Batch Active Learning with Graph Neural Networks via Multi-Agent Deep Reinforcement Learning

Yuheng Zhang[1], **Hanghang Tong**[1], **Yinglong Xia**[2], **Yan Zhu**[2], **Yuejie Chi**[3], **Lei Ying**[4]

[1] University of Illinois at Urbana-Champaign
[2] Facebook AI
[3] CMU
[4] University of Michigan, Ann Arbor
yuhengz2@illinois.edu, htong@illinois.edu, yxia@fb.com, yzhu@fb.com, yuejiechi@cmu.edu, leiying@umich.edu

Abstract

Graph neural networks (GNNs) have achieved tremendous success in many graph learning tasks such as node classification, graph classification and link prediction. For the classification task, GNNs' performance often highly depends on the number of labeled nodes and thus could be significantly hampered due to the expensive annotation cost. The sparse literature on active learning for GNNs has primarily focused on selecting only one sample each iteration, which becomes inefficient for large scale datasets. In this paper, we study the batch active learning setting for GNNs where the learning agent can acquire labels of multiple samples at each time. We formulate batch active learning as a cooperative multi-agent reinforcement learning problem and present a novel reinforced batch-mode active learning framework (BiGeNe). To avoid the combinatorial explosion of the joint action space, we introduce a value decomposition method that factorizes the total Q-value into the average of individual Q-values. Moreover, we propose a novel multi-agent Q-network consisting of a graph convolutional network (GCN) component and a gated recurrent unit (GRU) component. The GCN component takes both the informativeness and inter-dependences between nodes into account and the GRU component enables the agent to consider interactions between selected nodes in the same batch. Experimental results on multiple public datasets demonstrate the effectiveness and efficiency of our proposed method.

Introduction

Graph data are pervasive in many real-world scenarios, e.g., social networks, citation networks, protein-protein interaction networks and many more. Recently, graph neural networks (GNNs) have attracted tremendous attention because of significant success in downstream tasks such as node classification (Kipf and Welling 2016), link prediction (Zhang and Chen 2018), and graph classification (Morris et al. 2019). However, many of these successes have been limited due to expensive annotation costs. Active learning is a promising direction to tackle this challenge. With effective active learning algorithms for graphs, the learning agent could identify samples which are most informative for improving the training. Thus, a high accuracy GNN could be trained with minimal labeling effort.

Although there exist many effective active learning algorithms for computer vision and natural language processing (Aggarwal et al. 2014) (Settles 2009), these methods are primarily designed for independent and identically distributed (i.i.d.) data which is quite different from graph-structured data containing inter-dependent connections between different samples (i.e., nodes). How to design an effective and efficient active learning method for graphs largely remains a challenge.

Recently, several methods have been proposed (Cai, Zheng, and Chang 2017) (Gao et al. 2018) (Hu et al. 2020) to incorporate the graph characteristics into active learning. They design several selection metrics such as information entropy, node centrality, and information density to measure the informativeness of candidate unlabeled nodes. To balance the weights of different metrics, (Cai, Zheng, and Chang 2017) proposes to use a time-sensitive parameter drawn from a beta distribution. (Gao et al. 2018) utilizes a multi-armed bandit mechanism to learn an adaptive weight vector with performance improvement as the reward. To further exploit the topology information on graphs, (Hu et al. 2020) proposes a transferable reinforcement learning framework and parameterizes the policy network as a graph convolutional network (GCN) (Kipf and Welling 2016). At each iteration, they select the sample with the highest action probability output from the policy network. Although (Hu et al. 2020) has achieved remarkable results with small query budgets, it becomes inefficient when applied to a large scale dataset where selecting samples one by one takes prolonged time. Hence, in order to improve the efficiency of the active learning method, a more promising setting is *batch active learning* where the agent acquires labels of multiple samples at each time.

Different from one-by-one active learning, batch active learning poses several unique challenges as follows. First, in the batch setting, the agent not only needs to select the most informative samples but also needs to avoid selecting redundant samples. Effective acquisition strategies should simultaneously consider informativeness and diversity. Second, when applying reinforcement learning to batch active learning, each action represents the selection of multiple nodes instead of a single node, which causes a combinatorial explosion of the action space. One has to carefully design a batch-mode algorithm in order to train the policy network efficiently. To overcome the above challenges, we propose a novel multi-agent reinforced active learning framework. We formulate batch active learning as a cooperative multi-agent reinforcement learning problem. Suppose we want to select n samples in each iteration, then there are n agents in our framework. At each time step, each agent chooses an action (i.e., an unlabeled sample), forming a joint action (i.e., a selected batch). Every agent shares the same joint reward function and their goal is to collaborate with each other to maximize the improvement of the classification model.

Inspired by the prior arts on multi-agent reinforcement learning (MARL) (Sunehag et al. 2017) (Son et al. 2019), we employ the paradigm of centralized training with decentralized execution (CTDE) (Oliehoek, Spaan, and Vlassis 2008). The key idea lies in the Individual-Global-Max (IGM) principle (Son et al. 2019): the optimal joint action of the team is equivalent to the union of the optimal individual actions of each agent. In CTDE, all the agents share the same global reward and learn decentralized policies that allow each agent to take optimal actions simply with individual observations.

