF kernel can well distribute the negative node representations uniformly in the embedding space for good *identifiability*. Besides, with the perturbations defined in Section 3.1, the *stability* in SIGNNAP is equivalent to the isomorphism between two sub-graphs [32]. Through the above analysis, we conclude that the contrastive objective in our SIGNNAP matches the two properties well. Similarly, by substituting the anchor representation z_0^1 in Eq. (3) with z_0^2, we have:

$$\mathcal{L}_2 = -\mathbb{E}_{\{z_0^2, z_0^1, z_j^1|_{j=1}^K\}}\left[\log \frac{h_\phi(z_0^2, z_0^1)}{\sum_{j=0}^K h_\phi(z_0^2, z_j^1)}\right] \quad (5)$$

where $\mathcal{L}_2$ is a symmetric form of $\mathcal{L}_1$ and can help to stabilize the training.

3.3. Objective function

By summing Eqs. (3) and (5) up, we have our final objective function:

$$\min_\theta \mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 \quad (6)$$

where θ is the network parameter from the GNN backbone. Since the expectations in Eqs. (3) and (5) do not have analytic formulations, we instead resort to unbiased Monte Carlo estimation to approximate the two equations. In the objective, ρ and K control the variations of *stability* and *identifiability* of the inputs, which is empirically analyzed with experiments in the experiments. Moreover, we demonstrate that the proposed objective has connections with the mutual information between z_0^1 and z_0^2, which is shown as:

Lemma 1. *The proposed objective $\mathcal{L}$ is an estimator of the mutual information between z_0^1 and z_0^2, showing that:*

$$\mathcal{I}(z_0^1, z_0^2) \geq \log(K) - \mathcal{L} \quad (7)$$

therefore minimizing $\mathcal{L}$ actually maximizes the lower bound of $\mathcal{I}(z_0^1, z_0^2)$. The lower bound becomes tighter when K becomes larger.

Details about Lemma 1 is demonstrated in Appendix A.1. Minimizing $\mathcal{L}$ maximizes the mutual information between z_0^1 and z_0^2 towards a direction where the positive pairs are more similar or dependent than the negative ones.

3.4. Acceleration strategies for training

DropEdge Sampling Strategy: If we denote the number of training epochs as T, the complexity of constructing various sub-neighbors of N nodes is $\mathcal{O}(NT)$. In our implementation, following DropEdge, we randomly drop edges on the adjacent matrix A one time for each epoch and then feed the corrupted adjacent matrices A^1 and A^2 instead of $\mathcal{N}_0^1$ and $\mathcal{N}_0^2$ into the model training. Then the complexity is reduced to $\mathcal{O}(T)$ and dropping edge is accelerated. To further accelerate training, like other contrastive loss based algorithms (e.g. DGI and CMV), we set the sampling method of negative pairs as the random sampling.

Memory Bank Strategy: In order to efficiently sample K negative nodes for $\mathcal{L}$, we follow the memory bank strategy in [33]. In particular, the latent features of all nodes are stored in memory and synchronously updated after loss back propagation. It is also worthwhile to mention that the memory bank strategy here requires to save the representations of all nodes in learning, which may bring a memory challenge when the graph is extremely large.

4. Experiments and analysis

4.1. Experiment setups

4.1.1. Dataset description

We conduct the experiments on six benchmarks varying in graph types and sizes.[1] Pubmed is a widely used citation network. Facebook [34] is a web page dataset where nodes are official Facebook pages and the edges are mutual connections between sites. Coauthor-CS and Coauthor-Phy [35] are two coauthor datasets based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. Amazon-Com (i.e. Amazon-Computer) and Amazon-Pho (i.e. Amazon-Photo) are two segments of the Amazon co-purchase graph [36], where nodes are items and edges indicate two items are frequently co-purchased together. Each dataset has raw node features and class labels, following mainstream works [1,20], we perform the node classification task to make evaluation. For Pubmed, Facebook, Coauthor-CS, Amazon-Com and Amazon-Pho, we conduct the transductive learning where all nodes and their raw features are accessible during training. For Coauthor-Phy, we conduct the inductive learning where the test nodes are not seen during training.

Different dataset splits on the node classification can have different evaluation values. The way of data splitting used in many graph CL works (i.e., DGI [19], CMV [20], GCC [22]) originates from the semi-supervised works of graph representation learning. Whereas some graph CL works, e.g., (GRACE [23], BGRL [25], G-BT [28], GCA [24]), utilize a random splitting of the nodes into (80%-10%–10%) train/validation/test set. Usually, compared to that in the former way, the latter way of splitting would favor the training, except for Coauthor-Phy in inductive learning setting. In order to make comparison of different models, by following common graph learning works [19,20], we use the widely-recognized semi-supervised setting of data splitting. On Pubmed, the train/val/test nodes are the same as previous works [12,19]. For Facebook, Coauthor-CS, Amazon-Com and Amazon-Pho, we follow the setting in [35] where 20 nodes of each class are randomly sampled as the train set and 30 nodes of each class are randomly sampled as the validation set and the rest is the test set. For Coauthor-Phy in inductive learning, we randomly sample 20,000 nodes as train set and 5000 nodes as validation set and the rest as test set. The dataset statistics are shown in Table 1.

