ActiveRGCN: Finding Valuable Samples for Heterogeneous GNN Training

Xinyu Zhao
University of California, Los Angeles
California, USA
xinyuz9611@g.ucla.edu

Haowei Jiang
University of California, Los Angeles
California, USA
hwj@g.ucla.edu

Nuocheng Pan
University of California, Los Angeles
California, USA
np9@g.ucla.edu

Hang Zhang
University of California, Los Angeles
California, USA
johnnyzhang0611@g.ucla.edu

ABSTRACT

Active Learning has been proved effective in representation learning applications including heterogeneous information network embeddings by proposing a framework to select the most valuable node for labeling to reduce the labeling cost while maximizing task performance. Inspired by a state-of-the-art active learning framework on Heterogeneous Information Network ActiveHNE, we propose ActiveRGCN as an extension to the classic heterogeneous network embedding method RGCN. ActiveRGCN selects the most informative nodes for labeling information query according to three node selection metrics based on uncertainty and representativeness. Different metrics are combined with dynamically-updated weights by learning the effectiveness of each metric using the multi-armed bandit mechanism. Experiments on three real-world datasets show within the budget of 30% of overall training dataset, ActiveRGCN outperforms baseline in aspect of effectiveness and efficiency of ActiveRGCN, also proving its extendability to other Graph Neural Network methods . We design thorough ablation studies for the active selection strategy to investigate its validity, showing the CIE metric has the best performance among all and dynamic weights update mechanism needs further improvement.

KEYWORDS

Active Learning, Heterogeneous Information Network, Graph Convolutional Networks

ACM Reference Format:
Xinyu Zhao, Haowei Jiang, Nuocheng Pan, and Hang Zhang. 2018. ActiveRGCN: Finding Valuable Samples for Heterogeneous GNN Training. In *Proceedings of Current Topics in Data Structures (CS249).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the advance in representative learning, network embedding is applied to a wide range of areas using networks as an underlying data structure, e.g. social network model in social media[4], knowledge graph in recommendation tasks[12, 14], and so on. Network embedding embeds a network into a low-dimensional space preserving the network structure and inherent information, which makes it helpful in solving downstream network analysis tasks like node classification, link prediction, etc.

The end-to-end Graph Neural Network(GNNs)[13] becomes a technical trend recently to perform feature transformation and neighborhood aggregation. We may take advantage of the node labels to improve the performance of GNNs, but the requirement of labelled information brings new challenge since it is not always the case that the labelled information is given in real-world scenario and it is labelled with the involvement of human experts.

Active learning (AL), a technique widely used in representative learning to acquire labels of nodes during learning, is proposed to solve this problem[1, 6, 9]. Given a labelling budget, the active learning framework is designed to optimize the performance of GNNs by selecting the most valuable nodes with some well designed metrics.

Most GNNs[5, 8] and AL approaches are designed for homogeneous network with only a single type of nodes and relationships, while GNNs and AL framework are rarely studied for heterogeneous information networks (HINs)[11]. Compared with homogeneous information networks, HINs are closer to more real-world scenarios because they contain diverse node types and relationships between nodes. However, the structure of HIN also brings challenges to the network embedding and AL because different node types and different edge types should be treated differently.

ActiveHNE[3] makes the first attempt to apply AL to their proposed heterogeneous GNN. However, despite its good performance, ActiveHNE discusses little on the relationship between the active select strategy and the proposed heterogeneous GNN, leaving an open problem whether the active select strategy can be applied to other backbone heterogeneous GNN architecture. Moreover, its unfair experimental setting and inappropriate evaluation weaken their claims on the effectiveness of the active select strategy.

In this paper, we propose ActiveRGCN, a new active learning framework for heterogeneous network embedding. The method is largely motivated by the active learning framework of ActiveHNE, but is equipped with the widely used Relational Graph Convolutional Network (RGCN) as the backbone heterogeneous GNN to improve its performance specifically on node classification tasks. By careful engineering, this framework can be extended to other network embedding solutions. Carefully designed experiments are conducted to prove the effectiveness of active learning on HIN by comparing baselines all in heterogeneous experimental settings. The contribution of this paper can be summarised as follows:

- We propose the active learning framework ActiveRGCN on top of the classic RGCN model. The proposed active learning setting can be further extended to other heterogeneous GNN backbones.
- We systematically discuss the experiment settings on information score-based active learning on HIN. We conduct comprehensive experiments based on the setting, proving the effectiveness of ActiveRGCN framework over the non-AL setting and ActiveHNE.