Due to the exponentially growing joint action space, it is intractable to directly model the joint action-value function. Therefore, we introduce a value-decomposition approach factorizing the joint Q-function Q_{tot} into the average value of individual Q-functions Q_i. We show that this approach provides a good approximation to Q_{tot} and works very well in practice.

Furthermore, we propose a novel multi-agent Q-network to learn the batch active learning policy. Our proposed Q-network consists of two parts, including a graph convolutional network (GCN) component Q^{GCN} and a gated recurrent unit (GRU) component Q^{GRU}. The GCN component takes both the informativeness and the inter-dependent connections between candidate nodes into account and selects the first node of the batch. The GRU component is employed to select the remaining $n-1$ nodes. The hidden state of the GRU component represents the aggregated information of the previously selected nodes which enables it to consider the interactions between different agents. Our Q-network admits a variety of state representation including both informativeness and diversity.

The main contributions of the paper are summarized as follows:

1. We propose a novel multi-agent reinforced active learning framework for GNNs (BIGENE), which is based on Q-learning and value decomposition. To the best of our knowledge, we are the first to introduce multi-agent reinforcement learning to batch active learning.
2. We propose a novel multi-agent Q-network consisting of a GCN component and a GRU component. The GCN component considers both the informativeness and the inter-dependences between candidate nodes. The GRU component aggregates information from the selected nodes in the batch and thus considers the interactions between different agents.
3. We implement BIGENE on the semi-supervised node classification task and conduct extensive experiments to demonstrate its effectiveness and efficiency.

Related Work

Active Learning with Reinforcement Learning. To maximize the performance gain in the training process, there are several active learning studies that employ deep reinforcement learning methods to learn a labeling policy. (Fang, Li, and Cohn 2017) uses deep Q-learning to tackle the stream-based active learning problem where unlabeled data arrive one by one. (Konyushkova, Sznitman, and Fua 2018) (Casanova et al. 2020) propose a reinforced active learning framework for pool-based active learning, which means all the unlabeled data are provided at the beginning. Similar to ours, (Casanova et al. 2020) also deals with the batch mode active learning setting. They randomly divide the candidate samples into several groups and learn a policy to select one sample from each group. The reward of each selected node is approximated with the reward of the entire batch. This strategy decomposes the batch selection into several independent single sample selections which overlooks the relationship between selected samples. In contrast, we consider batch active learning from the perspective of multi-agent learning and model the collaboration between different agents. Unlike these methods which are proposed for i.i.d. data, our paper focuses on graph data with the goal of reducing the labeling cost of training GNNs.

Active Learning on Graphs. Recently, there have been several studies focusing on active learning with GNNs. AGE (Cai, Zheng, and Chang 2017) proposes an active query strategy with a combination of several selection criteria including information entropy, embedding representativeness and graph centrality. To balance different criteria, they introduce time-sensitive parameters drawn from a beta distribution. Similarly, ANRMAB (Gao et al. 2018) also uses these criteria and employs a multi-armed bandit mechanism to learn the weights of the linear combination adaptively. To deal with active learning on heterogeneous graphs, ActiveHNE (Chen et al. 2019) introduces several query strategies and also assembles them with the multi-armed bandit mechanism. All these methods consider each node independently with a combination of different selection criteria but overlook the connections between nodes in the graph-structured data. GPA (Hu et al. 2020) proposes a reinforcement learning framework to tackle the active learning on graphs and parameterizes the policy network as another GNN which could model node interactions. However, their method is designed for one by one selection setting, which is inefficient compared with the batch mode setting. Moreover, they measure the informativeness of samples but neglect the diversity of selected samples which plays an important role in active learning. Our proposed method is capable of considering both informativeness and diversity and achieves promising results in the batch mode active learning setting.

Multi Agent Reinforcement Learning (MARL). Value-based MARL algorithms have recently achieved state-of-the-art performance on challenging tasks, e.g., unit micromanagement tasks built in StarCraft II (Samvelyan et al.

2019). (Sunehag et al. 2017) develops value decomposition networks (VDN) decomposing the joint action-value function into the sum of individual action-value functions. Instead of assuming additivity, QMIX (Rashid et al. 2018) employs a neural network to model the relationship between joint Q-function and individual Q-functions. QPLEX (Wang et al. 2020) proposes to use the dueling network architecture (Wang et al. 2016) and reformalizes the IGM principle as an advantage-based IGM. Despite the success achieved on micromanagement tasks, none of the previous work has studied how to apply MARL to active learning problems.