4.1.2. Baselines

We make the comparison of our model with the following supervised models: GCN [12], ResGCN [16], JKNet [37], GraphSage [7], GAT [38], DropEdge [10] and other unsupervised models: DeepWalk [1], Node2vec [2], ARWMF [39], DGI [19], CMV [20], GCC [22], GRACE [23], BGRL [25], G-BT [28] and GCA [24]. Among the unsupervised models, BGRL and G-BT are learned without negatives while the others are not.

---

[1] Note that Reddit dataset is quite large and is not included here since we do not have much computation resources. Instead, we include other benchmarks.

Table 2

Classification accuracy (%) on different benchmarks. Standard deviation is reported in percentage format. The best value for unsupervised models is emphasized in bold. The second value for unsupervised models is emphasized with underline. "-" means the method does not support this setting. We also show the average rank of unsupervised models on the six benchmarks. Lower rank score means better performance.

	Method	Transductive					Inductive	Average rank
		Pubmed	Facebook	Coauthor-CS	Amazon-Com	Amazon-Pho	Coauthor-Phy	
Supervised	GCN	79.20(±0.38)	66.37(±0.24)	92.01(±0.14)	81.18(±0.27)	85.82(±0.30)	93.35(±0.02)	
	ResGCN	77.74(±0.39)	67.69(±0.60)	92.84(±0.24)	81.10(±0.70)	87.21(±0.50)	95.88(±0.03)	
	JKNet	77.84(±0.11)	68.09(±0.75)	92.76(±0.22)	80.91(±0.83)	87.25(±0.50)	95.56(±0.15)	
	GraphSage	79.02(±0.31)	69.62(±0.38)	92.60(±0.16)	82.10(±0.22)	87.60(±0.34)	95.28(±0.16)	
	GAT	78.71(±0.21)	72.24(±0.07)	91.23(±0.06)	81.85(±0.73)	87.02(±1.51)	93.96(±0.04)	/
	DropEdge(GCN)	78.82(±0.29)	66.10(±0.19)	92.12(±0.12)	81.42(±0.21)	85.58(±0.13)	93.32(±0.02)	
	DropEdge(ResGCN)	77.52(±0.38)	67.30(±0.75)	93.01(±0.16)	81.50(±0.74)	87.46(±0.27)	95.89(±0.04)	
	DropEdge(JKNet)	77.85(±0.16)	67.68(±0.54)	92.74(±0.20)	81.70(±0.43)	87.01(±0.42)	95.77(±0.02)	
	DropEdge(GraphSage)	78.53(±0.31)	69.25(±0.30)	92.77(±0.08)	82.11(±0.30)	87.64(±0.40)	95.41(±0.30)	
	DropEdge(GAT)	78.90(±0.30)	71.57(±0.21)	91.32(±0.14)	82.20(±0.50)	87.59(±0.73)	93.80(±0.13)	
Unsupervised	DeepWalk	65.59(±1.43)	63.04(±1.32)	77.81(±0.86)	76.93(±0.73)	81.50(±0.82)	91.17(±0.33)	9.83
	Node2Vec	70.34(±0.69)	69.69(±0.77)	79.93(±0.62)	75.49(±1.08)	82.21(±0.67)	91.43(±0.73)	10.0
	ARWMF	78.03(±0.85)	61.97(±0.94)	86.02(±0.82)	68.34(±1.54)	78.35(±0.56)	–	–
	DGI	79.24(±0.60)	69.53(±1.25)	91.41(±0.12)	71.41(±0.85)	79.34(±0.66)	93.26(±0.35)	8.66
	CMV	80.10(±0.10)	67.24(±0.13)	90.73(±0.71)	67.15(±1.98)	79.54(±1.47)	91.49(±0.41)	9.83
	GCC	80.60(±0.45)	70.36(±0.53)	91.76(±0.12)	74.18(±1.02)	83.60(±0.52)	**93.97(±0.16)**	5.5
	GRACE	78.10(±0.10)	65.81(±0.27)	90.20(±0.33)	69.32(±0.78)	66.48(±0.30)	72.90(±0.18)	12.16
	BGRL	71.01(±0.42)	62.42(±0.21)	89.31(±0.45)	**81.72(±0.86)**	86.02(±0.53)	75.85(±0.20)	9.0
	G-BT	80.04(±0.63)	64.14(±0.47)	90.38(±0.62)	75.17(±1.05)	84.49(±0.66)	74.05(±0.25)	8.83
	GCA	80.02(±0.18)	71.52(±0.34)	90.97(±0.29)	77.30(±0.58)	85.50(±0.40)	73.09(±0.21)	6.5
	SIGNNAP(JKNet)	80.82(±0.48)	68.77(±1.31)	91.48(±0.18)	73.91(±0.50)	83.55(±0.29)	92.82(±0.08)	7.0
	SIGNNAP(GAT)	80.93(±0.40)	**78.39(±0.64)**	90.14(±0.37)	75.11(±0.47)	**89.84(±0.26)**	92.34(±0.03)	4.5
	SIGNNAP(ResGCN)	77.34(±0.53)	71.21(±0.29)	<u>92.01(±0.19)</u>	77.24(±0.28)	84.09(±0.21)	<u>93.43(±0.27)</u>	5.16
	SIGNNAP(GCN)	<u>81.34(±0.54)</u>	71.13(±0.16)	**92.35(±0.27)**	74.65(±0.16)	85.74(±0.25)	92.76(±0.10)	4.66
	SIGNNAP(GraphSage)	**81.81(±0.20)**	<u>71.55(±0.48)</u>	91.21(±0.24)	<u>78.49(±0.08)</u>	<u>86.26(±0.43)</u>	93.33(±0.02)	**2.83**

