Learning Robust Representations with Graph Denoising Policy Network

Lu Wang[1], Wenchao Yu[2*], Wei Wang[3], Wei Cheng[2], Wei Zhang[1], Hongyuan Zha[4], Xiaofeng He[1*], Haifeng Chen[2]
[1] School of Computer Science and Technology, East China Normal University
[2] NEC Laboratories America, Inc., [3]University of California Los Angeles, [4]Georgia Institute of Technology
[1]{luwang,xfhe}@stu.ecnu.edu.cn, zhangwei.thu2011@gmail.com, [2]{wyu,weicheng,haifeng}@nec-labs.com,
[3]weiwang@cs.ucla.edu, [4]zha@cc.gatech.edu

***Abstract*—Existing representation learning methods based on graph neural networks and their variants rely on the aggregation of neighborhood information, which makes it sensitive to noises in the graph, e.g. erroneous links between nodes, incorrect/missing node features. In this paper, we propose Graph Denoising Policy Network (short for GDPNet) to learn robust representations from noisy graph data through reinforcement learning. GDPNet first selects *signal* neighborhoods for each node, and then aggregates the information from the selected neighborhoods to learn node representations for the down-stream tasks. Specifically, in the *signal neighborhood selection* phase, GDPNet optimizes the neighborhood for each target node by formulating the process of removing noisy neighborhoods as a Markov decision process and learning a policy with task-specific rewards received from the *representation learning* phase. In the *representation learning* phase, GDPNet aggregates features from signal neighbors to generate node representations for down-stream tasks, and provides task-specific rewards to the *signal neighbor selection* phase. These two phases are jointly trained to select optimal sets of neighbors for target nodes with maximum cumulative task-specific rewards, and to learn robust representations for nodes. Experimental results on node classification task demonstrate the effectiveness of GDNet, outperforming the state-of-the-art graph representation learning methods on several well-studied datasets.**

***Index Terms*—graph representation learning, graph neural networks, graph embedding, reinforcement learning**

I. Introduction

Recently, remarkable progress has been made toward graph representation learning, a.k.a graph/network embedding, which solves the graph analytics problem by mapping nodes in a graph to low-dimensional vector representations while effectively preserving the graph structure [1]–[3]. Graph neural networks (GNNs) have been widely applied in graph analysis due to the ground-breaking performance with deep architectures and recent advances in optimization techniques [4], [5]. Existing representation learning methods based on GNNs, e.g. GraphSAGE [6], Graph Convolution Networks (GCNs) [7], [8] and Graph Attention Networks (GATs) [9], rely on the aggregation of neighborhood information, which makes the model vulnerable to noises in the input graph.

Good graph representations are expected to be robust to the erroneous links, mislabeled nodes and partial corrupted features in the input graph, and capture geometric dependencies among nodes in the graph. However existing approaches have limited efforts on robustness study in this regard. In order to overcome this limitation of graph representation learning in handling noisy graph data, we propose Graph Denoising Policy Network, denoted as GDPNet, to learn robust representations through reinforcement learning. GDPNet includes two phases: *signal neighbor selection* and *representation learning*. It first selects *signal* neighbors for each node, and then aggregates the information from the selected neighbors to learn node representations with respect to the down-stream tasks.

The major challenge here is on how to train these two phases jointly, particularly when the model has no explicit knowledge about where the noise might be. We address this challenge by formulating the graph denoising process as a Markov decision process. Intuitively, although we do not have an explicit supervision for the signal neighbor selection, we can measure the performance of the representations learned with the selected neighbors on tasks like node classification, then the task-specific rewards received from the representation learning phase can be used for trial-and-error-search. In the *signal neighbor selection* phase, GDPNet optimizes the neighborhood for each node by formulating the process of removing the noisy neighbors as a Markov decision process and learning a policy with the task-specific rewards received from the *representation learning* phase. In the *representation learning* phase, GDPNet trains a set of aggregator functions that accumulate feature information from the selected signal neighbors of each target node. Thus in the test time, the representations of unseen nodes can be generated with the trained GDPNet with graph structure and the associated node feature information. The task-specific rewards computed w.r.t the down-stream tasks are passed to the *signal neighbor selection* phase. These two phases are jointly trained to select optimal sets of neighbors for target nodes with maximum cumulative task-specific rewards, and to learn robust representations for nodes.