Preliminary

Given a graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ along with its adjacency matrix $A \in \mathbb{R}^{N \times N}$, node feature matrix $X \in \mathbb{R}^{N \times F}$. $\mathcal{V}$ is the node set, and $\mathcal{E}$ is the edge set, N denotes the number of nodes and F denotes the dimensionality of the node input feature. In this paper, we focus on the semi-supervised node classification task on $\mathcal{G}$ and train a graph convolutional network (GCN) (Kipf and Welling 2016) as the classification model. The propagation rule of GCN is written as:

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

where $\tilde{A}$ is the adjacency matrix with self-connections $\tilde{A} = A + I$, $\tilde{D}$ is the degree matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and σ denotes the ReLU function (Glorot, Bordes, and Bengio 2011). $W^{(l)}$ denotes the layer-specific trainable weight matrix and $H^{(l)}$ denotes the matrix of activations in the l^{th} layer. $H^{(0)} = X$. We split the node set V into three disjoint subsets V_{train}, $V_{\text{validation}}$, V_{test} and employ cross entropy loss to optimize the GCN model f with labeled nodes from V_{train}. The performance of model f is validated on $V_{\text{validation}}$ and tested on V_{test}.

For batch active learning with GNNs, we first initialize the labeled dataset L_0 by querying the oracle for the labels of M samples drawn uniformly from the unlabeled pool $U = V_{\text{train}}$. At each iteration t, we are allowed to select a batch of n samples $u_t^1, u_t^2, \ldots, u_t^n$ from unlabeled pool $U = V_{\text{train}} \setminus L_{t-1}$, query the oracle for their labels, and expand our labeled dataset $L_t = L_{t-1} \cup \{u_t^1, u_t^2, \ldots, u_t^n\}$. This process is continued until the query budget B is used up and our goal is to maximize the performance of classification model f on V_{test} in each iteration. Notice, the one by one selection setting studied in (Hu et al. 2020) is a special case of our batch mode setting when we set the size of query batch n to be 1.

We summarize four desired properties for effective batch active learning strategies to train GNNs: (1) Informativeness, the amount of information a single node contains for training GNNs. It includes both uncertainty and centrality. (2) Diversity measures the redundancy of selected nodes. Training sets with high diversity are often more representative of the underlying data distribution. (3) Inter-dependence captures the graph structural information which makes it different from i.i.d. data. (4) Batch interaction means considering the interactions between selected nodes in the same batch. From Table 1, we can see that our proposed BIGENE is the only method possessing all the desired properties.

Property	AGE	ANRMAB	GPA	Ours
Informativeness	✓	✓	✓	✓
Diversity	✓	✓		✓
Inter-dependence			✓	✓
Batch Interaction				✓

Table 1: The properties of methods for active learning on graphs.

Methodology

Batch Active Learning as Cooperative Multi-Agent Reinforcement Learning

Effective batch active learning query strategies should be capable of selecting not only informative but also diverse samples. It requires us to consider the relationship between samples in the same batch. To achieve this goal, we formulate batch active learning as a cooperative multi-agent reinforcement learning problem. Here, we describe the multi-agent task as a Markov Decision Process (MDP) consisting of a tuple $G = \langle S, U, P, r, n, \gamma \rangle$. State $s \in S$ is defined as the combination of the original graph-structured data and the current condition of the trained classification model. n denotes the number of agents which is equivalent to the batch size of active learning. At each time step, each agent $i \in \mathcal{N} := \{1, \ldots, n\}$ chooses an unlabeled node $u_i \in \mathcal{U}$, forming a joint action (i.e., the selected batch) $\mathbf{u} \in \mathbf{U} \equiv U^n$. P describes the transition function $P(s' \mid s, \mathbf{u}) : S \times \mathbf{U} \times S \mapsto [0, 1]$. Specifically, it depicts the change of the status of the classification model after trained on an expanded dataset. All agents share the same reward r defined as the performance gain of the classification model with expanded training set and γ is the discount factor. In our setting, all agents share the same observation and do not have the individual action-observation history which is different from the general MARL (Oliehoek and Amato 2016).

State

Node Centrality. Different from active learning methods commonly used in other domains such as computer vision and natural language processing, active learning on graphs needs to handle non i.i.d. samples with inter-dependent connections between them. The intuition behind using node centrality is that with a higher centrality score, the node is more likely to be a hub in the graph. Therefore, when we obtain the ground-truth label of the node, it is more likely to provide more information about other surrounding nodes. Various node centrality metrics have been proposed such as closeness centrality (Stephenson and Zelen 1989), betweenness centrality (Brandes 2001) and PageRank centrality (Page et al. 1999; Tong, Faloutsos, and Pan 2006). In our experiments, we observe that PageRank centrality outperforms others, which is also consistent with (Cai, Zheng, and Chang 2017). Hence, we adopt PageRank centrality to measure the informativeness of node v_i in the graph as follows:

$$\phi_i^c = d \sum_j A_{ij} \frac{\phi_j^c}{\sum_k A_{jk}} + \frac{1-d}{N}, \quad (2)$$