2 PRELIMINARIES

2.1 Node Classification on Heterogeneous Graphs

In this paper, we mainly focus on the semi-supervised node classification tasks in heterogeneous graph. Comparing to homogeneous graph, a heterogeneous graph is a directed graph which has multi-type of both edges and nodes[15]. A heterogeneous graph is much closer to a real-world scenario. In an example scenario, we are given a heterogeneous graph representing the nodes and structural relations between them, a designated type of node to find, and some group of labeled nodes in the same types as a subset of all nodes. We then pick some of the training data under the budget allowance to further supervise the learning of a model using the labels which eventually help the prediction of labels on the rest of originally unlabeled nodes.

2.2 Heterogeneous Information Network

In the field of homogeneous information network, the Graph Neural Networks (GNNs)[13] applies Deep Neural Networks on the nodes of the graph and aggregating the information from neighbouring nodes, granting much powerful embedding, and can be used to perform induction on fresh new nodes aside from those showed up in the training process, such as Graph Convolutional Network (GCN)[8]. In contrast to single-type nodes in the Homogeneous Information Network settings, Heterogeneous Information Networks (HINs)[11] have directed graphs that have multi-type structural information among multi-type nodes, encompassing multi-typed interactions thus heterogeneity, leading to more study on relational inferences, and especially node classifications where our interests lie.

Network Embedding (NE) is used to capture the intrinsic information of a given network by embedding the given data into low dimension space, thus facilitating analysis of the network such as node classification tasks we care about in this paper. Because HINs are closer to real-world scenarios in which case nodes are classified differently given different types, finding the embedding which in such case known as Heterogeneous Network Embedding (HNE) can be challenging. Most HNE methods are unsupervised, but by providing supervised information one can possibly leverage the performance of classification. Discriminative Heterogeneous Information Network (DHNE)[3] is a semi-supervised network embedding method based on GCN. Due to the nature of heterogeneous settings that nodes of different types are treated differently, DHNE decomposes the original network into homogeneous networks and bipartite networks, and from each of which it learns the nodes and concatenates the results of all of them together as the final output.

2.3 Active Learning and The ActiveHNE Framework

Active Learning (AL) frameworks aims to train a classifier as accurately as possible given a budget number of labeled training nodes to reduce the cost of gaining supervised information. There exists two parts of a typical AL system: A query system to pick a subset of training data that is most informative for label query and an oracle that labels the queried subset. In the homogeneous setting, lots of networks address the AL framework. The informative score-based Active Graph Embedding (AGE)[1] addressed and solved the above challenge in reducing such cost, with two popular criteria, uncertainty and representativeness. Graph-Partition-based GraphPart[9] separated the graph into several partitions based on modularity without introducing additional hyperparameters which can be crucial in the absence of validation data. GPA[6] modeled AL problem as a Markov Decision Process and train the selection policy π by using Reinforcement Learning method.

In our context, an active select strategy uses AL that finds budget size of nodes with the highest value of reward function consisting of a weighted sum of 3 different rewards, from which train and maximize the performance of each iteration, specifically in a heterogeneous setting. The weights of the reward function are trained by the AL method with not only the graph structure but also the embedding.

ActiveHNE is a DHNE network that incorporates AL strategy to leverage the challenge posed by the expensive cost of label acquisition due to the intervention of human experts. Specifically, it takes both the advantage of the Active Selection Strategy for selecting a valuable subset of nodes to train on and heterogeneous graph for rich dependency of nodes. By iterative query the nodes set, it updates its selection strategy solving Multi-Arm Bandit problem (MAB)[2] and finds the best batch of nodes to query. Researchers of ActiveHNE[3] claim it has better performances using baselines such as GCN.