4.1.3. Parameter settings

We implement SIGNNAP with Pytorch on a machine with one Nvidia 1080-Ti GPU. For GraphSage, GAT and DGI, we use the codes from a famous GNN library- DGL.[2] For other baselines, we use the codes released by the authors. In SIGNNAP, we construct the positive nodes with the same drop ratio $\rho = 0.3$ like DropEdge [10] and randomly sample $K = 1024$ negative pairs. We set the learning rate as 0.001 with 5000 iterations and the temperature τ as 0.1. For unsupervised methods, we train a one-layer linear classifier for evaluation. More detailed parameter settings are summarized in Appendix B.1.

4.2. Performance comparison

4.2.1. Node classification performance

Following mainstream works [1,11,19,20,40], we conduct the node classification task to verify the effectiveness of the proposed method. The results on different benchmarks are summarized in Table 2. In this table, we report the mean classification accuracy (with standard derivation) on the test nodes after 5 runs with different random seeds. From the table, we have the following observations.

(1). The proposed method generally performs better than recent unsupervised models and even exceeds the supervised models in some cases. On Pubmed, SIGNNAP(GCN) reaches a 2.10% gain over DGI and a 0.74% gain over GCC. With the same encoder, SIGNNAP(GCN) can generally achieve a better rank score of 4.66, compared to GRACE with 12.16, G-BT with 8.83, GCA with 6.5 and GCC with 5.5, which illustrates that the improvement is from the model itself instead of the backbone. **(2).** For the inductive learning on Coauthor-Phy, we can see both DGI, GCC and SIGNNAP perform a bit worse than the supervised methods. A possible reason is that the label rate of Coauthor-Phy is quite large compared to other datasets. The supervised models are not easy to be over-fitting with a large label rate. **(3).** It is interesting to find that SIGNNAP(GraphSage) not always performs better than SIGNNAP(GCN). This is understandable since different datasets have

different distributions that favor different backbones. In our experiments, we use these common backbones to demonstrate SIGNNAP's flexibility in use.

4.2.2. On preventing over-smoothing

As discussed in [8], when stacking too many GCN layers, the over-smoothing issue arises which means the top-layer node embeddings converge to the same subspace and become unidentifiable from each other. As SIGNNAP enforces the *identifiability* property which tries to learn different embeddings for nodes with different structures, it is curious to see if SIGNNAP has better ability of preventing the over-smoothing issue. In this experiment, we visualize the node embeddings by t-SNE when increasing the depth of GCN layers. The results of different models are shown in Fig. 3.

From this figure, we can see that: **(1).** When increasing the number of GCN layers (e.g. 10-layer), SIGNNAP(GCN) shows more identifiable node embeddings while baseline models easily have collapsed subspace. **(2).** DropEdge(GCN) can alleviate the over-smoothing problem in GCN, while it does not perform better than SIGNNAP(GCN). The contrastive objective in SIGNNAP(GCN) can work with DropEdge together and show better performance. **(3).** Note that the embeddings of GCN and DropEdge(GCN) are learned in a supervised manner while those of DGI and SIGNNAP(GCN) are learned in an unsupervised manner, so the supervised models may have better performance than the unsupervised ones when the number of layers is 2 or 4.