In summary, our contributions in this work include:

- We propose a novel model, GDPNet, for robust graph representation learning through reinforcement learning. GDPNet consists of two phrases, namely *signal neighbor selection* and *representation learning*, which enables

*Corresponding authors.

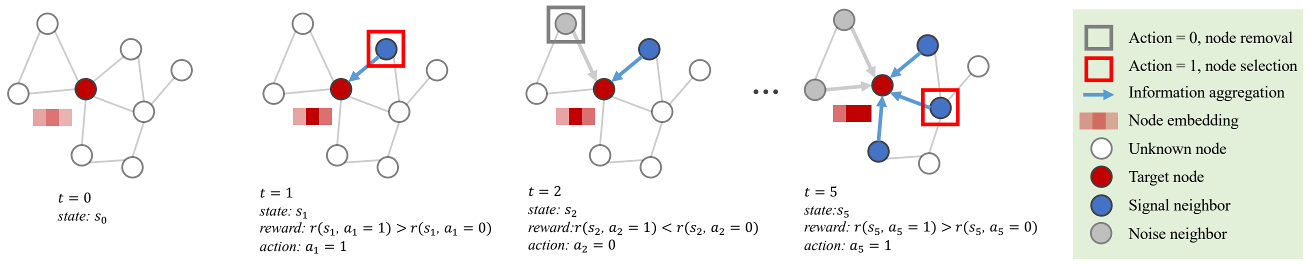


Fig. 1. Illustration of the GDPNet model from the view of signal neighbor selection

GDPNet to effectively learn node representations from noisy graph data.

- We formulate signal neighbor selection as a reinforcement learning problem, which enables the model to perform graph denoising just with weak supervision from the task-specific reward signals.
- GDPNet is able to generate representations for unseen nodes in an *inductive* fashion, which leverages both graph structure and the associated node feature information.

II. Approach

We formulate the robust graph representation learning problem as sequentially selecting an optimal set of neighbors for each node with maximum cumulative reward signals and aggregating features from nodes' optimal neighborhoods. In this part, we formally define the problem, the environment setting for signal neighbor selection, and the GDPNet model.

A. Problem Formulation

Given an attributed graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}, X\}$, where $\mathcal{E}$ is the edge set and $\mathcal{V}$ is the node set. $X \in \mathbb{R}^{|\mathcal{V}| \times D}$ collects the attribute information for each node where $x_v \in \mathbb{R}^D$ is a D-dimensional attribute vector of node $v \in \mathcal{V}$. Note that we can simply use one-hot encoding for node features for a graph without attributes. Given a target node v, let $\mathcal{N}(v) = \{u_1, u_2, ..., u_{|\mathcal{N}(v)|}\}$ be the one-hop neighbors of v.

We aim to find a lower-dimensional representation h_v for node $v \in \mathcal{V}$. Firstly, a function $f : 2^{\mathcal{N}(v)} \rightarrow 2^{\hat{\mathcal{N}}(v)}$ is learned to map a neighborhood set $\mathcal{N}(v)$ into a signal neighborhood set $\hat{\mathcal{N}}(v)$, where $\hat{\mathcal{N}}(v) \subseteq \mathcal{N}(v)$. Then the node representations are generated based on the signal neighborhood set, $h : 2^{\hat{\mathcal{N}}(v)} \xrightarrow{f} \mathbb{R}^d$. Given an order of the neighbors $u_1, ..., u_{|\mathcal{N}(v)|}$, we decompose the conditional probability of $\hat{\mathcal{N}}(v)$ given $\mathcal{N}(v)$ as $p(\hat{\mathcal{N}}(v)|\mathcal{N}(v)) = \Pi_{t=1}^{|\hat{\mathcal{N}}(v)|} p(a_t|\mathcal{N}(v), a_1, ..., a_{t-1})$ using chain rule [10], where $a_t = \{0, 1\}$, $a_t = 1$ indicates selecting u_t as a signal neighbor while $a_t = 0$ indicates removing u_t. We solve this signal neighbor selection problem by learning a policy $\pi_\theta(a_t|s_t) = p(a_t|\mathcal{N}(v), a_1, ..., a_{t-1})$ with neighborhood set $\mathcal{N}(v)$ and the predicted action values $\{a_i\}_{i=1}^{t-1}$ as inputs. The objective of signal neighbor selection is to select a subset of neighbors that maximize a given reward function $R_\pi(\hat{\mathcal{N}}(v)) = \mathbb{E}_{\hat{\mathcal{N}}(v)} \left[\sum_{t=1}^{|\hat{\mathcal{N}}(v)|} r_t \right]$, where $\hat{\mathcal{N}}(v)$ is the generated signal neighborhood set, r_t is the task-specific reward used to evaluate the action a_t, and R_π is the cumulative reward function. The representation of node v can then be learned by aggregating the neighborhood information from the signal neighbors $\hat{\mathcal{N}}(v)$.