3 METHOD

3.1 Overview

We thus propose ActiveRGCN, which applies AL strategies on the RGCN backbone in the heterogeneous setting on the node classification task. It includes two main blocks: Active Select Strategy, which will select most valuable nodes to train in the budget allowance in each iteration; RGCN, a heterogeneous GNN backbone which will handle the semi-supervised node classification task. The RGCN backbone and the active select strategy will work together and enhance each other, specifically the active select strategy will take

Dataset	Entities	Edges	Entity Types	Edge Types	Class Numbers	**Training Set Size**
MovieLens	28,491	138,352	4	4	3	918
DBLP	37,791	170,803	4	3	4	1014
Cora	56,670	244,088	3	3	10	3911

Table 1: Details of the datasets.

use of the learned embedding from RGCN backbone, and the backbone model will take the new labeled set (training set in the budget allowance) selected by the strategy and learn the new embeddings. An overall workflow of the whole framework is shown in the Figure 1. In the following sections we will explain each components in detail.

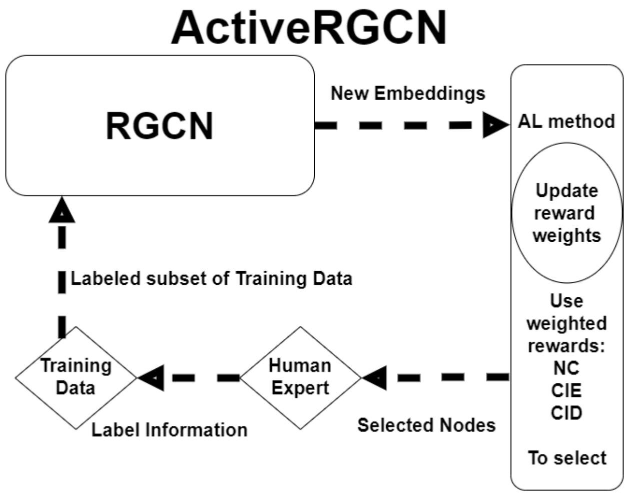

Figure 1: An example workflow and building blocks of ActiveRGCN framework.

3.2 Relational Graph Convolutional Network

Relational Graph Convolutional Network (RGCN)[10] is a model that applies to neighbourhood exploiting the locality of a graph that predicts upon large number of relational data under a heterogeneous setting, as an extension of GCN on homogeneous counterpart. RGCN focuses on directed edges in the graph for modelling the directed relations, aggregate each of them independently while fetching the neighbourhood information, such that it models relational data specifically for the purpose of link prediction and entity classify task. Just as GCN works as a good baseline in the field of homogeneous settings, RGCN works great in the line of HINs. In this paper, we are more interested in the entity classify task, or node classification.

3.3 Active Select Strategy

We started by a close examination of both the concept and codes of active select strategies employed to train more effective DHNE by proposing an active query of nodes in the ActiveHNE context, and adapt the active select strategy to RGCN. Uncertainty is a measure of a node whose classification is least certain to the current model. Representativeness is a selection of nodes that can well represent the overall patterns among all the nodes. Both uncertainty and representativeness are popular measures used in AL. In the light of that spirit, three of the following rewards are used.

Convolution Parameters. In a spectral graph convolution, signals are convoluted weighted sum or neighbours. To take the advantage heterogeneous graph node dependencies, two of the three rewards will be calculated in a convoluted manner in the form of weighted sum of neighbours. The weight parameters denoted as $w_i = tanh(\frac{n_i}{N} + \frac{m_i}{V_T}) \in [0,1)$[3] for node v_i. n_i is number of neighbour nodes of v_i and m_i is number of different types in neighbour. N and V_T are total number of nodes and number of types in the whole graph respectively.

Individual Rewards Description. [3]

- **Network Centrality (NC).** It is commonly used measure for evaluation of nodes in a graph, by simply measuring the degree of the graph, in the following $\mathcal{N}_i$ is the set of neighbour hood nodes of v_i.

$$\phi_{nc}(v_i) = |\mathcal{N}_i| \tag{1}$$

- **Convolutional Information Entropy (CIE).** It is an adaption using convolution method and parameters introduced, to Information Entropy (IE) which is a commonly used metric for uncertainty. Specifically, it is convolutional weighted sum of neighbouring uncertainties. Here $\mathbf{F}_{ic}$ denotes the probability of node v_i belongs class c of all C classes in current model.

$$\phi_{cie}(v_i) = \sum_{v_j \in \{v_i \cup \mathcal{N}_i\}} w_j(-\sum_{c=1}^{C} \mathbf{F}_{jc}(log\mathbf{F}_{jc})) \tag{2}$$