### 4.2.3. On preventing over-fitting

In this experiment, we investigate the over-fitting problem of different models. We show the results of different training layers, train accuracy and validation accuracy of different methods in Fig. 4.

By analyzing the results in Fig. 4, we make the following summarization: **(1).** From Fig. 4(a), we see that the proposed SIGNNAP has improvement over baseline models. Especially when the number of layer increases from 2 to 10, with nearly four times increased learning parameters, the accuracy of baselines drops from 67% to nearly 30%, while the accuracy of SIGNNAP(GCN) drops from 72% to nearly 60%. **(2).** Besides, from Fig. 4(b) and (c), we observe that the supervised models (GCN and DropEdge) have a severe over-fitting

---

[2] https://www.dgl.ai/
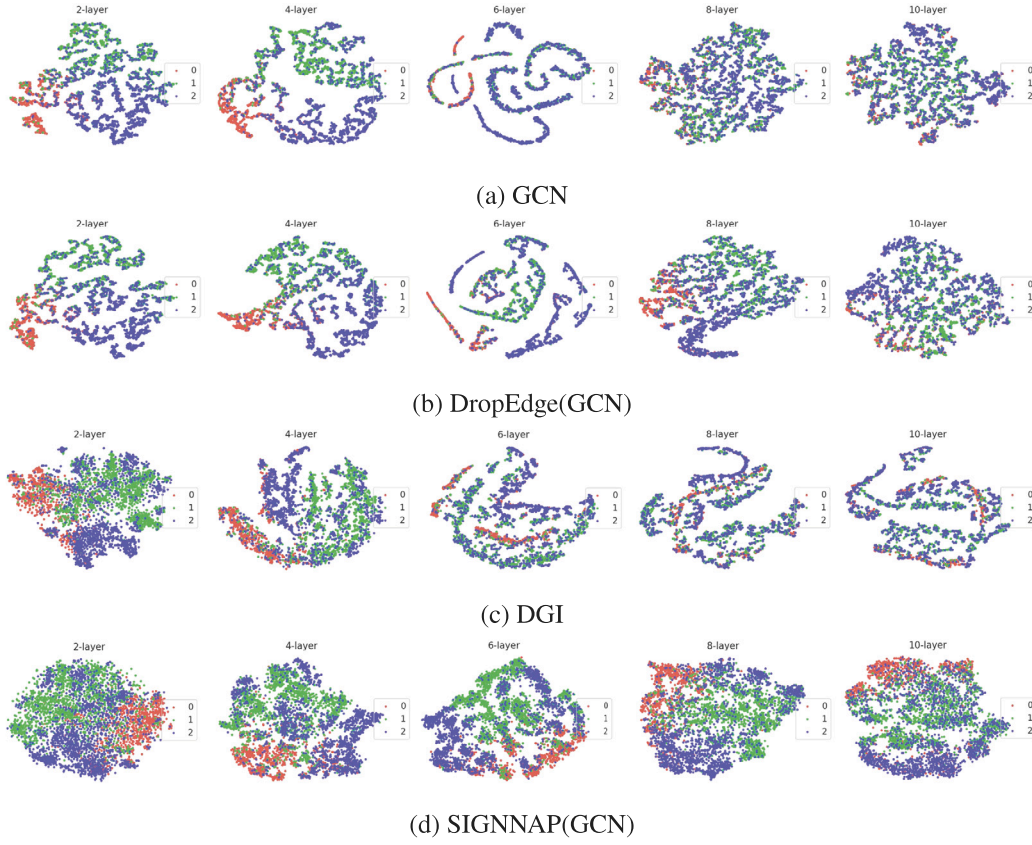
Fig. 3. The results of over-smoothing with different number of GCN layers on Pubmed. The number of layers is 2,4,6,8,10 from left to right in each row. We show the results of four models here to give an example. Different colors indicate different categories. Results of other methods and datasets follow similar patterns.
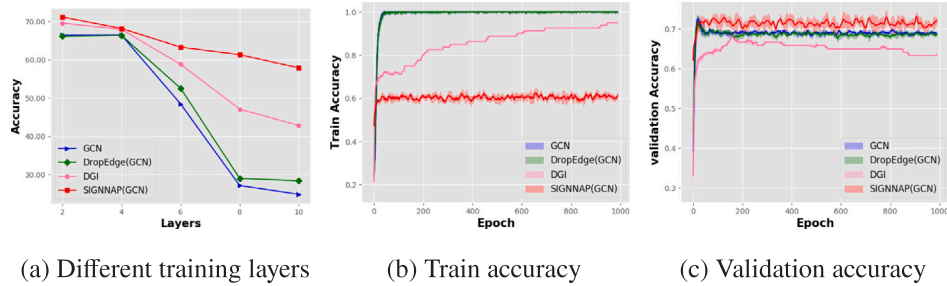
Fig. 4. The results of over-fitting on Facebook. (a) shows the comparison results of different training layers. (b) indicates the train accuracy of different methods. (c) indicates the validation accuracy of different methods. The accuracy curves of unsupervised models are plot based on the one-layer linear classifier. We show the results of several baselines here to illustrate the idea. Results of other methods and datasets follow similar patterns.

problem. For SIGNNAP, we can see that although it has the lowest train accuracy, it shows the best validation accuracy. It is mainly because SIGNNAP learns more stable node representations, namely the GNN feature extractor has less variance and has better generalization ability.

