Neurocomputing 456 (2021) 88–96

Contents lists available at ScienceDirect

Neurocomputing

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

Self-supervised graph representation learning via bootstrapping

Feihu Che [a,b], Guohua Yang [a], Dawei Zhang [a], Jianhua Tao [a,b,c,*], Tong Liu [a]

[a] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[c] CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China

ARTICLE INFO

Article history:
Received 29 November 2020
Revised 12 March 2021
Accepted 29 March 2021
Available online 6 April 2021
Communicated by Zidong Wang

Keywords:
Graph representation learning
Self-supervised
Bootstrapping
Graph neural network

ABSTRACT

Graph neural networks (GNNs) apply deep learning techniques to graph-structured data and have achieved promising performance in graph representation learning. However, existing GNNs rely heavily on labeled data or well-designed negative samples. To address these issues, we propose a new self-supervised graph representation method: deep graph bootstrapping (DGB). DGB consists of two neural networks: online and target networks, and the input of them are different augmented views of the initial graph. The online network is trained to predict the target network while the target network is updated with a slow-moving average of the online network, which means the online and target networks can learn from each other. As a result, the proposed DGB can learn graph representation without negative examples in an unsupervised manner. In addition, we summarize three kinds of augmentation methods for graph-structured data and apply them to the DGB. Experiments on the benchmark datasets show the DGB performs better than the current state-of-the-art methods and how the augmentation methods affect the performances.

1. Introduction

Graph neural networks (GNNs) have made remarkable advancements in representation learning for graph-structured data [1–3]. Combining modeling the rich topology of graphs and unparalled expressive ability of deep learning, GNNs learn low-dimensional embeddings from variable-size and permutation-invariant graphs. The success of GNNs has also benefited a wide range of applications, such as in social networks [1], molecules [4], robot designs [5] and knowledge graphs [6].

Like other deep learning methods, many existing GNNs and their variants are mainly based on semi-supervised setting so they need a certain number of labeled data. However, requiring enough quality labeled data may meet some challenges in the real application scenarios. For instance, biology graphs represent specific concepts [7], so it is difficult and expensive to annotate the graphs; in addition, the reliability of the given labels may sometimes be questionable [8].

Though the present unsupervised graph representation algorithms do not need labels, they rely heavily on negative samples. For example, random walk-based methods [9,10] consider node pairs that are "close" in the graph are positive samples, meanwhile, take node pairs that are "far" in the graph as negative samples. The loss function lets "close" node pairs have more similar representations than "far" node pairs. In addition, deep graph informax (DGI) [11] maximizes mutual information between local representations and corresponding graph-level representations while taking the corruption graph as negative samples. The performances of these methods are highly dependent on the choices of negative samples, but in the wild negative pairs are not easy or computationally expensive to acquire. Consequently, how to obtain high-quality graph representation without supervision or negative examples becomes necessary for a number of practical applications, which motivates the study of this paper.

Recently, BYOL (bootstrap your own latent) [12] introduces bootstrapping mechanism to visual representation learning and learns from its previous version, which has achieved state-of-the-art results without negative examples. Nevertheless, BYOL is based on image data, and to the best knowledge of us, no work has applied bootstrapping mechanism to graph-structure data. In this work, we extend BYOL to graph-structured data and propose deep graph bootstrapping (DGB). DGB relies on two neural networks: online and target networks. During one time training, the target network is fixed, then the online network is updated by gradient descent to predict the target network; after training, the target network parameters are updated with a slow-moving average of the

* Corresponding author at: National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
E-mail address: jhtao@nlpr.ia.ac.cn (J. Tao).

https://doi.org/10.1016/j.neucom.2021.03.123
0925-2312/

online network parameters. Such training mechanism can make the online and target networks learn from each other, hence, DGB no longer needs labeled data or negative examples.

Data augmentations play an important role in DGB, but how to design efficient augmentation methods for graph-structured data is still challenging. In this paper, we systematically summarize three kinds of augmentation methods: node augmentation, including node feature dropout and node dropout; adjacent matrix augmentation, including personalized PageRank (PPR) [13] and heat kernel [14]; and the combination of them. We also apply these augmentation methods to DGB and show how augmentation methods help to improve the performances of DGB.

In conclusion, we propose a new unsupervised graph representation learning method without negative examples and systematically summarize three augmentation methods for graph-structured data. The main contributions of this paper are as follows:

- We first generalize bootstrapping mechanism to graph-structured data and propose an unsupervised graph representation learning method DGB without negative examples.
- The experimental results of the benchmark datasets show the DGB model is superior to the present supervised and unsupervised graph representation models.
- We systematically summarize three kinds of augmentation methods for graph-structured data, and apply them to DGB then analyze how data augmentations affect the performances of DGB.

2. Related works

Graph Representation Learning To mine the rich semantic and structure information in real-world graphs and networks, graph representation learning methods are proposed to embed nodes or edges into low-dimensional vectors on the basis of preserving the graph structure information. There are two classical kinds of models for graph representations: random-walk based models and graph neural network models. Random walk-based methods [9,10,15] generate random walks across nodes, and then apply neural language models to get network embedding. They emphasize proximity information while ignoring structural information [11,16]. GNNs apply deep learning techniques to non-Euclidean data, and the main idea of GNNs is to combine the neighborhood node information with the center node, which helps to perserve the graph structure [17–19].

Target Networks Target networks have a wide range of applications in deep reinforcement learning [20]. As one of the two important components of deep Q-network [21], target networks make the training process more stable and alleviate oscillations or divergence. In deep Q-network, the target network $\widehat{Q}$ is obtained by copying the network Q every C updates, and $\widehat{Q}$ is used to get the Q-learning targets for the following C updates. Target networks are extended to soft target updates, rather than directly copying the weights in [22], as a result, the target values have to change slowly, which can improve the stability of learning.