- **Convolutional Information Density (CID).** A measure of representativeness of nodes. Unlike NC is in the structural space, CID is in the embedding space. Specifically, it is a k-mean clustering on the embeddings to calculate Information Density (ID) which is common in finding representativeness. The number of clusters used is the same as the number of class labels. Thus the CID of given v_i based on the neighbour nodes is given as:

$$\phi_{cid}(v_i) = \sum_{v_j \in \{v_i \cup \mathcal{N}_i\}} w_j \frac{1}{1 + dis(\mathbf{E}_j, \varphi(v_j))} \tag{3}$$

dis is a Euclidean distance in embedding space, $\varphi(v_i)$ is the center vector of the cluster to which v_i belongs, $\mathbf{E}_j$ is the embedding of node j.

Reward Function. Then the reward function is a measure of importance of nodes in the current model, meaning it is more beneficial for us to choose nodes with high rewards. The reward function is then a sum of 3 weighted previously defined rewards:

$$Reward = w_{nc}\phi_{nc} + w_{cie}\phi_{cie} + w_{cid}\phi_{cid} \tag{4}$$

Then the problem becomes how to select the weights wisely so that we can jointly select the most valuable nodes. Each weight captures the importance of the corresponding strategy, so it is a balance of the importance of each strategy, which can be solved through the Reinforced Learning (RL) based method as follows.

Multi-Armed Bandit Problem (MAB). It is a well-known problem in the field of RL problems, which states what a player should do with the arms to maximize the cumulative reward of a multi-armed bandit machine given a budget number of runs (iterations) with feedback of reward of each iteration under the constraint player can only manipulate one arm during each iteration. Combinatorial MAB is an extension of the MAB problem without the constraint, allowing players to play with multiple arms during each iteration.

In our context, each arm corresponds to one of the selection rewards, and we can approximate the importance (weight) of each selection reward by estimating the utility of the corresponding arm in the MAB problem with RL method[3].

3.4 The ActiveRGCN Framework

In our propsed framework ActiveRGCN, it applies a active select strategy described above to select valuable nodes in the training set to train for RGCN, instead of randomly selecting or simply going through each one of them. Compared to ActiveHNE, we will use different hyperparameters, similar active select and rewards function but a variety of ways to select weights for each reward, and finally apply on RGCN network instead of DHNE.

In implementation wise, we extracted the Active Select strategy from the ActiveHNE codes, and connect it with the RGCN network (and possibly all the networks that work with DGL library) so it can read embeddings and choose nodes in the RGCN, while preserving its own interface to the datasets used for demonstrating ActiveHNE, and DHNE network for experiment purposes. We also enabled ourselves with the ability of hyperparameter tuning to ensure we can avoid the experiment setting problem of ActiveHNE later. As a result, most of the work involves reading and writing a huge amount of codes to get a full understanding of Active Select, DHNE, and RGCN in the node classification domain.

Workflow and Hyperparameters To give a better understanding of the active select process, and how it interact with RGCN backbone, we introduce the following piece of learning process with all involved hyperparameters.

- for **ITER** times:
  - Select **BATCH** nodes from *Training Set* have top *Rewards*
  - Add selected nodes to *Labeled Set*, remove them from *Training Set*
  - Create new *Model* (*Model* on previous **ITER** is destroyed)*
  - for **EPOCH** passes:
    - Train *Model* on *Labeled Set* once
  - Take *Embedding* from *Model* through *RGCN*
  - Update *Weights* in *Rewards* from *Embedding* using RL method based on MAB

Here **ITER**, **BATCH**, **EPOCH** are hyperparameters we get to choose. **ITER** is the total number of iterations, which is also the number of times *Labeled Set* size get updated. **BATCH** indicates the number of new nodes to query during each iteration, which is also the size of increase in *Labeled Set* during each iteration. **EPOCH** indicates simply the number of passes on the *Labeled Set* for the model in each iteration.

3.5 ActiveHNE Limitations

Comparing to ActiveRGCN, ActiveHNE have following limitations or leads to an possible extension or improvement.