4.3. Empirical results of the three properties

4.3.1. Stability of node representations

Here we conduct an experiment to show that our method learns more stable node representations on graphs. Specifically, after we train a model, we fix the model and add perturbations to the input by dropping edges with $\rho = 0.3$ ten times. In this way, for an arbitrary node v_0, we can have its ten variant inputs and their corresponding ten node representations by the fixed model. Next, we calculate the cosine similarity matrix $S \in \mathbb{R}^{10\times10}$ of these ten variants' representations. If a model learns stable node representations, the similarity between each two representations should be large. Here, we sample the test nodes

and calculate the S matrix of each node for four methods. For each method, we average the S matrices of test nodes to obtain one matrix denoted as $\overline{S}$. For better comparison, we apply min–max normalization on the $\overline{S}$ matrices and show the results in Fig. 5.

From Fig. 5, we summarize that the proposed method learns more stable node representations. **(1)**. DGI have many light-colored blocks, which means the node representation is sensitive to the slight changes on the input. **(2)**. Compared to DGI, DropEdge(GCN) shows a better result with deeper-colored blocks, because DropEdge(GCN) tries to maintain the *stability* by assigning the same label to the input with slight perturbations. When using the label as an agent, it is hard to guarantee the *stability* on the node representations directly. Instead, the proposed SIGNNAP explicitly imposes the *stability* on learned node representations. **(3)**. Compared to GCC, SIGNNAP shows better *stability* results, because SIGNNAP can guarantee an unbiased *stability* estimation by marginalizing the perturbation distribution. While the random
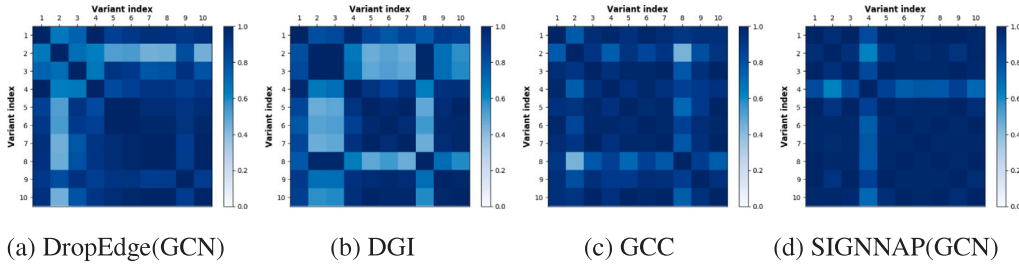
(a) DropEdge(GCN) (b) DGI (c) GCC (d) SIGNNAP(GCN)

Fig. 5. The representation *stability* comparison on Facebook. The mean value of the normalized cosine similarity in (a) (b) (c) (d) respectively is 0.857, 0.804, 0.915 and 0.946. Deeper color indicates larger similarity and more stability against perturbations on the input.
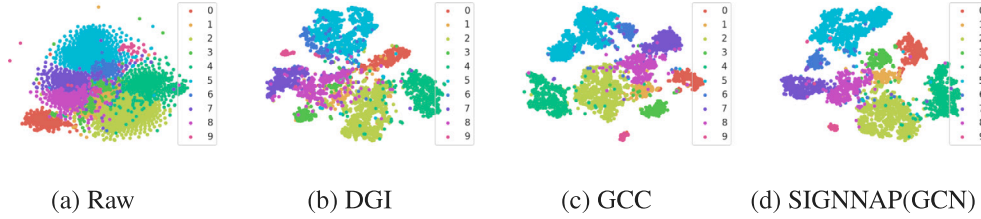

(a) Raw (b) DGI (c) GCC (d) SIGNNAP(GCN)

Fig. 6. The t-SNE comparison of learned node representations on Coauthor-CS. Note that for clear presentation, we visualize the first ten categories of nodes. (a) "Raw" means the raw node features are used. (b) (c) and (d) indicate the features are learned by DGI, GCC and SIGNNAP(GCN), respectively.

walk sampling in GCC without analytic formulation may have large bias in learning *stability*.