Selecting an optimal subset from a candidate set by maximizing an objective function is NP-hard which can be approximatively solved by greedy algorithms with a submodular function [11]. With this observation, we design our reward function that satisfies submodularity, and show that the proposed GDPNet is mathematically equivalent to solving the submodular maximizing problem. Thus our solution can be bounded by $(1 - \frac{1}{e})R(\mathcal{N}(v)^*)$, where $\mathcal{N}(v)^*$ is the optimal neighborhood set.

B. Signal Neighbor Selection Environment

We formulate the problem of selecting a set of signal neighbors from a given neighborhood set as a Markov decision process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the state transition probability matrix that describes the transition probability of the state after taking an action, R is the reward function and γ is discount factor of the MDP. The signal neighbor selection process can be described by a trajectory with $\hat{\mathcal{N}}(v)$ time steps $s_0, a_0, r_0, ..., s_{|\hat{\mathcal{N}}(v)|}, a_{|\hat{\mathcal{N}}(v)|}, r_{|\hat{\mathcal{N}}(v)|}$. MDP requires the state transition dynamics to satisfy the Markov property $p(s_{t+1}|s_t)$. Thus we learn a policy $\pi_\theta(a_t|s_t)$ that only considers the current state s_t.

In reinforcement learning, the agent learns a policy via interacting with the environment. The main components (i.e., state, action, and reward) in the signal neighbor selection environment are described as follows,

- **State** ($\mathcal{S}$): The state $s_t = [h_v^t, h_{u_t}]$ encodes the information from the current node v and the selected node u_t, which is concatenation of the intermediate embeddings h_v^t and h_{u_t} of the target node v and the t^{th} neighbor u_t, respectively. The calculation of h_v^t and h_{u_t} are defined in Section II-C. Consequently, a newly selected neighbor u_t will update the embedding of v from h_v^t to h_v^{t+1} which can be viewed as state transition.
- **Action** ($\mathcal{A}$): Given an order of the neighbors $u_1, ..., u_{|\mathcal{N}(v)|}$ of node v, the policy $\pi_\theta(a_t|s_t)$ maps the state s_t into an action $a_t = \{0, 1\}$ at each time step t, $t = 1, ..., |\hat{\mathcal{N}}(v)|$. $a_1 = 1$ indicates u_1 is selected as a signal neighbor, while $a_1 = 0$ means u_1 is not selected.

- **Reward** (R): Our goal is to find an optimal set of signal neighbors $\hat{\mathcal{N}}(v)$ from a finite neighborhood set $\mathcal{N}(v)$ to learn robust graph embedding for downstream tasks such as node classification, link prediction and node clustering. The downstream tasks can produce task-specific scores as the reward signal for the signal neighbor selection phase. To ensure that the combination of the selected neighbors have maximum cumulative rewards. We employ the submodular function framework to define the marginal value reward function:

$$r_t = \frac{f_c(\textsc{Agg}(x_v, \{x_{u_t}\}))}{\sum_{\tilde{u} \in \hat{\mathcal{N}}(v)_t} f_c(\textsc{Agg}(x_v, \{x_{\tilde{u}}\}))} \tag{1}$$

where $\textsc{Agg}(\cdot)$ aggregates both the target node feature x_v and the neighbors' features $\{x_{u_t}\}$ to update the representations of the target node [6], and $f_c(\cdot)$ returns the micro-averaged F1 score from the node classification task when considers u_t as the neighbor.

The environment updates the states from $s_t = [h_v^t, h_{u_t}]$ to $s_{t+1} = [h_v^{t+1}, h_{u_{t+1}}]$ by calculating the representations $h_v^{t+1} = \textsc{Agg}(x_v, \{x_{\tilde{u}}, \forall \tilde{u} \in \hat{\mathcal{N}}(v)_t\})$ at time $t+1$. It can be considered as a state transition:

$$p(s_{t+1}|s_t) = \sum_{a_t} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{2}$$

If $a_t = 1$, $\hat{\mathcal{N}}(v)_t = \hat{\mathcal{N}}(v)_{t-1} \cup \{u_t\}$, otherwise $\hat{\mathcal{N}}(v)_t = \hat{\mathcal{N}}(v)_{t-1}$.