Self-supervised Learning Unlike supervised learning, self-supervised learning utilizes input data itself as supervision [23]. To be specific, self-supervised learning can obtain labels from the input data via a semi-automatic process and predicts other part of the data from the known parts. Many self-supervised learning models can be divided into two categories: generative or discriminative [24,25]. Generative models are to build a distribution over the data and latent representations and contain auto-regressive models [26,27], flow-based models [28,29] and auto-encoding models [30,31]. However, the weakness of generative models is requiring lots of computing resources. As the recently popular discriminative models, contrastive models employ a scoring function that enforces a higher score on positive pairs and a lower score on negative pairs [11,32]. Some recent works adapt contrastive ideas in image representation learning to unsupervised graph learning [11,7]. Deep graph informax [11] extends deepInfomax(DIM) [33] and contrasts node and graph embedding to learn node embeddings; in addition, InfoGraph extends DIM to learn graph embeddings.

Bootstrapping Methods Different from contrastive methods' requiring many negative examples to work well [34,25], bootstrapping methods can learn representations without negative examples in an unsupervised manner. DeepCluster [35] produces targets for the next representation by bootstrapping the previous representation; it clusters data points based on the prior representation and uses the clustered index of each example as a classification target to train the new representation. Predictions of Bootstrapped Latents(PBL) [36] apply bootstrapping methods to multitask reinforcement learning. PBL predicts latent embeddings of future observations to train its representations, and the latent embeddings are themselves trained to be predictive of the aforementioned representations. BYOL [12] uses two neural networks, known as online and target networks; the outputs of the target network serve as targets to train the online network and the parameters of the target network can be updated by weighted summation between the previous target network parameters and the online network parameters.

3. Unsupervised graph representation learning

A graph can be represented as $\mathcal{G} = \{\mathbf{X}, \mathbf{A}\}$, where $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ represents the node features, n is the number of nodes in the input graph and $\vec{x}_i \in \mathbb{R}^d$ means the feature vector of node i; $\mathbf{A} \in \mathbb{R}^{n\times n}$ is an adjacency matrix, $A_{ij} = 1$ represents there exists an edge from node i to node j and $A_{ij} = 0$ otherwise.

The objective of DGB is to learn an encoder, $\mathcal{E} : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times d'}$, as a result we can get $\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) = \left\{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n\right\}, \vec{h}_i \in \mathbb{R}^{d'}$ represents the final learned embedding for node i, which is for the downstream tasks, such as node classification or node cluster.

4. Deep graph bootstrapping

In this section, we introduce the proposed DGB model in detail. DGB model is inspired by BYOL [36], which learns visual representations by bootstrapping the latent representations. The DGB model learns node embeddings by predicting previous versions of its outputs, without leveraging negative examples. As is shown in figure1, DGB refers to two neural networks, online network and target network. We train the online network by predicting the target network output, and update the target network by an exponential moving average of the online network [22]. To be specific, DGB model consists of the following components:

- Graph convolution network as graph encoders to get node representations.
- Data augmentations for graph-structured data including node augmentation and graph diffusion.
- A bootstrapping mechanism to make the online network and target network learn from each other.

4.1. Graph neural network encoder

GNNs learn representations through transforming and aggregating from topological neighbors iteratively. In this paper, for

simplicity and generalization, we opt for the commonly used graph convolution network(GCN) [1] as our encoders:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{1}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the input graph $\mathscr{G}$ with added self-connections, I_N is the identity matrix and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix, $H^{(l)} \in \mathbb{R}^{N\times D}$ is the representation in l^{th} layer, $\sigma(\cdot)$ is a non-linear activation function, $W^{(l)}$ is a trainable weight matrix, which is our final goal to learn.

4.2. Graph-structured data augmentations

Recent successful self-supervised learning approaches in visual domain learn representations by contrasting congruent and incongruent augmentations of images [12]. Nevertheless, unlike images with standard augmentation methods, how to get effective augmentation methods for graph-structured data has no consensus. Considering there are actually two kinds of information in the graph: node information and adjacent information, in this paper, we introduce three kinds of graph data augmentations: node augmentation and graph diffusion network for adjacent matrix augmentation and the combination of them.

4.2.1. Node augmentation

For the node information, considering feature matrix $\mathrm{X} \in \mathbb{R}^{n\times d}$, n is the number of nodes and d is the dimensionality of features. Following [37] we use node dropout (ND) and node feature dropout (NFD): node dropout denotes randomly zeroes one node's entire features with a pre-defined probability, i.e. dropping the row vectors of X, randomly; while node feature dropout means randomly discards each element of X. The specific formulation of ND and NFD are

$$\tilde{\mathrm{X}}_i = \frac{\tilde{\epsilon}_i}{1-\delta_{nd}}\mathrm{X}_i \tag{2}$$

$$\tilde{\mathrm{X}}_{ij} = \frac{\tilde{\epsilon}_{ij}}{1-\delta_{nfd}}\mathrm{X}_{ij} \tag{3}$$

where δ_{nd} and δ_{nfd} are the dropout probability of ND and NFD respectively, $\tilde{\epsilon}_i$ and $\tilde{\epsilon}_{ij}$ seperately draws from *Bernoulli*(1 - δ_{nd}), *Bernoulli*(1 - δ_{nfd}). The factor $1/(1-\delta_{nd})$ and $1/(1-\delta_{nfd})$ are to make the perturbed feature matrix $\tilde{\mathrm{X}}$ equal to X in expectation. To note that, NFD and ND are only used during training. After training, we use initial node features to calculate representations.