Experiment Settings. As far as we concerned, the experiment settings are not fair enough. ActiveHNE is a framework works under the heterogeneous setting, but the baseline it is using to compare against such as GCN is under the homogeneous setting. If one want to show that active select is working under heterogeneous settings, it is better to compare a baseline in heterogeneous settings, such as using RGCN instead of GCN.

Reward Function Performance. Whether or not MAB really solves the problem of finding the best weights for each of the rewards so that we can pick the most important nodes, is an issue left unaddressed. We want to explore if MAB has a beneficial outcome in training the reward weights for active selection, comparing to, for example, a reward functions that evaluate all 3 rewards equally (with the same weight).

The generality of Active Select. Active Select is a really nice scheme, possibly an API that can apply to many different networks even in the heterogeneous setting like ActiveHNE. We want to explore the possibility of making it work on other commonly used heterogeneous networks besides DHNE such as RGCN to prove the feasibility of such combination, both implementation-wise and performance-wise.

4 EXPERIMENT

In this section, our target is to answer three following questions:

- **Q1.** Does the active select strategy work completely in the HIN setting?
- **Q2.** Can ActiveRGCN performs as effectively as ActiveHNE?
- **Q3.** Why does the active select strategy work?

4.1 Experiments Setup

Dataset. We adopt the dataset setting from ActiveHNE and evaluate ActiveRGCN on three HINs extracted from MovieLens, DBLP, and Cora. In the experiment, we focus on the node classification task and evaluate the performance of each method using the accuracy of node classification. In particular, we classify "movie" nodes in the MovieLens dataset into three classes, "author" nodes in DBLP into four classes and "paper" nodes in Cora into ten classes. We randomly select 25% of the labeled nodes as the training set, 25%

as the validation set and the remaining as the testing test without overlapping. The detailed dataset information can be seen in Table.1.

Baselines. We compare ActiveRGCN against ActiveHNE and the naive RGCN. To adapt RGCN in Active Learning settings, for each training, nodes are **randomly** selected for query in each iteration with a given budget.

For ActiveHNE, we follow the hyperparameters used in [3]. For ActiveRGCN and RGCN, we train our model using a network with two convolutional layers with the dimensionality of 16 and C, where C is the number of classes. We apply the dropout probability of 0.5 to the training process and optimize the network y using Adam [7] optimizer.

One unfair experiment setting in ActiveHNE is that for each iteration, it trains models for 200 epochs with batch size $b = 20$ for Cora and MovieLens and $b = 5$ for DBLP and selects the training result of the last epoch for comparison. However, in our preliminary experiments on checking the experiment settings, we find that their experiment setting has the potential of overfitting, like the example results shown in Figure 2. Thus in our experiments for each iteration, we train models for 50 epochs with batch size $b = 80$ for Cora and $b = 20$ for DBLP and Movielens, selecting the best model with the lowest validation loss for comparison.

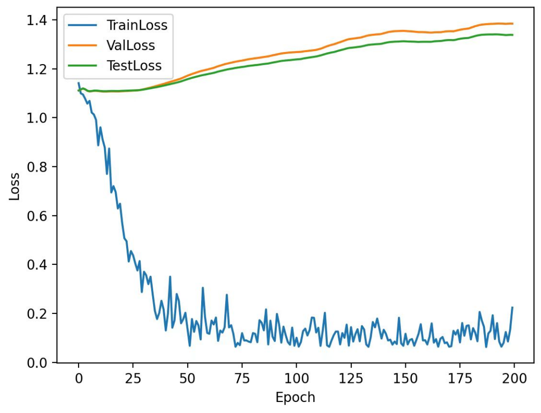

Figure 2: An example of loss curve in one training iteration under Cora Dataset. After 50 epochs, the validation and test loss increase, indicating the overfitting.

4.2 Effectiveness of ActiveRGCN

To answer **Q1**, we compare ActiveRGCN with RGCN adapted with random node selection as the active select strategy on all three datasets. As we can see from Figure 4, after the first several iterations with a limited budget size, ActiveRGCN outperforms RGCN by 5% in all three datasets, while there is no significant difference in larger training budget. When the budget is big enough, the training set can nearly have all the information that is useful for improving classification accuracy, thus the result in larger training budget meets our expectations. However, since the goal of ActiveRGCN is to improve classification performance with as small training budget as possible, the experiment results prove that ActiveRGCN reaches its goal.