C. *Graph Denoising Policy Network*

With the definitions of the signal neighbor selection environment, we introduce the GDPNet model which includes two phases: signal neighbor selection and representation learning. Given a target node v, GDPNet first takes its neighborhood set $\mathcal{N}(v)$ as input and outputs a signal neighborhood subset $\hat{\mathcal{N}}(v)$. Then the representations h_v is learned by aggregating the information from the signal neighborhood subset $\hat{\mathcal{N}}(v)$.

1) Determine the Neighborhood Order: As aforementioned, we use chain rule to decompose the signal neighbor selection as a sequential decision making process. However, it requires an order to make decisions. Here we design a high-level policy to learn an order $[u_1, ..., u_{|\mathcal{N}(v)|}]$ for the policy π_θ to take action.

We define a *regret score* l for each neighbor to help determine the order. A neighbor with large regret score indicates it will be selected with higher probability. At each time step, we calculate the regret score of each neighbor and sample one of the neighbor to be the *th* neighbor. The regret score is described as follows:

$$l_k = W_1 \cdot \text{ReLU}(W_2 \cdot s_t), s_t = [h_v^t, h_{u_k}] \tag{3}$$

where u_k is the k-th neighborhood in the neighborhood set $\mathcal{N}(v)$ with a random order and W_1, W_2 are parameter matrices. To reduce the size of $\hat{\mathcal{N}}(v)$ for computational efficiency, we add an ending neighbor u_e to $\mathcal{N}(v)$ for early stopping purpose. When u_e is sampled, the neighborhood selection process of node v stops. We use the *Softmax* function to normalize the regret scores, and sample one neighbor from the distribution generated by *Softmax* to be the t^{th} neighbor.

$$u_t \sim \textsc{Softmax}([l_1, l_2, ..., l_e, ..., l_{|\mathcal{N}(v)_t^\complement|}]) \tag{4}$$

where $u_t \in \mathcal{N}(v)_t^\complement$ is the t^{th} neighbor for signal neighbor selection, $\mathcal{N}(v)_t^\complement = (\mathcal{N}(v) \setminus \hat{\mathcal{N}}(v)_t)$. l_e indicates the regret score of the ending neighbor u_e. After selecting a neighbor u_t, we adopt the policy π_θ to determine whether to select u_t as a signal neighbor. Then u_t will be removed from $\hat{\mathcal{N}}(v)_t^\complement$.

2) Signal Neighbor Selection: Given the t^{th} neighbor u_t, GDPNet takes an action $a_t = \{0, 1\}$ at time step t to decide whether to select the u_t. We will make $|\hat{\mathcal{N}}(v)|$ decisions to select the signal neighbors for node v. Here the total number of signal neighbors can be automatically determined. As illustrated in Fig. 1, a policy $\pi_\theta(a_t|s_t)$ is learned to map the state s_t to the action a_t at time step $t, t = 1, ..., |\hat{\mathcal{N}}(v)|$, meanwhile the corresponding reward r_t will be provided. Our goal is to maximize the total reward of all the actions taken during these time steps, which can be learned by the following policy network,

$$\begin{aligned} \pi_\theta(a_t|s_t) &= \sigma\left(W_1 \cdot \text{ReLU}(W_2 \cdot s_t)\right) \\ a_t &\sim \pi_\theta \in \{0, 1\} \end{aligned} \tag{5}$$

where W_1 and W_2 are weight matrices shared with Eq. (3), and action a_t is sampled from a Bernoulli distribution which is generated by $\pi_\theta(a_t|s_t)$.