4.2.2. Adjacent matrix augmentation

For the adjacent matrix augmentation, we consider graph diffusion networks [38]. In previous GNNs, there exists one obvisious problem: edges are often not clean enough or defined with an inappropriate threshold [39]. Graph diffusion networks(GDN) [38] are proposed to tackle the problem. GDN combines spatial message passing with a sparsified form of graph diffusion which can be regarded as an equivalent polynomial filter.

For a graph $\mathscr{G} = \{\mathbf{X}, \mathbf{A}\}$, the generalized graph diffusion is formulated as:

$$S = \sum_{k=0}^{\infty}\theta_k\mathbf{T}^k \tag{4}$$

where $\mathbf{T} \in \mathbb{R}^{n\times n}$ denotes the generalized transition matrix, θ_k is the weighting coefficient which determines the ratio of global–local information. In order to guarantee convergence, two conditions are considered: $\sum_{k=0}^{\infty}\theta_k = 1, \theta_k \in [0,1]; \lambda_i \in [0,1]$ where λ_i are eigenvalues of $\mathbf{T}$.

There are two classic examples of the graph diffusion: personalized PageRank (PPR) [13] and heat kernel [14]. For an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and a degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$, $\mathbf{T}$ in Eq. (4) is defined as $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$, $\theta_k = \alpha(1-\alpha)^k$ for PPR and $\theta_k = e^{-t}t^k/k!$ for heat kernel, where $\alpha \in (0,1)$ is teleport probability and t is the diffusion time. The specific formulation is showed in Eqs. (5) and (6) [16]:

$$\mathbf{S}^{\text{heat}} = \exp\left(t\mathbf{A}\mathbf{D}^{-1} - t\right) \tag{5}$$

$$\mathbf{S}^{\text{ppr}} = \alpha\left(\mathbf{I}_n - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \tag{6}$$

In the "augmentation" part of Fig. 1, we apply NFD to the first view, and NFD + graph diffusion to the second view. Comparing with the initial graph, we can easily see how these augmentation methods work. For simplicity, we do not scale the augmented node features in Fig. 1. We apply different combinations of the augmentation methods above to DGB, and the results and discussions will be in Section 5.

4.3. Bootstrapping process

The bootstrapping process is the core of the DGB model. As is shown in Fig. 1, there are three steps in the online network: one graph convolution network layer called GNN_θ, one multilayer perceptron for projection $\mathrm{MLP}_\theta^{\text{pro}}$, and one multilayer perceptron for prediction $\mathrm{MLP}_\theta^{\text{pre}}$; similar to the online network, the target network has corresponding GNN_ε and $\mathrm{MLP}_\epsilon^{\text{pro}}$. For simplicity, in the following, we use θ to denote the parameters of $\mathrm{GNN}_\theta, \mathrm{MLP}_\theta^{\text{pro}}$, and use ε to denote the parameters of $\mathrm{GNN}_\varepsilon, \mathrm{MLP}_\epsilon^{\text{pro}}$. The reason for using a multilayer perceptron for projection has been proved to improve performances [25], and we will have a further discussion in the ablation study.

During each epoch training, we first fix the parameters of the target network, and the regression loss between the online network outputs $q(z)$ and the target network outputs Z' are used to update the online network parameters. After one epoch training, the target network parameters ε are updated using an exponential moving average of the θ [22,36]. Specifically, after one epoch training, for a given target decay rate $p \in [0,1]$, this paper uses the following updating process:

$$\xi \leftarrow p\xi + (1-p)\theta \tag{7}$$

when $p = 1$, the target network will be never updated and remains at a constant value; when $p = 0$, the target network will be updated to the online network at each step. Therefore, we need a trade-off value for p to update the target network at a proper speed.

4.4. Training method

As is shown in Fig. 1, the loss function in DGB is the mean squared error between online network predictions $q(z)$ and target network projections z'. Before calculating the error, we first ℓ_2-normalize the predictions and projections:

$$\overline{q}(z) \triangleq q(z)/\|q(z)\|_2 \tag{8}$$

$$\overline{z}' \triangleq z'/\|z'\|_2 \tag{9}$$

The loss function is:

$$\mathscr{L}_\theta^{\text{DGB}} \triangleq \|\overline{q}(z) - \overline{z}'\|_2^2 \tag{10}$$

After getting $\mathscr{L}_\theta^{\text{DGB}}$, we seperately feed the second view augmentation to the online network and the first view augmentation to the target network to compute $\tilde{\mathscr{L}}_\theta^{\text{DGB}}$. During each epoch