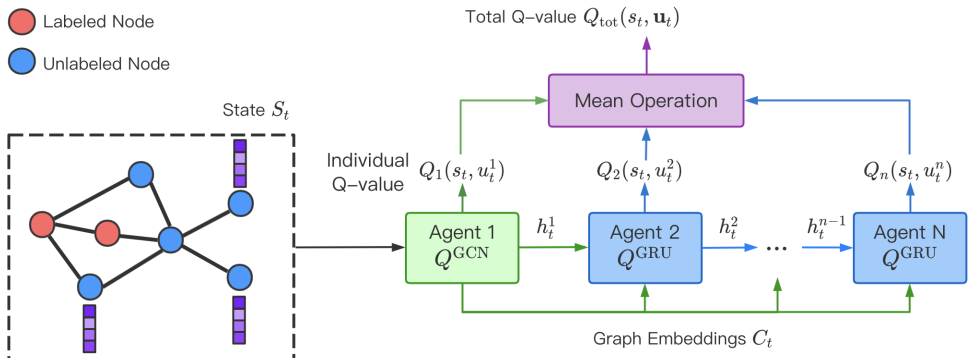

Figure 1: Illustration of the proposed multi-agent Q-network. At each iteration, the state s_t is fed into Q^{GCN} to obtain the Q-value of the first agent and the graph embeddings of candidate nodes. Then, we use Q^{GRU} as the Q-network for the remaining $n-1$ agents. The input of Q^{GRU} is the graph embeddings from Q^{GCN} and the hidden state is the embedding of the most recent selected node. The final total Q-value equals to the average of individual Q-values.

where $0 < d < 1$ is the damping parameter.

Uncertainty. Uncertainty-based selection strategy is widely used in active learning studies. The classification model is more likely to make mistakes on samples with less confidence. Here, we use the entropy of the prediction as the uncertainty metric:

$$\phi_i^e = -\sum_{c=1}^{C} \hat{y}_{ic} \log \hat{y}_{ic}, \tag{3}$$

where C is the number of classes and $\hat{y}_{ic}$ is the probability of node v_i belonging to class c predicted by the current classification model f.

Diversity. Inspired by Core-set method (Sener and Savarese 2017) which performs well in image classification tasks, we propose a novel metric to measure the diversity of the candidate nodes with the embeddings from the classification model. The motivation is as follows. If we only select the most informative samples, the distribution of the selected training set could be quite different from the underlying distribution of the entire dataset, which introduces a distribution shift. By calculating the distance between the embeddings of unlabeled nodes and the embeddings of labeled nodes, we could select faraway nodes which makes the training set more representative of the entire dataset. Thus, the node embedding distance is calculated as:

$$\phi_i^d = \min_{j \in L} d(x_i, x_j), \tag{4}$$

where x_i denotes the embedding of node v_i from the classification model, $d(\cdot,\cdot)$ denotes the Euclidean distance measurement and L denotes the current labeled training set.
We concatenate these above metrics as the state representation of each node. The overall state S_t is a matrix of N rows and each row s_t^i represents the state of node v_i.

Action

Since the size of the joint action space is $O(N^n)$ where n denotes the number of nodes in the batch, it is intractable to directly model the joint action-value function. Here, we follow the Individual-Global-Max (IGM) principle (Son et al. 2019): the optimal joint action is equivalent to the union of the optimal individual actions of each agent. For each iteration t, n agents sequentially take the optimal action based on the current observation, i.e., selecting the optimal unlabeled node. The joint action $\mathbf{u}_t$ is the union of individual actions $\{u_t^1, \cdots, u_t^n\}$.

Reward

Considering that our goal is to improve the classification performance of model f, our reward is naturally designed as the performance gain on the validation set, given by

$$r_t = \mathcal{M}(f_{L_t}, V_{\text{validation}}) - \mathcal{M}(f_{L_{t-1}}, V_{\text{validation}}), \tag{5}$$

where f_{L_t} is the classification model trained with labeled subset L_t and $\mathcal{M}$ is the evaluation metric such as the classification accuracy.

Proposed Multi-Agent Q-network

To fully utilize the graph structural information and consider the interactions between different agents, we propose a novel multi-agent Q-network consisting of a GCN component Q^{GCN} and a GRU component Q^{GRU}. The overall architecture is shown in Figure 1. Q^{GCN} is capable of exploiting the interconnections between nodes. For iteration $t = 1, \cdots, T$, the state s_t is fed into Q^{GCN} to obtain the Q-value of the first agent $Q_1(s_t, u_t^1)$ and the graph embeddings of candidate nodes C_t. The first selected node is denoted as u_t^1 and the corresponding graph embedding C_t^1 is used as the initial hidden state h_t^1 of Q^{GRU}. Q^{GRU} is employed to estimate the Q-value of the remaining $n-1$ agents. The choice of GRU architecture enables it to aggregate information from the already selected nodes. For agent i, the graph embeddings are input to Q^{GRU} together with hidden state h_t^{i-1}. Action u_t^i is made based on the Q-Value $Q_i(s_t, u_t^i)$. The embedding of selected node E_t^i that is output from Q^{GRU} represents the aggregated information of the already selected nodes. It is further used as the hidden state h_t^i. Hidden state h_t^i enables us to consider the actions of the previous agents. Hence, we could take the interaction between different agents into account. The joint action $\mathbf{u}_t$ is the union of individual action taken by each agent. For the total Q-Value Q_{tot}, we propose to approximate it with the average of individual Q-Value Q_i:

$$Q_{tot}(s_t, \mathbf{u}_t) \approx \frac{1}{n} \sum_{i=1}^{n} Q_i(s_t, u_t^i). \tag{6}$$