3) Representation Learning: At each time step, GDPNet calculates the embeddings of the target node v and the t-th neighbor u_t as follows,

$$h_v^t \leftarrow \textsc{Agg}(x_v, \{x_{\tilde{u}}, \forall \tilde{u} \in \hat{\mathcal{N}}(v)_t\}) \tag{6}$$

$$h_{u_t} \leftarrow \textsc{Agg}(x_{u_t}, \{\varnothing\}) \tag{7}$$

where $\textsc{Agg}(x, \{y_i, \forall i \in \mathcal{I}\}) = \sigma(W \cdot \texttt{MEAN}(\{x\} \cup \{y_i, \forall i \in \mathcal{I}\})$, x_v and x_{u_t} are the features of node v and u_t respectively. We computed the embedding of neighbor u_t via its own feature x_{u_t}, because the goal is to evaluate the individual contribution of u_t. In this work we only consider one-hop neighbors for simplicity. The GDPNet model can be easily extended to aggregate the information from multi-hop neighbors with an augmented candidate neighborhood set for selecting the signal neighbors.

As defined in Section II-B, the state at time step t, $s_t = [h_v^t, h_{u_t}]$, is a concatenation of the intermediate node embeddings h_v^t and h_{u_t}. Eventually, the representations h_v and state $s_t = [h_v^t, h_{u_t}]$ can be obtained.

4) Iteration-wise Optimization: We consider an iteration-wise optimization approach to optimize the GDPNet model, which optimizes the signal neighbor selection phrase and representation learning phrase iteratively to learn the policy π_θ and the representations h_v. As for representation learning phase, it aggregates the information from the signal neighbors selected by π_θ to learn an embedding h_v for target node v. Meanwhile, the policy π_θ is trained with the states calculated by h_v and the corresponding rewards. In this paper, π_θ is optimized with Proximal Policy Optimization (PPO), one of the widely used policy gradient method [12].

$$\max \quad \mathbb{E}_{s \sim \rho_{\theta_{old}}, a \sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)} Q_{\theta_{old}}(s,a)\right], \tag{8}$$

$$s.t. \;\; \mathbb{E}_{s \sim \rho_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}}(\cdot|s) \parallel \pi_\theta(\cdot|s))\right] \leq \delta$$

where KullbackLeibler (KL) divergence penalty is used to control the change of the policy at each iteration to perform a trust region update with a threshold δ. $q(a|s)$ and $Q_{old} = \sum_{i=t}^{T} \gamma r_i$ are the policy and Q-value, respectively, which are saved before the current time step during training. $\rho_{\theta_{old}}$ is the discounted state distribution defined as,

$$\rho_{\theta_{old}}(s_t) = \sum_{t=0}^{T} \gamma^{t-1} p(s_t = s|\pi_{\theta_{old}}) \tag{9}$$

III. Experiment

Experiments are conducted to evaluate the robustness of the representations learned by the proposed GDPNet model. As for quantitative experiments, we focus on two tasks: (1) *Robustness Evaluation*, we use micro-averaged F1 score to evaluate our model against baselines on node classification task, and (2) *Denoising Evaluation*, we evaluate the denoising capability of GDPNet by comparing with baselines running on the denoised graph generated by GDPNet. We extract four datasets Cora, Citeseer, PubMed and DBLP followed by splitting them for training, test and validation with the supervised learning scenario which follows the previous work [6], [8], [9]. As for qualitative experiments, we conduct the embedding visualization which projects the learned high-dimension representations to a 2D space. In all these experiments, we separate out test data from training and perform predictions on nodes that are not seen during training.

A. Experimental Setup and Baselines

For all these tasks, we apply a two-layer policy network to select the signal neighbors. The embedding dimension is 128. The size of the two hidden layers in policy network are 64 and 36, respectively, with active function ReLU. The batch size is 256. The discount factor is optimized as 0.95 for Cora and DBLP, 0.9 for PubMed and 1.0 for Citeseer. We compare our method with the following baselines: (1) Logistic regression (LR) model which takes the node features as inputs, and ignores graph structure; (2) GCN [7] which uses the local connection structure of the graph as the filter to perform convolution. We use inductive version of GCN in this paper for comparison; (3) GAT [9] which utilizes the attention mechanism to enhance the performance of GCN; (4) FastGCN [8]which samples the neighborhoods in each layer independently to addresses the recursive expansion of neighborhoods, and (5) GraphSAGE [6].

Our proposed model is denoted as **GDPNet**. We also introduce a variant **GDPNet*RO*** which performs the signal neighbor selection with a random order of the neighbors.

B. Performance Comparison

In this section, we first visualize the node representations learned by different methods, followed by the performance comparison on node classification task. Additionally, we show the distributions of the selected signal neighbors with GDPNet on different dataset.