4.3.2. Smoothness and identifiability of node representations

We also conduct an experiment to show the *smoothness* and *identifiability* of learned node representations. Discussing *smoothness* and *identifiability* without any data is hard to understand. Here we use node categories to illustrate this by assuming that nodes from the same category should be clustered together (i.e. *smoothness*) and nodes from different categories should be discriminative (i.e. *identifiability*). In particular, we visualize the learned node representations on Coauthor-CS by t-SNE in Fig. 6.

From Fig. 6(a), it is clear that raw features are easily overlapped together. DGI, GCC and SIGNNAP(GCN) show better performance by identifying nodes from different clusters. Although GCC has similar loss formulation as SIGNNAP, it still shows inferior performance to SIGNNAP(GCN). We consider it is caused by the differences between them. **(1)**. GCC aims to learn the structural similarity and transferability in latent space by a pertaining task, so as to benefit downstream tasks. By contrast, SIGNNAP concentrates on the model's robust representation learning under different perturbations. **(2)**. SIGNNAP enforces an unbiased *stability* and *identifiability* estimation by marginalizing the perturbation distribution on original samples, while the random walk sampling in GCC without analytic formulation may have large bias in estimation. The better performance of SIGNNAP over GCC can also be verified in Table 2.

4.4. Ablation study

4.4.1. Different perturbations

In the method part, we use DropEdge which is an edge removal strategy to obtain different samples. We do not consider edge addition here since it has a complexity of $\mathcal{O}(N^2)$ and will incur much more noise than edge removal. For any model, proper noise in training can help it; while overwhelming noise level leads to deterioration. How to add edges is an interesting topic but is beyond the scope of SIGNNAP here.

In addition, we can also add perturbations on node attributes to have different samples of an anchor node v_0. Thus, we further explore the effects of different perturbations on node attributes. In particular, if we denote the attribute vector of v_0 as x_0 and a perturbation vector as δ, then the corrupted attribute vector can be represented as $x'_0 = x_0 + \delta$,

where δ can be sampled from different distributions such as Gaussian distribution. We show the results of different perturbations in Table 3. Note that we multiply 0.01 for noise from these distributions and then denote them as δ to adapt to the scale of normalized features. Here, we fix the backbone as GCN for SIGNNAP to analyze the effects of different perturbations.

From Table 3, we can see: **(1)**. Comparing perturbations on "A" and "X", the result of 'X'' row generally has worse performance than that of "A" row. A possible reason is that edges in graphs may have more uncertainty than attributes. Enforcing *stability* on structures can help the model adapt to the uncertain edges in information aggregation and learn more robust node features. **(2)**. Comparing perturbations on "A" and "A+X", DropEdge generally has better performance. Based on DropEdge, adding node attribute perturbations sometimes has positive effects while sometimes does not. It is hard to find a proper distribution when adding perturbations on node attributes since the node attributes are usually heterogeneous. By contrast, DropEdge matches different graphs well and is more practical and easier for usage in practice.

4.4.2. Hyper-parameter analysis

In the proposed method, the ratio of dropping edges ρ and the sampling size K respectively control the variations of *stability* and *identifiability* of inputs. The temperature τ in the score function controls the similarity scale in contrastive learning. In order to explore the effects of these hyper-parameters, we further conduct an experiment to illustrate this. The results on Pubmed are shown in Fig. 7. From this figure, we can summarize that: **(1)**. In Fig. 7(a), when K becomes too large, there is a risk of sampling structure-similar nodes as negative pairs. In other words, too large K can lead to inappropriate contrastive signals and deteriorates the quality of node representations. In general, the empirical $K = 1024$ can provide satisfied performance. **(2)**. In Fig. 7(b), a proper drop ratio ρ tends to be around 0.5. A too small ρ leads to less variants of a node's neighbors. A too large ρ will lose too much neighborhood information, which hinders the information propagation on graphs. In general, $\rho = 0.3$ is an empirical value of dropedge technique [10]. **(3)**. The temperature τ controls the distribution of the score function. The method has its best performance when $\tau = 0.1$. Empirically, $\tau \ll 0.1$ will cause value explosion in networks because of the exponential property. In summary, $\tau = 0.1$ is an empirical value for satisfied model performance.

Table 3

The results of SIGNNAP(GCN) with different kinds of perturbations. "A", "X" and "A+X" respectively indicate the perturbations on structures, node attributes or both. "Gaussian", "Laplace" and "Uniform" respectively indicate perturbations that are sampled from $N(0,1)$, $L(0,1)$, $U(0,1)$.