Multi-Agent DQN Training

We use deep Q-learning (Mnih et al. 2013) to learn an optimal multi-agent policy. Q-learning is an off-policy learning algorithm and it aims to fit the Bellman optimality equation as follows.

$$Q^*(s_t, \mathbf{u}_t) = r(s_t, \mathbf{u}_t) + \gamma \max_{\mathbf{u}'} Q^*(s_{t+1}, \mathbf{u}'). \tag{7}$$

To train the Q-network, we build a replay buffer ε which stores the transition tuples $\tau_k = (s_t, \mathbf{u_t}, r_{t+1}, s_{t+1})$ and optimize the Q-network with a loss based on temporal difference (TD) error (Sutton 1988):

$$\mathbb{E}_{\tau_k \sim \varepsilon}[(Y_t - Q(s_t, \mathbf{u}_t)^2], \tag{8}$$

where Y_t is the TD target. In our experiments, we observe that the training of Q-network is unstable due to the overestimation phenomenon. To address this issue, we use double DQN (Van Hasselt, Guez, and Silver 2016) to stabilize the training process. In double DQN, the max operation in the target is decomposed into action selection and action evaluation. We use the online network to select actions and evaluate the Q-values with the target network. With double DQN, the TD target becomes:

$$Y_t = r(s_t, \mathbf{u}_t) + \gamma Q(s_{t+1}, \arg\max_{\mathbf{u}_{t+1}} Q(s_{t+1}, \mathbf{u}_{t+1}; \theta_t); \theta_t'), \tag{9}$$

where θ_t is the parameter of the online network and θ_t' is the parameter of the target network.

Similar to (Hu et al. 2020), we adopt a transferable active learning scheme. Specifically, we use a source graph $\mathcal{G}_s$ with full label information to train our DQN and directly test it on the target graph $\mathcal{G}_t$ without any adaptation. Our overall framework is summarized in Algorithm 1.

Experiment

Experiment Settings

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,278	1,433	7
Citeseer	3,327	4,676	3,703	6
Pubmed	19,718	44,327	500	3
Reddit	4,584	19,460	300	10
Co-CS	18,333	81,894	6,805	15
Co-Phy	34,493	247,962	8,415	5

Table 2: Statistics of the datasets used in our experiments.

Algorithm 1: BIGENE: Batch Active Learning with GNNs

```
Input: Target graph $\mathcal{G}_t$, adjacency matrix $A$, node feature matrix $X$, multi-agent Q-network trained on source graph $\mathcal{G}_s$, unlabeled pool of samples $U$, initial number of samples $M$, number of iterations $T$, number of samples in a batch $n$.
1: $L_0 \leftarrow M$ samples selected randomly from $U$ and query the oracle for their labels.
2: Train an initial GNN model $\theta_0$ on $L_0$ with the cross entropy loss.
3: for $t = 1, 2, \cdots, T$: do
4:     For each node $v_i$, compute the embedding $x_i$ with current model $\theta_{t-1}$.
5:     Compute the state $s_t$ based on Eqs. (2)-(4)
6:     Compute the $Q$-value $Q_1(s_t, u_t^1)$ of the first agent with $Q^{\text{GCN}}$ and obtain the graph embeddings of candidate nodes $C_t$.
7:     Initialize the hidden state $h_t^1$ of $Q^{\text{GRU}}$ as the graph embedding $C_t^1$ of the first selected node $u_t^1$.
8:     for $i = 2, \cdots, n$: do
9:         Feed $C_t$ together with $h_t^{i-1}$ into $Q^{\text{GRU}}$ and compute the $Q$-value $Q_i(s_t, u_t^i)$ of the $i^{\text{th}}$ agent.
10:        $h_t^i \leftarrow E_t^i$ the corresponding embedding from $Q^{\text{GRU}}$ of $u_t^i$.
11:    end for
12:    Query the oracle for the labels of $\{u_t^1, u_t^2, \ldots, u_t^n\}$ and update labeled dataset $L_t = L_{t-1} \cup \{u_t^1, u_t^2, \ldots, u_t^n\}$.
13:    Train the GNN model $\theta_t$ on $L_t$.
14: end for
15: return Final GNN model $\theta_T$ on $\mathcal{G}_t$.
```

Datasets. We consider 6 widely used benchmark datasets, including Cora, Citeseer, Pubmed[1], Reddit[2], Coauthor-CS (Co-CS) and Coauthor-Physics (Co-Phy) (Shchur et al. 2018). In the first three citation datasets, each node represents a document and edges between them represent citation links. For Reddit dataset, the posts are regarded as nodes and two posts are connected with an edge if there are at least two users both of whom comment or post them. We follow (Hu et al. 2020) to process Reddit dataset in our experiments. Co-CS and Co-Phy are co-authorship networks where each node represents an author. There is an edge between two nodes only if they have coauthored with each other. Table 2 summarizes the statistics of the datasets. We train our BIGENE model and GPA on Cora dataset and evaluate them together with other baselines on the other five datasets. Our BIGENE works in both same domain transfer setting (Citeseer, Pubmed) and different domain transfer setting (Reddit, Co-CS, Co-Phy).