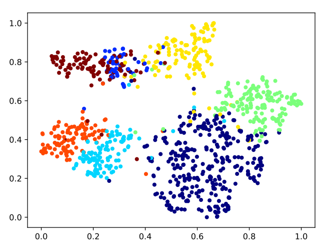
1) Embedding Visualization: Node representations are learned by GAT, GCN, GraphSAGE and GDPNet on test dataset of Cora, and visualized with t-SNE [13], as shown in Fig. 2. Different colors in the figure represent different categories in Cora. The following observations can be made from Fig. 2,

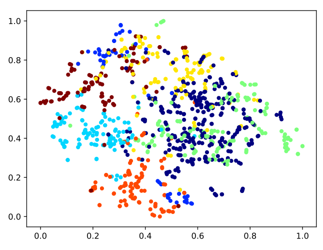
- GDPNet correctly detects the classes in Cora, providing empirical evidence for the effectiveness of our method. This can be seen by the clear gap between samples with different colors. It also demonstrates that, removing the noisy neighbors can help nodes learn better representations.
- GAT cannot effectively identify different classes as other methods, it might because it considers all the neighbors with attention weights, which is easily to introduce noisy neighbors.

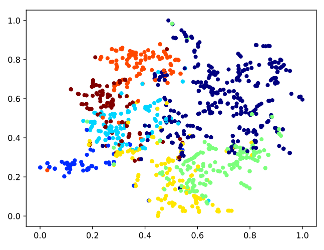
2) Results on Node Classification: In this part, we compare the performance of GDPNet against the baselines on Cora, Citeseer, PubMed and DBLP. For all methods, we run the experiments with random seeds over 15 trials and record the mean and standard variance of the *micro-average F1* scores. The results are summarized in Table I. From the table we observe that,

- GDPNet consistently outperforms the other methods, which demonstrates there exists a set of noisy neighbors in each dataset on node classification task, and GDPNet can learn robust embeddings by effectively removing these noisy neighbors.
- GCN, FastGCN and GraphSAGE show lower F1 scores. The reason is that these methods randomly sample a subset of neighbors for representation learning, which is hard to avoid the noisy neighbors. In addition, variance is higher via random sampling.
- GAT learns the importance of the neighbors with attention weights, which is also sensitive to noisy data according to the reported results.
- Another interesting observation is that Logistic regression achieves better performance than the other baselines on PubMed, which indicates that there would be less signal neighbors for the nodes in PubMed. This observation can also be verified in Fig.3.
- GDPNet*RO* has a lower F1 score with higher variance than GDPNet, which demonstrates that the order of the decisions has an effect on the performance of representation learning. Thus learning an order for the neighbors is beneficial for selecting signal neighbors and robust graph representation learning.

3) Distribution of the Selected Neighbors: Fig. 3 shows the distribution of the selected neighbor percentages, where


(a) Cora-GDPNet


(b) Cora-GAT


(c) Cora-GCN

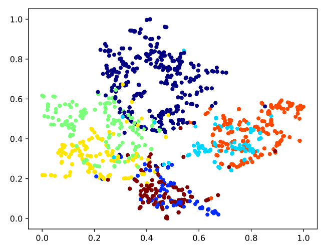

(d) Cora-GraphSAGE

Fig. 2. Visualizations of the compared methods on Cora.

TABLE I
SUMMARY OF NODE CLASSIFICATION RESULTS IN TERMS OF MICRO-AVERAGED F1 SCORE, FOR CORA, CITESEER, PUBMED AND DBLP

Method	LR	GAT	GCN	FastGCN	GraphSAGE	GDPNet$_{RO}$	GDPNet
Cora	$0.799 \pm 1.06\%$	$0.819 \pm 0.45\%$	$0.838 \pm 0.50\%$	$0.865 \pm 4.50\%$	$0.867 \pm 1.05\%$	$0.879 \pm 2.14\%$	$\mathbf{0.881 \pm 0.31\%}$
PubMed	$0.871 \pm 0.82\%$	$0.778 \pm 0.71\%$	$0.826 \pm 0.22\%$	$0.867 \pm 1.05\%$	$0.854 \pm 0.87\%$	$0.880 \pm 2.51\%$	$\mathbf{0.893 \pm 0.57\%}$
DBLP	$0.784 \pm 1.03\%$	$0.736 \pm 0.82\%$	$0.805 \pm 2.17\%$	$0.774 \pm 0.41\%$	$0.803 \pm 1.28\%$	$0.832 \pm 0.97\%$	$\mathbf{0.836 \pm 0.57\%}$
Citeseer	$0.813 \pm 0.58\%$	$0.719 \pm 0.50\%$	$0.829 \pm 1.56\%$	$0.779 \pm 0.53\%$	$0.910 \pm 0.73\%$	$0.952 \pm 1.15\%$	$\mathbf{0.957 \pm 0.33\%}$