Type	Approach	Pubmed	Facebook	Coauthor-CS	Amazon-Com	Amazon-Pho	Coauthor-Phy
A	DropEdge	**81.34**	71.13	**92.35**	**74.65**	85.74	92.76
X	Gaussian	77.82	70.70	90.11	74.17	**86.26**	92.03
	Laplace	76.98	71.22	89.25	72.00	85.43	92.24
	Uniform	77.48	67.04	87.88	67.67	83.20	92.11
A+X	DropEdge+Gaussian	80.02	71.15	91.06	72.84	85.64	**92.91**
	DropEdge+ Laplace	77.58	**71.38**	90.56	72.05	86.00	92.69
	DropEdge+Uniform	77.68	69.67	89.48	58.65	76.54	92.60

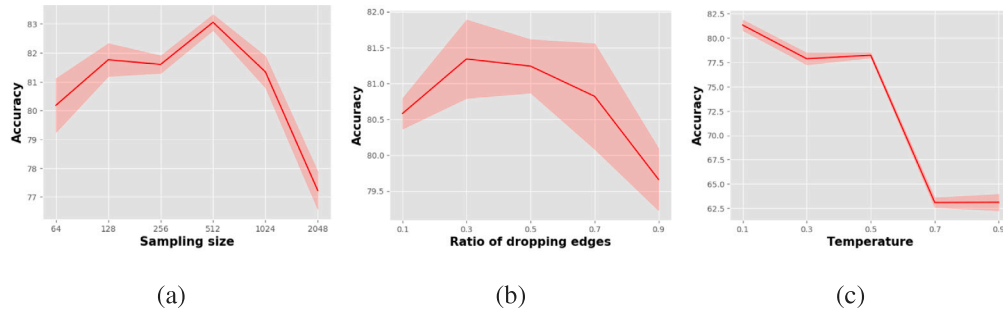| (a)	(b)	(c)

Fig. 7. The effect of different hyper-parameters on Pubmed. (a) indicates the sampling size K. (b) indicates the ratio ρ of dropping edges. (c) represents the temperature τ in the score function.

5. Conclusion and future work

In this paper, we propose a novel model SIGNNAP which illustrates the necessary properties of reliable node representations against perturbations on graphs. Apart from the widely-used *smoothness* property, SIGNNAP contains the *stability* and *identifiability* property, which provides a new insight of learning high-quality node representations for numerous graph algorithms. Through extensive experiments on various benchmarks, we show that SIGNNAP prevents the over-smoothing and over-fitting issue and learns more reliable node representations for downstream classification task.

However, there are still some limitations of our method. For example, the sampling method of negative pairs is random sampling, which would lead to the risk of sampling structure-similar representations of the anchor node. In future, we would explore how to revise the negative sampling distribution and correct the sampling bias.

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

I have shared the link to my data and code.

Acknowledgments

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patcog.2023.109976.

References

[1] 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, KDD '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 701–710.

[2] 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, ACM, 2016, pp. 855–864.

[3] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005, Vol. 2, 2005, pp. 729–734.

[4] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.

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

[6] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks 1 (2), 2017, arXiv preprint arXiv:1710.10903.

[7] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.

[8] 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 and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.

[9] S. Verma, Z.-L. Zhang, Stability and generalization of graph convolutional neural networks, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1539–1548.

[10] Y. Rong, W. Huang, T. Xu, J. Huang, DropEdge: Towards deep graph convolutional networks on node classification, in: International Conference on Learning Representations, 2019.

[11] 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, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

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

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018, accepted as poster.

[15] R. Zheng, W. Chen, G. Feng, Semi-supervised node classification via adaptive graph smoothing networks, Pattern Recognit. 124 (2022) 108492.

[16] G. Li, M. Muller, A. Thabet, B. Ghanem, Deepgcns: Can gcns go as deep as cnns? in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9267–9276.

[17] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, X. Lin, Topology attack and defense for graph neural networks: An optimization perspective, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI '19, AAAI Press, 2019, pp. 3961–3967.

[18] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song, Adversarial attack on graph structured data, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 1115–1124.

[19] P. Veličković, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, 2019.

[20] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: Proceedings of the 37th International Conference on Machine Learning, ICML '20, JMLR.org, 2020.

[21] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, J. Huang, Graph representation learning via graphical mutual information maximization, in: Proceedings of the Web Conference 2020, ACM, 2020, pp. 259–270.

[22] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, J. Tang, GCC: Graph contrastive coding for graph neural network pre-training, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1150–1160.

[23] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep Graph Contrastive Representation Learning, in: ICML Workshop on Graph Representation Learning and beyond, 2020.

[24] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 2069–2080.

[25] S. Thakoor, C. Tallec, M.G. Azar, R. Munos, P. Veličković, M. Valko, Bootstrapped representation learning on graphs, in: ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

[26] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., Bootstrap your own latent-a new approach to self-supervised learning, Adv. Neural Inf. Process. Syst. 33 (2020) 21271–21284.