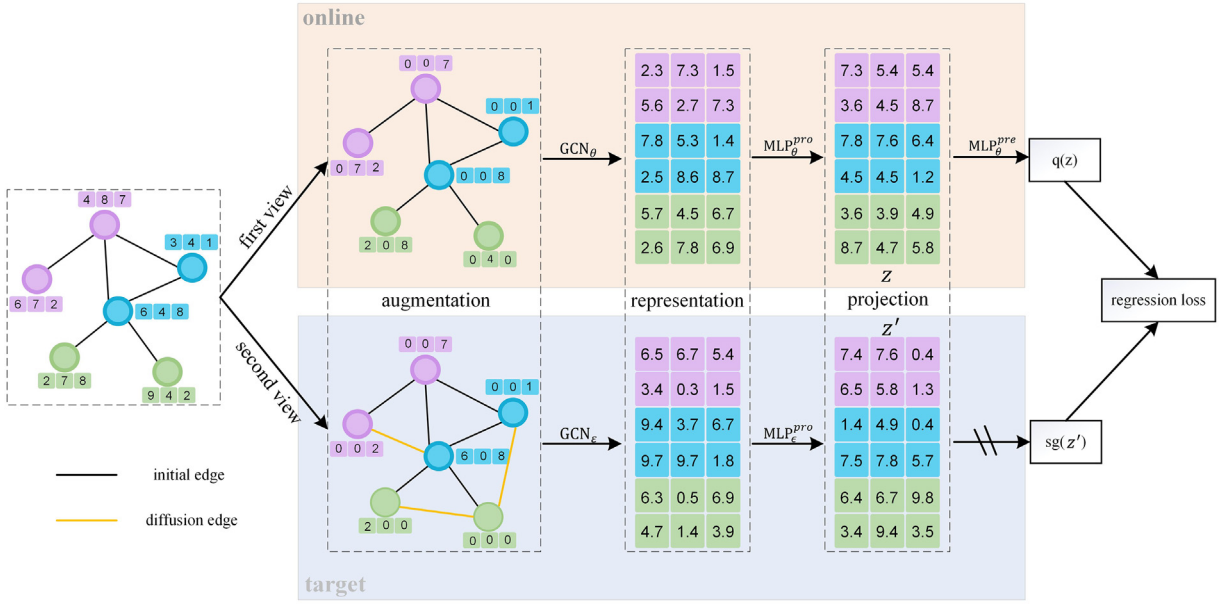

Fig. 1. The proposed deep graph bootstrapping model for graph representation learning. The model consists of online network and target network. The input of online and target networks are two augmented views of the initial graph. We feed the first view to a graph convolution network (GCN) in order to obtain node representations, then the representations are fed to two MLPs successively to get the projections and predictions; for the target network, the second view is fed to a GCN and one MLP to get representations and projections. The objective function of the model is to minimize the regression loss between the predictions of the online network and the projections of the target network. To note that, the parameters of the target network are not updated by gradient descent. After training, we only use the GCN layer of the online network to get node representations for downstream tasks.

training, $\mathscr{L}_{\theta}^{\mathrm{DGB}} + \tilde{\mathscr{L}}_{\theta}^{\mathrm{DGB}}$ is used to minimize via stochastic optimization with respect to the parameters of $\mathrm{GNN}_{\theta}, \mathrm{MLP}_{\theta}^{\mathrm{pro}}$ and $\mathrm{MLP}_{\theta}^{\mathrm{pre}}$.

After training, we only keep the GNN layer of the online network GNN_{θ}, and use GNN_{θ} to compute the representations for downstream tasks.

5. Experiments

5.1. Datasets

In this section, we implement experiments to prove the effectiveness of the DGB model. The datasets in our experiments are three standard benchmark citation network datasets, namely Cora, Citeseer, and Pubmed [40]. In the three datasets, nodes represent documents, edges correspond to citations, and each node has a feature vector corresponding to the bag-of-words representation. Each node can be divided into one of the several classes. More details about the three datasets are in Table 1. Following the experimental setup in [1], we use the same data splits.

5.2. Evaluation protocol

We train the DGB model to get each node's low dimensional representation based on the node features and the interactions between nodes. After the training process of the DGB model, we use a linear evaluation to prove the effectiveness of the learned representations. Following DGI [11], we show the mean classification accuracy on the test nodes after 50 runs of training followed by a linear model.

To note that, the standard deviation of 50 runs in our experiments is very small so we don't list it in Table 2.

5.3. Baselines

To comprehensively evaluate the proposed DGB model, we compare it with six supervised methods and eight unsupervised methods in Table 2.

- LP [41] combines a Gaussian random field model and a weighted graph representing labeled and unlabeled data for semi-supervised learning.
- PLANETOID [42] proposes joint training of class label prediction and neighborhood context prediction for each node.
- CHEBYSHEV [43] proposes a formulation of convolutional neural networks with the help of spectral graph theory.
- GCN [1] generalizes traditional convolution network to graph structure data.
- GAT [3] puts attention mechanism into GCN.
- GWNN [44] leverages graph wavelet transform instead of graph fourier transform.
- RAW FEATURES method trains the node features using a logistic regression classifier and gives the results on the test node features [8].
- DEEPWALK [9] learns representations via random walks and skip-gram models on graphs.

Table 1
Statistics of benchmark datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	5429	1433	7
Citeseer	3327	4732	3703	6
Pubmed	19717	44338	500	3

Table 2
Mean classification accuracy in percent for supervised and unsupervised models on three benchmark datasets. We list the data available to each model during training in the third column. X, A, and Y represent node features, adjacency matrix and node labels, respectively.

METHOD	Available Data	Cora	Citeseer	Pubmed
LP	A, Y	68.0	45.3	63.0
PLANETOID	X, Y	75.7	62.9	75.7
CHEBYSHEV	X, A, Y	81.2	69.8	74.4
GCN	X, A, Y	81.5	70.3	79.0
GAT	X, A, Y	83.0± 0.7	72.5±0.7	79.0±0.3
GWNN	X, A, Y	82.8	71.7	79.1
RAW FEATURES	X	56.6±0.4	57.8±0.2	69.1±0.2
DEEPWALK	X, A	70.7±0.6	51.4±0.5	74.3±0.9
EP-B	X, A	78.1±1.5	71.0±1.4	79.6±2.1
GMI-mean	X, A	82.7±0.2	73.0±0.3	80.1±0.2
GMI-adaptive	X, A	83.0±0.3	72.4±0.1	79.9±0.2
DGI	X, A	82.3±0.6	71.8±0.7	76.8±0.6
GMNN(with q_θ)	X, A	78.1	68.0	79.3
GMNN(with q_θ and p_ϕ)	X, A	82.8	71.5	81.6
DGB(ours)	X, A	**83.9**	**74.3**	**82.3**

- EP-B [45] learns node representations by passing label representations and gradients between neighboring nodes.
- GMI [8] maximizes the mutual information between node features and topological structure.
- DGI [11] maximizes mutual information between local patch representations and the global representation.
- GMNN [46] combines the advantages of graph neural network and the statistical relational learning to learn node representations.