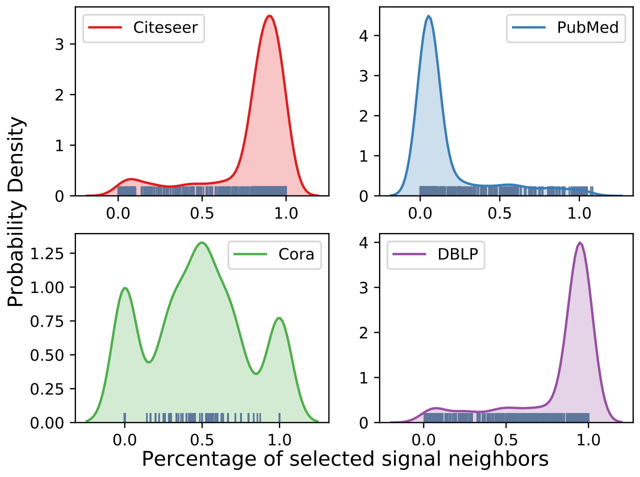


Fig. 3. The distribution of the selected signal neighbor percentages.

the x-axis indicates the percentage of the nodes been selected as signal neighbors, and the y-axis indicates the probability densities. We observe that most of the neighbors in Citeseer and DBLP are selected while only a few neighbors are selected in PubMed. The results show that there would be more "noisy" citations (e.g. cross-field citation) in PubMed than in Citeseer and DBLP. Interestingly, most of the research papers collected in Citeseer and DBLP are from computer science, while PubMed collects papers from biomedical.

4) Parameter Sensitivity Study: In Fig. 4, we vary the training percentage of nodes in Citeseer and PubMed to test the classification accuracy. We observe that, the performance of all the methods are improved with the increases of the training percentage. Additionally, it can be seen that GAT is very sensitive to the percentages of training data, and it requires larger proportion of training data in order to have a desirable performance. GraphSAGE, GCN and GDPNet achieve good performances on small training data, and GDPNet make more improvements as the training data percentage increases.

5) Convergence Analysis: Fig. 5 shows the convergence analysis of GDPNet on Citeseer and PubMed. We initialize the policy randomly when epoch equals 0, and the neighbors are randomly selected as signal neighbors. We observe that Citeseer converges faster than PubMed. One explanation would be that PubMed has more nodes than Citeseer, which requires more time to explore the policy for nodes.

IV. RELATED WORK

A. Graph Representation Learning

Graph representation learning tries to encode the graph structure information into vector representations. The main idea is to learn a mapping function from the nodes or entire graphs into an embedding space where the geometric relationships in the low-dimensional space coincide with the original graph. The methods can be grouped into two categories: matrix factorization based methods and graph neural network based methods [1].

1) Graph Neural Network based Embedding: A set of graph neural network based embedding methods are proposed recently for representation learning [14]–[17]. GCN [7] first proposes the first-order graph convolution layer to perform recursive neighborhood aggregation based on the local connection. Instead of utilizing full graph Laplacian during training in the GCN, GraphSAGE [6] considers the inductive setting to handle the large scale graph with batch training and neighborhood sampling. Followed by GraphSAGE, self-attention mechanism has been explored to enhance the representation learning performance [9], [18]. To accelerate the training of GCNs, [8] samples the nodes in each layer independently, while [19] samples the lower layer conditioned on the top one and the sampled neighborhoods are shared by different parent node. In this work, we propose to find an effective subset of neighbors for learning robust representations.