After proving the effectiveness of active select strategy, we can further investigate **Q1** mentioned at the beginning of this section. We compare ActiveHNE and ActiveRGCN on Cora dataset only since the Cora dataset is the most complex one among all datasets and thus has higher representative. As we can see in Figure 3, we can see that at the beginning of the training, ActiveRGCN has better performance than ActiveHNE, but with the increase of iteration, the performance of ActiveHNE reaches that of ActiveRGCN. Although in general the proposed ActiveRGCN has a similar performance to ActiveHNE, ActiveRGCN has a simpler network structure than DHNE proposed in ActiveHNE, requiring less training time and computational resources. We conclude that ActiveRGCN outperforms ActiveHNE in efficiency.

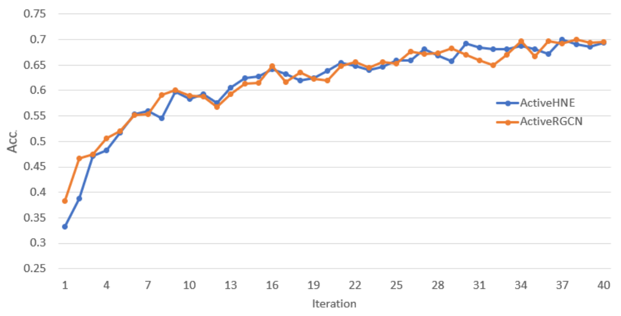

Figure 3: Comparison between ActiveRGCN and ActiveHNE on Cora dataset. ActiveRGCN reaches similar performance with a simpler network, showing its efficiency and extendability.

On the other hand, since the underlying active select strategy is similar between ActiveHNE and ActiveRGCN, this experiment also proves that this active select strategy is not network-structure-depend and has the potential to be applied to other graph neural networks.

4.3 Ablation Study on Active Select

In Section 3.3, the active select combines three node selection metrics: NC, CIE, and CID by using the multi-armed bandit. To validate the effectiveness of each reward together with the multi-armed bandit, we introduce three variants of ActiveRGCN which select nodes only based on one selection metric and are named NC-RGCN, CIE-RGCN and CID-RGCN respectively. In the ablation study, we compare their performance with the original multi-armed bandit ActiveRGCN and the RGCN baseline.

As shown in Figure 5, all three variants show better performance than the naive RGCN in most cases when tested on all three datasets, suggesting that all three node selection metrics contribute to the effectiveness of the active select policy. Among three variants, CIE-RGCN shows the best accuracy in all three datasets with the least variance in general, while the multi-armed bandit can only outperform NC and CID variants. The reason for poor performance of NC and CID could be that, NC is a simple metrics only focus on nodes