Baselines. We compare our proposed BIGENE method with the following baselines. We use PyTorch (Paszke et al. 2019) as our backend and all experiments were conducted

[1]https://linqs.soe.ucsc.edu/data

[2]https://bit.ly/3bumUtv. Meta was not involved in the creation of the dataset.

on a server with NVIDIA Tesla V100 SXM2 GPU.

1. **Random**: In each iteration, randomly select nodes to be labeled.
2. **Entropy-based method (Entropy)**: In each iteration, select nodes with the highest prediction entropy from the classification model.
3. **Centrality-based method (Centrality)**: In each iteration, select nodes with the highest centrality scores.
4. **Coreset (Sener and Savarese 2017)**: For each candidate node, calculate the minimum distance between the node's embedding and labeled nodes' embeddings, and select nodes with the maximum distances.
5. **AGE (Cai, Zheng, and Chang 2017)**: AGE combines three criteria to select samples, including the prediction entropy, the node centrality score, and the distance between the node's embedding and its nearest cluster center. AGE introduces a time sensitive parameter to balance these three criteria and select nodes with the highest weighted sum of three criteria.
6. **GPA (Hu et al. 2020)**: GPA formulates active learning problem as a Markov Decision Process (MDP) and trains a policy network to learn the optimal selection strategy. Since GPA is proposed for selecting nodes one by one, we adopt its batch-mode variant which selects a batch of nodes with the highest action probabilities in each iteration.

Parameter Settings. For deep Q-network, the GCN component is implemented as a two-layer GCN, and we set the hidden layer size to be 8. The hidden size of the GRU component is also set to be 8. Discount factor γ is set to be 0.99. We use Adam (Kingma and Ba 2014) as the optimizer with learning rate 0.001 and batch size 16. The GCN classification model contains one hidden layer with a size of 16. We train it with Adam optimizer and set the learning rate to be 0.01. For batch active learning setting, we set the initial number of samples $M = 20$.

Evaluation Metrics. We use 1,000 labeled nodes as the test set and randomly sample 500 labeled nodes from the remaining nodes for validation. We run 50 independent experiments with different initializations for the classification model and report the average classification accuracy on the test set.

Results and Analysis

Fixed Active Learning Batch Size. Firstly, we set the active learning batch size n to be 20. Figure 2 show the performance comparisons with different labeled nodes for training. We could see that our proposed BIGENE method consistently outperforms other baselines under different query budgets. The advantages are more obvious when the query budget is small. For instance, when the total number of labeled nodes is 60, our BIGENE outperforms the best baseline by at least 2% on Co-CS and Co-Phy. In addition, since Pubmed, Co-CS, and Co-Phy have more candidate nodes to select, the performance improvement brought by BIGENE is more evident. The improvement of BIGENE over GPA (the best competitor) is statistically significant on these datasets with a p-value smaller than 0.05. The superior performance of the proposed BIGENE can be mainly attributed to the MARL design scheme. It enables the agents to simultaneously consider the informativeness of single nodes and the interactions between selected nodes in the same batch.

Varied Active Learning Batch Size. We fix the total query budget to be 80 and study the active learning performance under different batch sizes. The batch size is varied from 1 to 20 and results on Reddit and Co-CS are shown in Figure 3. For 'Random' and 'Centrality', the selection strategies are independent of batch size. Therefore, their performances remain the same. We can see that the performance gap between GPA and proposed BIGENE becomes larger when the batch size increases. This is because, the main advantage of BIGENE over GPA is that it takes the interactions between selected nodes in the same batch into account. If the batch size increases, it becomes more critical to encode the batch interaction to avoid selecting redundant samples. GPA slightly outperforms BIGENE when the batch size is 1 in Figure 3(b). This is expected since GPA was designed for selecting unlabeled nodes one by one by formulating the problem as a sequential decision process.

Efficiency Comparison. Here, we compare the efficiency of our BIGENE with the original one-by-one GPA. The running time and classification accuracy are shown in figure 4. 'BiGeNe (10)' and 'BiGeNe (20)' represents performing active learning with batch size 10 and 20 respectively. We can see that our BIGENE is $10\times \sim 20\times$ faster than GPA without loss of performance. This demonstrates the scalability of our BIGENE when applied to large-scale graphs.