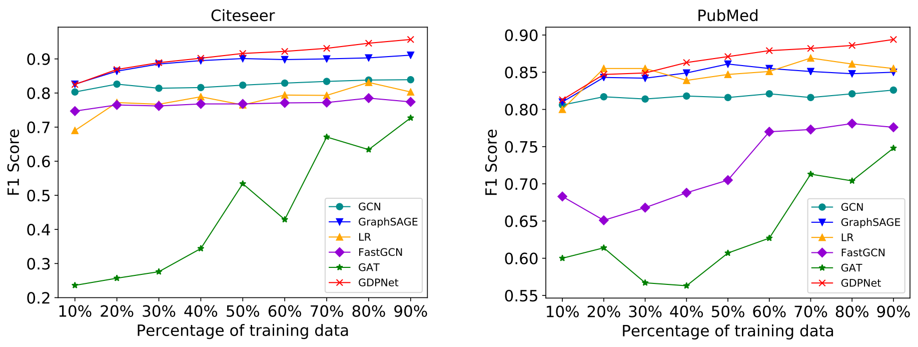


Fig. 4. Performance on different percentage of training data

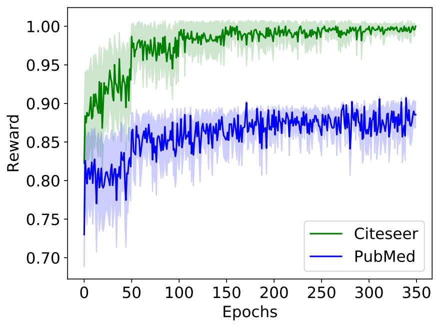


Fig. 5. Convergence analysis

B. *Reinforcement Learning on Graph*

Reinforcement learning solves the sequential decision making problem with the goal of maximizing cumulative rewards of these decisions. A set of work used reinforcement learning to solve the sequential decision making problems in graph, such as minimum vertex cover, maximum cut and travelling salesman problem [20], [21]. You et al. [22] considered the molecular graph generation process as a sequential decision making process where the reward function is designed by non-differentiable rules. Dai et al. [23] utilized reinforcement learning to learn an attack policy to make multiple decisions (delete or add edges in the graph) to attack the graph.

V. Conclusion

In this paper, we developed a novel framework, GDPNet, to learn robust representations from noisy graph data through reinforcement learning. GDPNet includes two phases: *signal neighbor selection* and *representation learning*. It learns a policy to sequentially select the *signal* neighbors for each node, and then aggregates the information from the selected neighbors to learn node representations for the down-stream tasks. These two learning phases are complementary and achieves significant improvement. Experiments on a set of well-studied datasets provide empirical evidence for our analytical results, and yield significant gains in performance over state-of-the-art baselines.

Acknowledgements

This work was partially supported by NSF DGE-1829071 and the National Key Research and Development Program of China under Grant No. 2016YFB1000904, NSFC (61702190, U1609220) and Shanghai Chenguang Program (16CG24). We thank the anonymous reviewers for their careful reading and insightful comments on our manuscript.

References

[1] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” *arXiv preprint arXiv:1709.05584*, 2017.

[2] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[3] P. Cui, X. Wang, J. Pei, and W. Zhu, “A survey on network embedding,” *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[5] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018.

[6] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *NeurIPS*, pp. 1024–1034, 2017.

[7] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[8] J. Chen, T. Ma, and C. Xiao, “Fastgcn: fast learning with graph convolutional networks via importance sampling,” *arXiv preprint arXiv:1801.10247*, 2018.

[9] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[10] W. Liu and I. Tsang, “On the optimality of classifier chain for multi-label classification,” in *NeurIPS*, pp. 712–720, 2015.

[11] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, “An analysis of approximations for maximizing submodular set functionsi,” *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[13] L. v. d. Maaten and G. Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[14] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in *ICLR*, 2015.

[15] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints,” in *NeurIPS*, pp. 2224–2232, 2015.

[16] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, “Gated graph sequence neural networks,” *arXiv preprint arXiv:1511.05493*, 2015.

[17] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in *ICML*, pp. 2014–2023, 2016.

[18] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, “Gaan: Gated attention networks for learning on large and spatiotemporal graphs,” *arXiv preprint arXiv:1803.07294*, 2018.

[19] W. Huang, T. Zhang, Y. Rong, and J. Huang, “Adaptive sampling towards fast graph representation learning,” in *NeurIPS*, pp. 4558–4567, 2018.

[20] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *NeurIPS*, pp. 6348–6358, 2017.

[21] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” in *ICLR*, 2017.

[22] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, “Graph convolutional policy network for goal-directed molecular graph generation,” in *NeurIPS*, pp. 6410–6421, 2018.

[23] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, “Adversarial attack on graph structured data,” in *ICML*, 2018.