The results of LP and DEEPWALK are taken from [1], and the results of RAW FEATURES and CHEBYSHEV are taken from [8,16] respectively. We take the results of other methods from their original papers.

5.4. Experiments setup

We implement all the experiments in Pytorch [47] on a single GPU Tesla K80 with 11 GB memory size. We use Glorot initialization [48] to initialize the parameters of the model. We also perform row normalization on the three datasets for preprocessing. For graph convolution encoder, we use one layer on the three datasets. The optimizer we use for training is Adam optimizer [49] and the initial learning rate is 0.001 for Cora and Pubmed, 0.0001 for Citeseer.

In DGB model, the data augmentation methods play an important role. Consequently, we conduct extensive experiments that apply different data augmentation combinations to DGB, and the results are listed in Table 3. In these experiments, after training, we only feed the initial node features and the adjacent matrix without any data augmentation to the GCN encoder then we get node representations for classification task.

5.5. Experiment analysis

From Table 2, it is apparent that DGB can perform the best among the recent state-of-the-art methods. For example, on Cora, DGB can improve GMI-adaptive by a margin 0.9%, and improve GMI-mean by a margin 1.3% on Citeseer.

We consider this improvement benefitting from two points: one is that the DGB model can learn from its previous version and does not need well-designed negative examples; the other is that node augmentation methods generate multiple node feature matrixes in different epochs, which alleviates overfitting to some extent and improves the DGB's robustness.

5.6. Results under different graph data augmentations

In order to demonstrate the relationship between different graph data augmentations and the performances of the DGB model, we combine different augmentations in DGB and show the results in Table 3. In node augmentation experiments, we only use the adjacent matrix; while in adjacent augmentation experiments, we use the initial node features without node dropout or node feature dropout. For fairness, the experiments on one dataset in Table 3 are under the same hyperparameters except for the dropout rate. We select the best dropout rate in node dropout and node feature dropout from 0.1 to 0.9 with step 0.1. After training, we feed the feature matrix and adjacent matrix without augmentation to get representations for node classification.

Experimental results in Table 3 show the performances under different graph data augmentations in DGB on Cora, Citeseer, and Pubmed. From Table 3, we obtain several observations as follows:

- The results on the three datasets show that node augmentation and adjacent augmentation can both improve the performances, and combining the two augmentations can achieve higher performances overall.
- For node augmentation, it is apparent to see NFD does better in improving performance than ND. We suspect randomly dropping one node's entire features may lose too much information and is difficult to predict for the other view.
- There is a clear trend: more combinations of data augmentations bring better performances. For node augmentation group, the combination of NFD & ND can beat the other combinations on the three datasets. Similarly, for node + adj augmentation group, the combination of NFD + ADJ & ND + DIFF and ND + ADJ & NFD + DIFF are better than other combinations.
- Since the graph diffusion brings more connections to the nodes, the combination of ND + DIFF does not hurt the graph information too much compared with ND + ADJ. As a result, the combination of NFD + ADJ & ND + DIFF is superior to the combination ND + ADJ & NFD + DIFF.
- Though node augmentation and adjacent augmentation can both benefit the model's performance, node augmentation is superior to adjacent augmentation. The results of NODE augmentation group are much better than ADJ augmentation group on all the three datasets. This may be because node augmentations can generate different node feature matrixes in each epoch because of the randomness of the Bernoulli distribution, while the diffusion matrix is the same across different epochs. The randomness improves the performance and robustness of the DGB model.

Table 3
Mean node classification accuracy under different combinations of data augmentations. In DGB model, two different augmentations of graph data learn from each other. For the first column, we list one initial graph data and three kinds of graph data augmentations, NO, NODE, ADJ, NODE + ADJ represent no augmentation, node augmentation, adjacent matrix augmentation, and the combination of node and adjacent matrix augmentation. For the second and third columns, we present the specific data augmentation combinations. IN, NFD, ND, ADJ, DIFF denote initial node, node feature dropout, node dropout, initial adjacent matrix, and diffusion matrix, respectively.

	First view	Second view	Cora	Citeseer	Pubmed
NO	IN + ADJ	IN + ADJ	68.4	60.7	68.0
NODE	IN	NFD	82.2	70.1	78.2
	IN	ND	79.8	69.1	78.7
	NFD	ND	82.6	71.1	79.6
	NFD	NFD	82.0	70.5	77.9
	ND	ND	79.0	68.4	78.0
ADJ	DIFF	ADJ	69.2	65.6	74.5
NODE + ADJ	IN + ADJ	NFD + DIFF	82.1	72.7	79.8
	IN + ADJ	ND + DIFF	80.5	66.6	79.6
	NFD + ADJ	IN + DIFF	82.8	72.5	78.9
	ND + ADJ	IN + DIFF	80.0	71.3	79.3
	ND + ADJ	ND + DIFF	80.9	70.1	78.4
	NFD + ADJ	NFD + DIFF	83.2	72.9	79.1
	ND + ADJ	NFD + DIFF	83.5	73.0	80.1
	NFD + ADJ	ND + DIFF	**83.9**	**74.3**	**82.3**

5.7. Ablation study

In this section, we consider the function of a multilayer perceptron for projection in our DGB model and how the bootstrapping mechanism helps the DGB model. We conduct experiments on the three datasets with and without a multilayer perceptron for projection. For analysizing the bootstrapping mechanism, we seperately let $p = 0$ and $p = 1$. When $p = 0$, the target network will copy the parameters of the online network after each epoch training; meanwhile the target network parameters will remain constant values when $p = 1$.