Comparison of Different RL Algorithms. We also investigate the contribution of our proposed multi-agent design scheme. We adopt the batch active learning algorithm in (Casanova et al. 2020) for images as the baseline 'DQN'. The idea is to divide the candidate nodes into n disjoint groups and select one node from each group. The reward of the single node is approximated as the reward of the entire batch. Results on Co-CS and Co-Phy are shown in Figure 5. We can see our BIGENE outperforms the baseline by a clear margin, especially with small query budgets. The reason is that individual selections may receive spurious reward signals due to the approximation method used in (Casanova et al. 2020). The improvement of the batch samples is actually attributed to the other nodes in the same batch. As a result, it misleads the learning process. Furthermore, baseline 'DQN' decomposes the batch selection into independent individual selections which might overlook the interactions between selected samples in the same batch.

Ablation Study of State Representations. We conduct experiments on Pubmed and Co-Phy to study the contribution of each metric in the state representations. In Figure 6, 'centrality' means that we only use node centrality, 'centrality+entropy' means that we use both prediction entropy and node centrality, and 'centrality+entropy+diversity' means that we use all the three metrics including the node

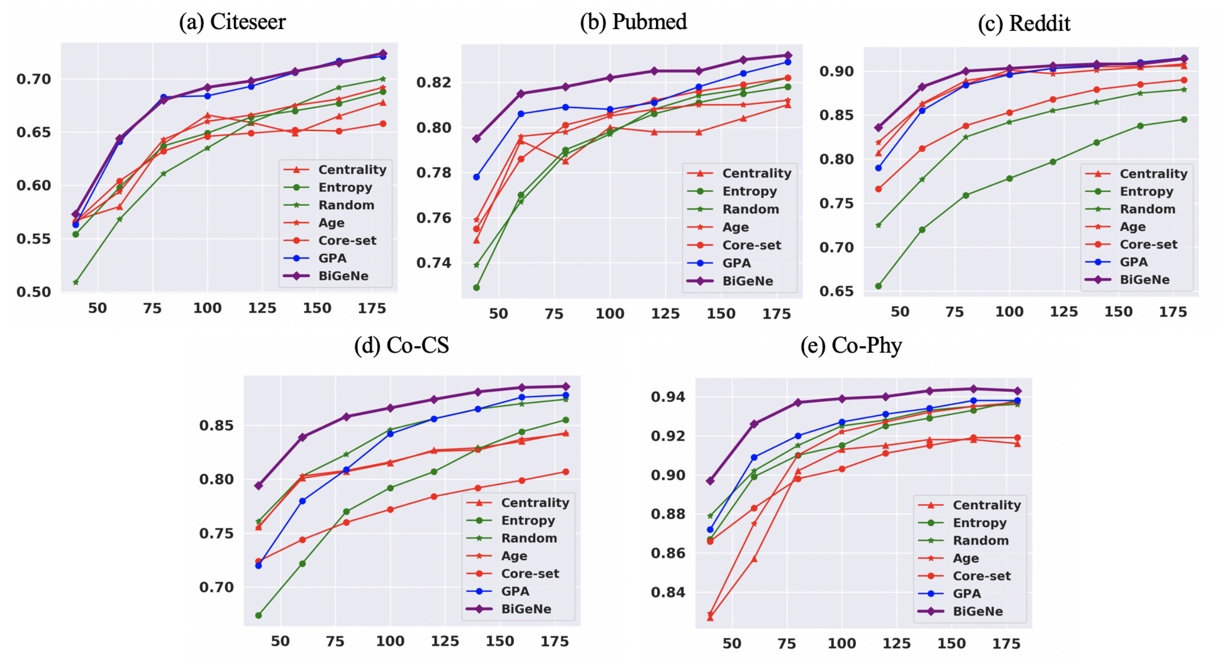

Figure 2: The classification performance comparisons with different number of labeled nodes for training. The x-axis represents the number of queried samples, the y-axis represents the classification accuracy. Our method BIGENE is marked as the purple line.

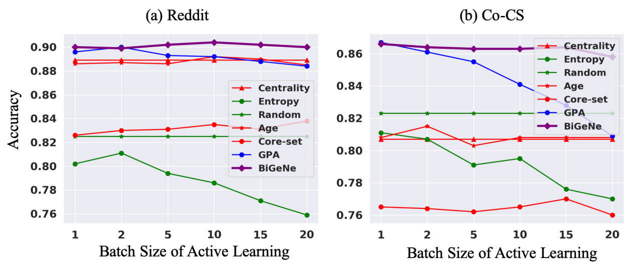

Figure 3: Results of different batch sizes with active learning. The query budget is fixed to 80.

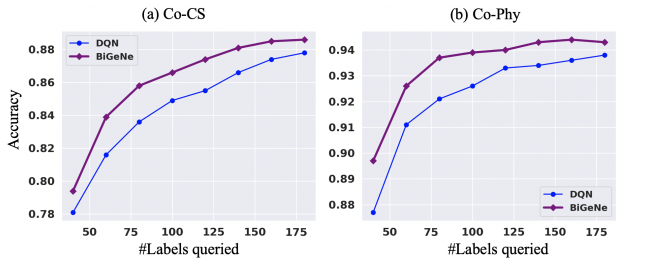

Figure 5: Results of different RL algorithms.