[27] J. Zbontar, L. Jing, I. Misra, Y. LeCun, S. Deny, Barlow twins: Self-supervised learning via redundancy reduction, in: International Conference on Machine Learning, PMLR, 2021, pp. 12310–12320.

[28] P. Bielak, T. Kajdanowicz, N.V. Chawla, Graph barlow twins: A self-supervised representation learning framework for graphs, 2021.

[29] Z. Luo, Y. Dong, Q. Zheng, H. Liu, M. Luo, Dual-channel graph contrastive learning for self-supervised graph-level representation learning, Pattern Recognit. 139 (2023) 109448.

[30] J. Ding, R. Cheng, J. Song, X. Zhang, L. Jiao, J. Wu, Graph label prediction based on local structure characteristics representation, Pattern Recognit. 125 (2022) 108525.

[31] M. Chen, K.Q. Weinberger, Z. Xu, F. Sha, Marginalizing stacked linear denoising autoencoders, J. Mach. Learn. Res. 16 (1) (2015) 3849–3875.

[32] E. Hrushovski, Extending partial isomorphisms of graphs, Combinatorica 12 (4) (1992) 411–416.

[33] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, Eur. Comput. Vis. Assoc. (2020).

[34] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, J. Complex Netw. 9 (2) (2021).

[35] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, in: Relational Representation Learning Workshop, NeurIPS 2018, 2018.

[36] J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 43–52.

[37] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, ICML, 2018.

[38] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, KGAT: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &Amp; Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 950–958.

[39] L. Chen, S. Gong, J. Bruna, M.M. Bronstein, Attributed random walk as matrix factorization, in: Graph Representation Learning Workshop NeurIPS, 2019.

[40] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, J. Tang, Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, ACM, 2018, pp. 459–467.

Xu Chen received a Ph.D. degree at Cooperative Meidianet Innovation Center in Shanghai Jiao Tong University in 2021. He also received a dual Ph.D. degree at University of Technology, Sydney in 2022. He has published papers of top conferences and journals such as CVPR, IEEE TPAMI and ACM TOIS. He is now working at Alibaba Group. His research interests include machine learning, graph representation learning, self-supervised learning and recommendation systems

Yuangang Pan is working as a research scientist at A*STAR Centre for Frontier AI Research. He completed his Ph.D. degree in Computer Science in Mar 2020 from University of Technology Sydney (UTS), Australia. Before joining A*STAR, he was a postdoctoral research associate at the Australian Artificial Intelligence Institute at UTS. He has authored or co-authored papers on various top conferences and journals, such as AAAI, IEEE TIFS, IEEE TKDE, IEEE TNNLS, ACM TOIS, MLJ, and JMLR. His research interests include Deep Clustering, Deep Generative Learning, Differential Privacy, and Robust Ranking Aggregation.

Ivor W. Tsang (Fellow IEEE) is currently the Director of the A*STAR Centre for Frontier AI Research, Singapore. He is also a Professor of artificial intelligence with the University of Technology Sydney, Ultimo, NSW, Australia, and the Research Director of the Australian Artificial Intelligence Institute. His research interests include transfer learning, deep generative models, learning with weakly supervision, Big Data analytics for data with extremely high dimensions in features, samples and labels.

In 2013, he was the recipient of the ARC Future Fellowship for his outstanding research on Big Data analytics and large-scale machine learning. In 2019, his JMLR paper Towards ultrahigh dimensional feature selection for Big Data was the recipient of the International Consortium of Chinese Mathematicians Best Paper Award. In 2020, he was recognized as the AI 2000 AAAI/IJCAI Most Influential Scholar in Australia for his outstanding contributions to the field between 2009 and 2019. Recently, he was conferred the IEEE Fellow for his outstanding contributions to large-scale machine learning and transfer learning. He serves as the Editorial Board for the JMLR, MLJ, JAIR, IEEE TPAMI, IEEE TAI, IEEE TBD, and IEEE TETCI. He serves/served as a AC or Senior AC for NeurIPS, ICML, AAAI and IJCAI, and the steering committee of ACML.
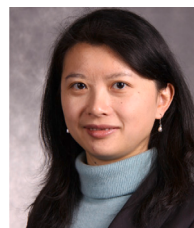
Ya Zhang received the B.S. degree from Tsinghua University and the Ph.D. degree in information sciences and technology from the Pennsylvania State University. Since March 2010, she has been a professor with Cooperative Medianet Innovation Center, Shanghai Jiao Tong University. Prior to that, she worked with Lawrence Berkeley National Laboratory, University of Kansas, and Yahoo! Labs. Her research interest is mainly on data mining and machine learning, with applications to information retrieval, web mining, and multimedia analysis. She is a member of the IEEE.