As is shown in Table 4, the performances of the DGB model with and without MLP have a big difference on the three datasets. Keeping the target network parameters constant or changing them to the online network parameters totally both hurt the performances.

We also visualize the node representations of the online network using t-sne [50] on Cora dataset in Fig. 2. In Fig. 2, there are four subgraphs: the representations of the DGB model, the DGB model without a projection layer, the DGB model with $p = 0$ and $p = 1$.

Comparing the four subgraphs, we can see different classes of the representations without the projection layer become tighter and are not easy to distinguish. When $p = 0$, the representations are better than the representations with $p = 1$, but they are both worse than the representations with slow-moving average mechanism. Not changing the target network parameters or changing them totally are both not the best choice, and there is a trade-off between the two choices.

5.8. Baselines with data augmentations

Data augmentations play an import role in the performances of the DGB model. To analyse how the data augmentations affect the performances in DGB and other baseline models, we apply the data augmentation methods, including node feature dropout, node dropout and graph diffusion, to five representative baseline methods: GCN [1], GAT [3], GMNN(with q_θ and p_ϕ) [46], GMI [8] and DGI [11]. To note that, in their release codes, GCN, GAT and GMNN all utilize node feature dropout, so we apply node dropout and graph diffusion to them. In the DGB model, data augmentation methods are only leveraged during the training process. To get the final node embeddings for node classification, we use the initial node features and graph structures. For fair comparsion, data augmentation methods are only used during training in the augmented GCN, GAT, GMI, GMNN and DGI. We first reproduce the results of the five models and change the dropout rate from 0.1 to 0.9 with step 0.1, then report the best results among different dropout rates.

Table 4
Ablation study results

	Cora	Citeseer	Pubmed
DGB	83.9	74.3	82.3
without projection	74.7	71.7	70.7
$p = 0$	80.9	71.9	80.0
$p = 1$	75.9	70.2	74.9

From Tables 5–9, we can see the influence of data augmentations on DGB model and baseline methods are totally different. For the five baseline methods, data augmentation methods have limited improvement in several cases, and they hurt the performances in most cases. However, from Table 3, data augmentation methods can efficiently improve the performances in a regular manner.

This is due to the fact that different methods learn representations based on different internal mechanisms. For instance, for supervised models such as GCN and GAT, it is the label information that supervises models to learn discriminative representations. The core of DGI is maximizing the mutual information between the high-level summaries and patch representations, which helps DGI to learn efficient representations. For GMNN, it treats neighborhood nodes as pseudo labels and models the joint distribution of node labels. For GMI, it is mainly maximizing the mutual information between the input of node features and topological structure and the corresponding output. When applying node dropout and node feature dropout to GMI, we find it is hard to converge, and the performances drop heavily.

However, many self-supervised learning methods are sensitive to the choice of data augmentations. For instance, SimCLR [25] and BYOL [12] both have a big performance decrease when improper data augmentation methods are selected. As for DGB, the main point is to let the representations of one augmented view predictive of another augmented view of the same graph. Based on the mutual predictions of different augmented views of the same graph, effective representations can be learned. As a result, data augmentations play a more important role in the performances than other baseline methods.