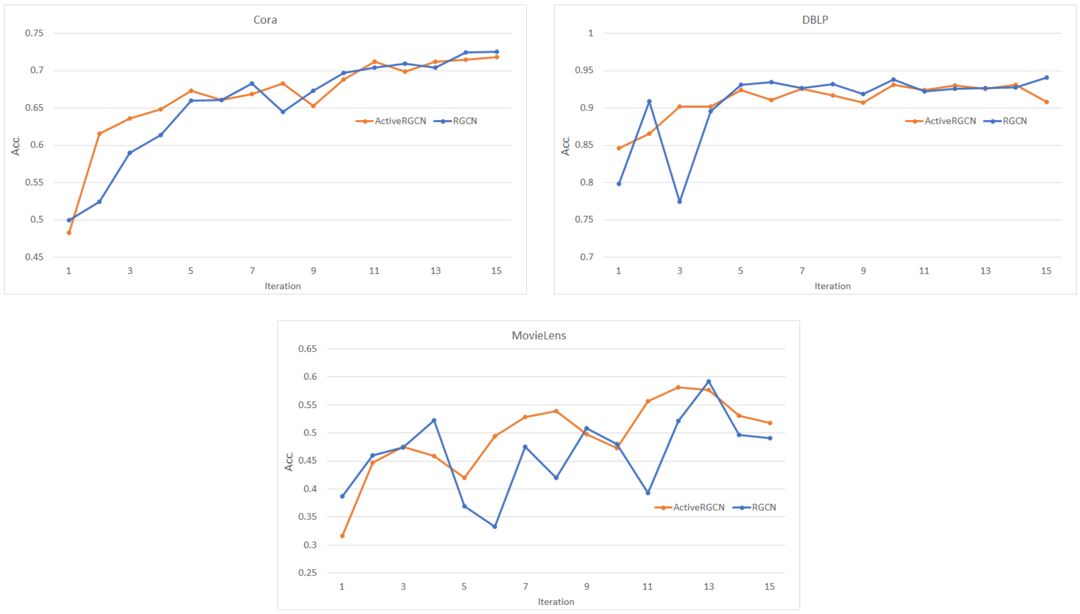

Figure 4: Comparison between ActiveRGCN and RGCN on three datasets. ActiveRGCN outperforms RGCN by 5% with limited budget size.

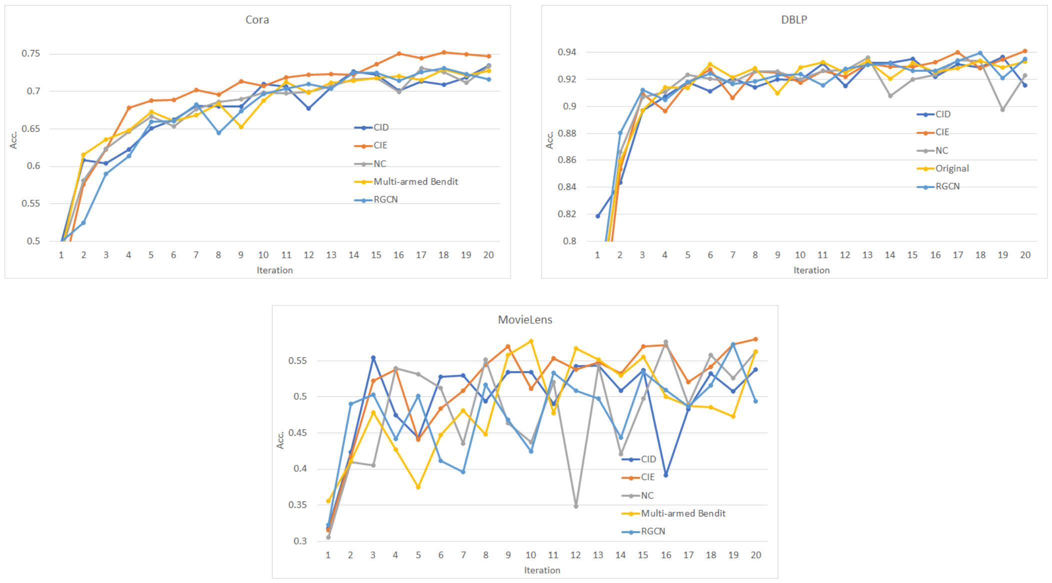

Figure 5: Comparison between all three variants of ActiveRGCN and RGCN on three datasets. All variants show better performance than naive RGCN in most cases, and CIE even outperforms ActiveRGCN on all three datasets in general.

themselves, and the distance metric used in CID doesn't suit for the learnt embedding.

This result suggests that there is a possibility that the multi-armed bandit mechanism doesn't work well as we expected while there are also chances that the result of multi-armed bandit is influenced by the relatively poor performance of NC and CID metrics. To validate the multi-armed bandit, we compare ActiveRGCN with a variant giving the same weights to all three metrics. The results are shown in Figure 6. In three datasets, the same weight variant shows relatively better accuracy over ActiveRGCN while also shows less variance, suggesting that the multi-armed bandit doesn't perform as we expected and needs further improvement on it. One potential reason is that the multi-armed bandit underestimates the effectiveness of NC and CID. Although NC and CID doesn't performs well