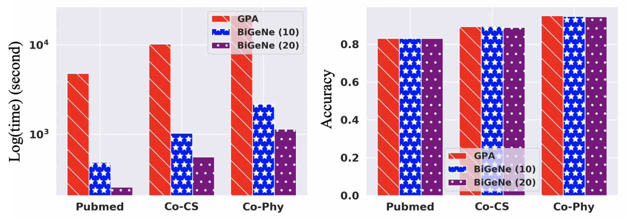

Figure 4: Efficiency comparison between GPA and Our BIGENE.

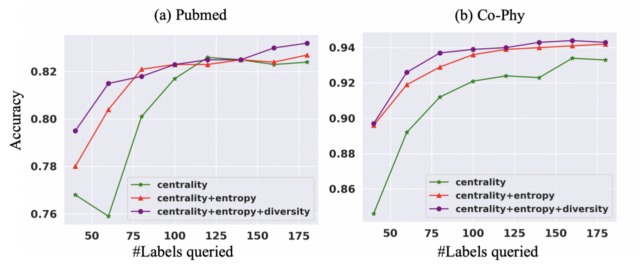

Figure 6: Results of ablation study on state representations.

embedding distance. As illustrated in Figure 6, with the increasing number of metrics in the state representation, the performance of the classification model improves. This is because, these three metrics are designed from different perspectives, and thus mutually complement each other well.

Conclusion

In this paper, we study batch active learning for GNNs and propose a novel reinforced batch-mode active learning method named BiGeNe. We formulate the problem as a cooperative multi-agent reinforcement learning problem and introduce a value decomposition method to avoid the combinatorial explosion of the joint action space. Besides, we propose a novel multi-agent Q-network which considers both informativeness and batch interaction simultaneously. Our BiGeNe consistently outperforms other active learning methods with different batch sizes. Comparison of different RL algorithms verifies the effectiveness of our proposed multi-agent design. In the future, we plan to extend our current method to deal with more complex graph-structured data such as knowledge graphs.

Acknowledgments

The work of Y. Zhang and H. Tong is supported in part by National Science Foundation under the grants 1947135 and 2134079, by NIFA award 2020-67021-32799, and Facebook. The work of Y. Chi is supported in part by ONR under the grant N00014-19-1-2404, and NSF under the grants 1901199, 2134080 and 2106778. The work of Lei Ying is supported in part by NSF under the grants 2003924, 2112471, and 2134081. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Facebook, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Aggarwal, C. C.; Kong, X.; Gu, Q.; Han, J.; and Philip, S. Y. 2014. Active learning: A survey. In *Data Classification: Algorithms and Applications*, 571–605. CRC Press.

Brandes, U. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2): 163–177.

Cai, H.; Zheng, V. W.; and Chang, K. C.-C. 2017. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085*.

Casanova, A.; Pinheiro, P. O.; Rostamzadeh, N.; and Pal, C. J. 2020. Reinforced active learning for image segmentation. *arXiv preprint arXiv:2002.06583*.

Chen, X.; Yu, G.; Wang, J.; Domeniconi, C.; Li, Z.; and Zhang, X. 2019. Activehne: Active heterogeneous network embedding. *arXiv preprint arXiv:1905.05659*.

Fang, M.; Li, Y.; and Cohn, T. 2017. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*.

Gao, L.; Yang, H.; Zhou, C.; Wu, J.; Pan, S.; and Hu, Y. 2018. Active discriminative network representation learning. In *IJCAI International Joint Conference on Artificial Intelligence*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

Hu, S.; Xiong, Z.; Qu, M.; Yuan, X.; Côté, M.-A.; Liu, Z.; and Tang, J. 2020. Graph Policy Network for Transferable Active Learning on Graphs. *Advances in Neural Information Processing Systems*, 33.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Konyushkova, K.; Sznitman, R.; and Fua, P. 2018. Discovering general-purpose active learning strategies. *arXiv preprint arXiv:1810.04114*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4602–4609.

Oliehoek, F. A.; and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 4295–4304. PMLR.

Samvelyan, M.; Rashid, T.; De Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G.; Hung, C.-M.; Torr, P. H.; Foerster, J.; and Whiteson, S. 2019. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*.

Sener, O.; and Savarese, S. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D. E.; and Yi, Y. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, 5887–5896. PMLR.

Stephenson, K.; and Zelen, M. 1989. Rethinking centrality: Methods and examples. *Social networks*, 11(1): 1–37.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1): 9–44.

Tong, H.; Faloutsos, C.; and Pan, J.-Y. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*, 613–622. IEEE.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Wang, J.; Ren, Z.; Liu, T.; Yu, Y.; and Zhang, C. 2020. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*.

Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; and Freitas, N. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, 5165–5175.