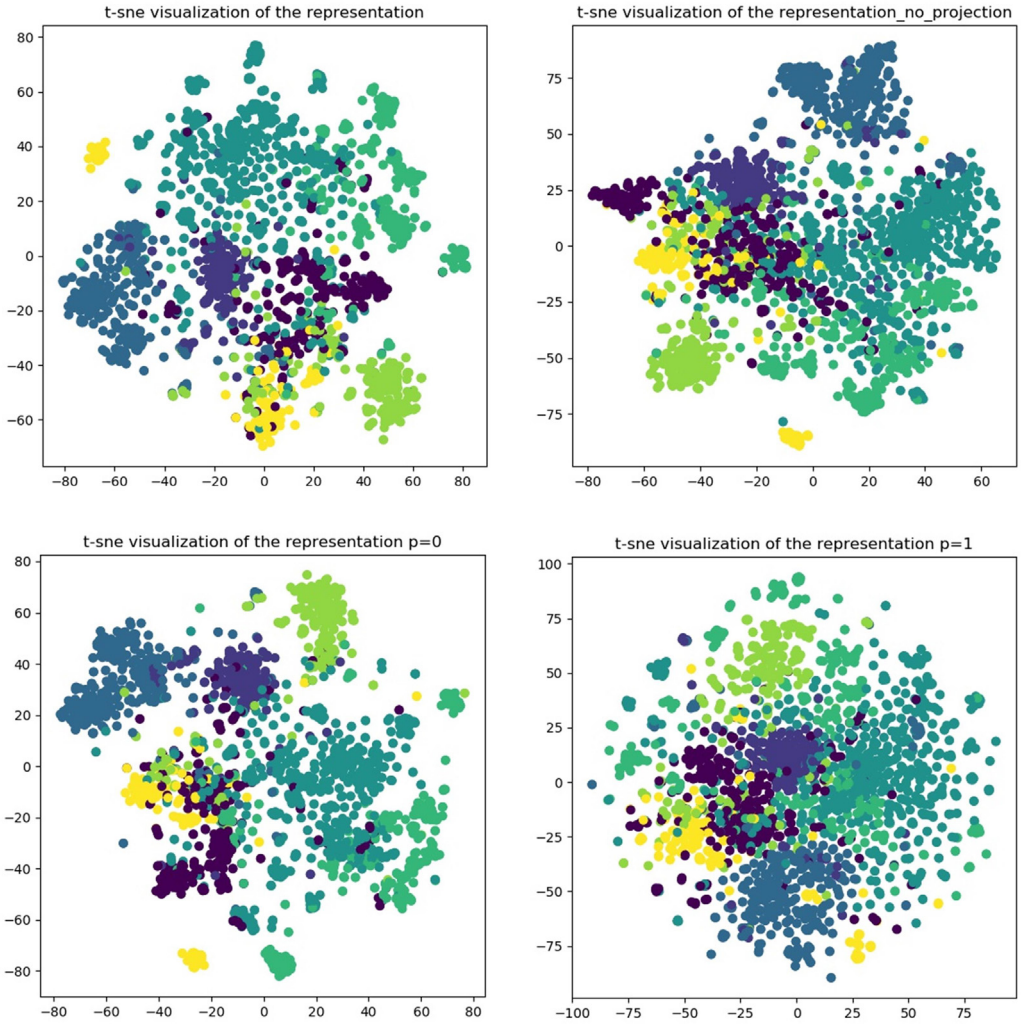

Fig. 2. The visualization of the representations on Cora dataset.

Table 5
The performances of GCN model with data augmentations

	Cora	Citeseer	Pubmed
Report	81.5	70.3	79.0
Reproduce	81.7	70.6	79.0
Add ND	83.1	71.4	78.2
Add Diffusion	81.6	65.4	78.9
Add Diffusion & ND	82.8	70.0	78.7
DGB	83.9	74.3	82.3

Table 6
The performances of GAT model with data augmentations

	Cora	Citeseer	Pubmed
Report	83.0	72.5	79.0
Reproduce	82.9	72.5	79.0
ADD ND	83.5	72.4	78.8
Add Diffusion	71.3	63.8	73.5
Add Diffusion & ND	72.4	64.8	74.2
DGB	83.9	74.3	82.3

Table 7
The performances of GMNN(with q_θ and p_ϕ) model with data augmentations

	Cora	Citeseer
Report	82.8	71.5
Reproduce	82.7	71.7
Add ND	82.2	70.3
Add Diffusion	79.2	57.6
Add Diffusion & ND	78.3	60.0
DGB	83.9	74.3

Table 8
The performances of GMI-mean model with data augmentations

	Cora	Citeseer
Report	82.7	73.0
Reproduce	82.8	72.0
Add ND	31.9	23.1
Add NFD	31.9	23.1
Add Diffusion	79.8	70.7
Add Diffusion & NFD	15.7	23.1
Add Diffusion & ND	31.9	23.1
Add Diffusion & NFD & ND	31.9	23.1
DGB	83.9	74.3

Table 9
The performances of DGI model with data augmentations

	Cora	Citeseer	Pubmed
Report	82.3	71.8	76.8
Reproduce	82.3	71.8	77.0
Add NFD	83.7	72.9	77.9
Add ND	79.6	64.1	71.4
Add NFD & ND	80.0	63.6	71.3
Add Diffusion	82.2	73.2	76.3
Add Diffusion & NFD	81.8	73.3	76.4
Add Diffusion & ND	80.5	66.2	71.7
Add Diffusion & NFD & ND	80.5	70.9	71.5
DGB	83.9	74.3	82.3

6. Conclusion

In this paper, we introduce a new self-supervised graph representation learning method DGB. DGB relies on two neural networks: online network and target network, and the input of each neural network is an augmentation of the initial graph. With the help of the bootstrapping process, the online network and target network can learn from each other. As a result, DGB does not need negative examples and can learn representations in an unsupervised manner. Experiments on three benchmark datasets show DGB is superior to state-of-the-art methods. In addition, we systematically conclude different graph data augmentation methods: node augmentations, adjacent matrix augmentations and the combination of them. We also apply different data augmentation types to DGB and experiment results show how different augmentation methods affect the performances of the DGB. As we only apply DGB to the homogeneous network, we will focus on how to generalize the DGB model to heterogenous information networks in the future.

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.

CRediT authorship contribution statement

Feihu Che: Conceptualization, Methodology, Validation, Writing - original draft, Writing - review & editing, Visualization. **Guohua Yang:** Writing - original draft, Supervision, Data curation, Validation. **Dawei Zhang:** Formal analysis, Resources, Writing - review & editing. **Jianhua Tao:** Project administration, Funding acquisition. **Tong Liu:** Writing - review & editing, Validation.

Acknowledgement

This work is supported by the National Key Research and Development Plan of China (No. 2018YFB1005003), the National Natural Science Foundation of China (NSFC) (No. 61831022, No. 61901473, No. 61771472, No. 61773379).