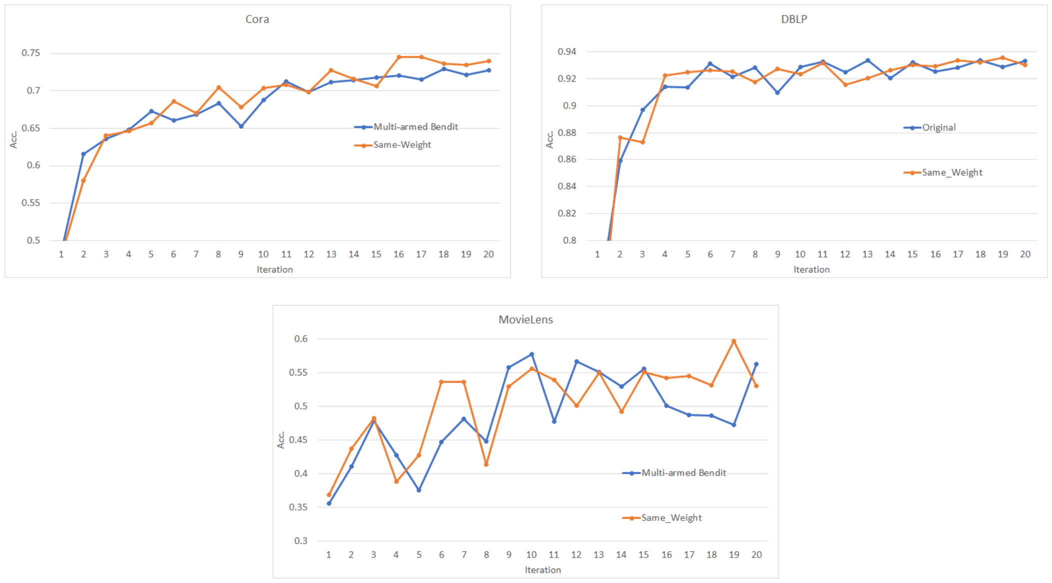

Figure 6: Comparison between ActiveRGCN with its same-weight variant on three datasets. Same-weight variant outperforms ActiveRGCN in all three datasets, indicating that MAB needs further improvements.

independently, their information helps choosing node when they are considered jointly.

5 CONCLUSION

In this paper, we pointed out the problems in existing active learning framework on Heterogeneous Information Network ActiveHNE, and proposed a new framework ActiveRGCN by extending the active learning framework to a new heterogeneous graph neural network. ActiveRGCN shows its effectiveness compared with random node selection and its efficiency compared with ActiveHNE in training and testing. We showed the expandability of the active select framework. We further proved the effectiveness of all three node selection metrics, and pointed out the current multi-armed bandit mechanism still needs further improvement.

As for future work, firstly, ActiveRGCN can be further wrapped up as an API for all GNNs to reduce labelling requirements in real-world HIN applications. Secondly, the dynamic weight update mechanism can be further improved by introducing more advanced Reinforcement Learning methods. Finally, the recently proposed one-step active learning algorithm GraphPart [9] for homogeneous information network can be utilized in ActiveRGCN to speed up the active learning process.

ACKNOWLEDGEMENT

This work is mentored by Yewen Wang. We would like to thank her for her help and support throughout the project.

TASK DISTRIBUTION

Please check Table 2.

Task	People
Cora & MovieLens data preprocessing	Hang
DBLP data preprocessing	Nuocheng
RGCN Pipeline	Hang & Nuocheng
ActiveRGCN implement	Xinyu & Haowei
Pipeline debugging	Xinyu & Haowei
Baseline comparison experiments	Xinyu
Ablation studies	Haowei
Report writing	Nuocheng & Hang

Table 2: Task Contribution

REFERENCES

[1] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2017. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085* (2017).
[2] Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial multi-armed bandit: General framework and applications. In *International conference on machine learning*. PMLR, 151–159.
[3] Xia Chen, Guoxian Yu, Jun Wang, Carlotta Domeniconi, Zhao Li, and Xiangliang Zhang. 2019. Activehne: Active heterogeneous network embedding. *arXiv preprint arXiv:1905.05659* (2019).
[4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.
[5] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
[6] Shengding Hu, Zheng Xiong, Meng Qu, Xingdi Yuan, Marc-Alexandre Côté, Zhiyuan Liu, and Jian Tang. 2020. Graph policy network for transferable active learning on graphs. *Advances in Neural Information Processing Systems* 33 (2020), 10174–10185.
[7] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=SJU4ayYgl

[9] Jiaqi Ma, Ziqiao Ma, Joyce Chai, and Qiaozhu Mei. 2022. Partition-Based Active Learning for Graph Neural Networks. *arXiv preprint arXiv:2201.09391* (2022).

[10] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[11] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. 2016. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2016), 17–37.

[12] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*. 878–887.

[13] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[14] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 974–983.

[15] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 793–803.