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907..
[2] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, arXiv preprint arXiv:1704.01212..
[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903..
[4] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Advances in Neural Information Processing Systems, 2015, pp. 2224–2232..
[5] T. Wang, Y. Zhou, S. Fidler, J. Ba, Neural graph evolution: Towards efficient automatic robot design, arXiv preprint arXiv:1906.05370..
[6] S. Vivona, K. Hassani, Relational graph representation learning for open-domain question answering, arXiv preprint arXiv:1910.08249..
[7] F.-Y. Sun, J. Hoffmann, J. Tang, Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization, arXiv preprint arXiv:1908.01000..
[8] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, J. Huang, Graph representation learning via graphical mutual information maximization, Proceedings of The Web Conference (2020) 259–270.
[9] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[10] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1067–1077.
[11] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax., in: ICLR (Poster), 2019..
[12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P.H. Richemond, E. Buchatskaya, C. Doersch, B.A. Pires, Z.D. Guo, M.G. Azar, et al., Bootstrap your own latent: A new approach to self-supervised learning, arXiv preprint arXiv:2006.07733..
[13] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, Tech. rep, Stanford InfoLab, 1999.
[14] R.I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in: Proceedings of the 19th International Conference on Machine Learning, vol. 2002, 2002, pp. 315–22..
[15] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[16] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, arXiv preprint arXiv:2006.05582..
[17] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, arXiv preprint arXiv:1812.08434..
[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems..
[19] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: A survey, IEEE Transactions on Knowledge and Data Engineering..
[20] H. Van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, J. Modayil, Deep reinforcement learning and the deadly triad, arXiv preprint arXiv:1812.02648..
[21] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[22] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971..
[23] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, J. Tang, Self-supervised learning: Generative or contrastive, arXiv preprint arXiv:2006.08218 1 (2)..
[24] C. Doersch, A. Gupta, A.A. Efros, Unsupervised visual representation learning by context prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1422–1430.
[25] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, arXiv preprint arXiv:2002.05709..
[26] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training..
[27] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, K. Kavukcuoglu, Conditional image generation with pixelcnn decoders, arXiv preprint arXiv:1606.05328..
[28] L. Dinh, D. Krueger, Y. Bengio, Nice: Non-linear independent components estimation, arXiv preprint arXiv:1410.8516..
[29] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real nvp, arXiv preprint arXiv:1605.08803..
[30] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.
[31] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805..
[32] Y. Li, C. Gu, T. Dullien, O. Vinyals, P. Kohli, Graph matching networks for learning the similarity of graph structured objects, arXiv preprint arXiv:1904.12787..
[33] R.D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, arXiv preprint arXiv:1808.06670..
[34] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[35] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep clustering for unsupervised learning of visual features, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 132–149.
[36] D. Guo, B.A. Pires, B. Piot, J.-B. Grill, F. Altché, R. Munos, M.G. Azar, Bootstrap latent-predictive representations for multitask reinforcement learning, arXiv preprint arXiv:2004.14646..

[37] Y.D.Y.H.H.L.Q.X.Q.Y.J.T. Wenzheng Feng, Jie Zhang, Graph random neural network, arXiv preprint arXiv:2005.11079..
[38] J. Klicpera, S. Weißenberger, S. Günnemann, Diffusion improves graph learning, Advances in Neural Information Processing Systems (2019) 13354–13366.
[39] Y.-H. Tang, D. Zhang, G.E. Karniadakis, An atomistic fingerprint algorithm for learning ab initio molecular force fields, The Journal of Chemical Physics 148 (3) (2018) 034101.
[40] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (3) (2008) 93.
[41] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.
[42] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, 2016, pp. 40–48.
[43] 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..
[44] B. Xu, H. Shen, Q. Cao, Y. Qiu, X. Cheng, Graph wavelet neural network, arXiv preprint arXiv:1904.07785..
[45] A.G. Duran, M. Niepert, Learning graph representations with embedding propagation, in: Advances in Neural Information Processing Systems, 2017, pp. 5119–5130..
[46] M. Qu, Y. Bengio, J. Tang, Gmnn: Graph markov neural networks, arXiv preprint arXiv:1905.06214..
[47] N. Ketkar, Introduction to pytorch, in: Deep Learning with Python, Springer, 2017, pp. 195–208..
[48] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[49] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980..
[50] L. v. d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (Nov) (2008) 2579–2605..

Feihu Che, born in 1996, he received the B.S. degree from School of Automation, China University of Geosciences in 2017. He is currently a PHD student in the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing. His main research covers graph representation learning, knowledge graph and data mining.

Guohua Yang received her Ph.D. degree from the Beijing University of Posts and Telecommunications in 2019. Now she is an assistant professor of National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. Her research interests include Natural Language Processing and Machine Learning. She is a member of the Human-Machine Interaction Committee of the Chinese Society of Image and Graphics, the Chinese Information Society of China, and the Natural Language Understanding Committee of China Artificial Intelligence Society.

Dawei Zhang received PhD from Institute of Automation, Chinese Academy of Sciences in 2017. He is now an associate professor of National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. His research interests include pattern recognition, natural language processing and knowledge reasoning. He has published tens of papers on MTA, AAAI, ICASSP, ICPR, JCAD, etc.

Jianhua Tao, received PhD from Tsinghua University in 2001. He is Winner of the National Science Fund for Distinguished Young Scholars and the deputy director in NLPR, CASIA. He has directed many national projects, including “863”, National Natural Science Foundation of China. His interests include speech synthesis, affective computing and pattern recognition. He has published more than eighty papers on journals and proceedings including IEEE Trans. on ASLP, and ICASSP, INTERSPEECH. He also serves as the steering committee member for IEEE Transactions on Affective Computing and the chair or program committee member for major conferences, including ICPR, Interspeech, etc.

Tong Liu received Phd from Shanghai Jiao Tong University in 2018. He is now the assistant professor in NLPR, CASIA. His interests include online big data analysis, decision support systems, knowledge graph, and natural language processing. He has published many papers on core academic journals include Data Analysis and Knowledge Discovery, System Engineering Theory and Practice, Application Research of Computers etc.