Role Equivalence Attention for Label Propagation in Graph Neural Networks

Hogun Park and Jennifer Neville

Purdue University, West Lafayette, IN 47907, USA
{hogun,neville}@purdue.edu

Abstract. Semi-supervised relational learning methods aim to classify nodes in a partially-labeled graph. While popular, existing methods using Graph Neural Networks (GNN) for semi-supervised relational learning have mainly focused on learning node representations by aggregating nearby attributes, and it is still challenging to leverage inferences about unlabeled nodes with few attributes—particularly when trying to exploit higher-order relationships in the network efficiently. To address this, we propose a Graph Neural Network architecture that incorporates patterns among the available class labels and uses (1) a *Role Equivalence* attention mechanism and (2) a *mini-batch importance sampling method* to improve efficiency when learning over high-order paths. In particular, our *Role Equivalence* attention mechanism is able to use nodes' roles to learn how to focus on relevant distant neighbors, in order to adaptively reduce the increased noise that occurs when higher-order structures are considered. In experiments on six different real-world datasets, we show that our model (REGNN) achieves significant performance gains compared to other recent state-of-the-art baselines, particularly when higher-order paths are considered in the models.

Keywords: Node Classification · Label Propagation

1 Introduction

Semi-supervised relational learning methods aim to classify unlabeled nodes in a partially-labeled graph by leveraging information about both the labeled and unlabeled nodes, and their connectivity. In particular, the methods exploit relational dependencies in the graph to jointly make predictions about unlabeled nodes. Prior work on semi-supervised learning in graphs has typically defined relational features via aggregation over the features of neighboring nodes, and then unknown class labels are inferred iteratively using approximate inference algorithms (e.g., Label Propagation (LP) [22], Gibbs sampling [15]). However, many previous methods are limited in their ability to leverage complex neighborhood patterns for learning and inference. For example, while LP works well in simple scenarios, it only exploits direct edges to make predictions on unlabeled examples. While information can propagate across the graph, messages are only passed among direct neighbors, which can be inadequate in complex, sparse

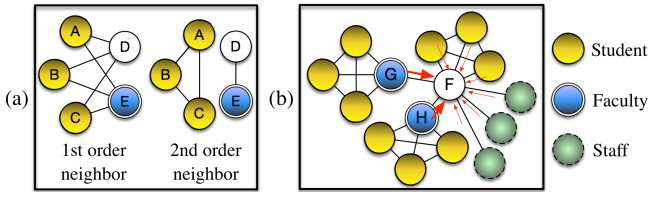


Fig. 1: Examples of capturing roles: (a) by High-order paths (b) by Similarity-based attention. Best viewed in color.

graphs with few labels. Recently, Graph Convolution Networks (GCNs) [10] were proposed to exploit message passing functions to aggregate nearby neighbors and learn a latent representation of each node, which is then used for predicting node labels. However, GCN mainly aims to learn neighbors' attribute patterns—they are not typically used in partially-labeled graphs with few attributes. In this case, *collective inference* is needed during learning, so that patterns among neighbor *class labels* can also be used in the model. In this work, we propose REGNN, a graph neural network architecture for enhancing the accuracy of label propagation, which uses a *role equivalence* attention mechanism to facilitate reasoning with higher-order relationships among labeled nodes.

To move beyond direct neighbors and exploit longer range information in sparse graphs, recent work has effectively incorporated higher-order relationships and paths into relational models (e.g., high-order GCNs [1] and GraRep [2]). Our proposed approach REGNN, considers high-order (or k^{th} order) proximity matrices and extends the existing high-order-based GCNs to leverage inferences about unlabeled nodes via neighborhood at various distances. Since reasoning with higher-order paths (i.e., large k) increases the computational complexity of learning[1], we propose a more efficient mini-batch learning method.

Incorporating higher-order paths can increase the relational signal by considering *nearby* but not directly linked nodes, however, it can also increase noise due to spurious connections as neighborhood size increases. To account for this, and enable the model to *learn* which distant nodes are more relevant, we propose a novel attention mechanism based on *role equivalence*, a social network property that quantifies similarity among nodes based on their relational context. The attention mechanism is used to merge the multiple node representations learned from the set of high-order-based GCNs. Our experiments show that attention based on *role equivalence* works significantly better in the context of label propagation.

Figure 1a-b show examples that high-order path and attention can help to capture roles in the label propagation scenarios. Each node and edge indicate a user and an interaction during a semester, respectively. Note that colors represent class labels, yellow for student, blue for faculty, and green for staff. In Figure 1a, we are trying to predict the label of a user D, who is a faculty. When we use

[1] When considering direct links in sparse graphs, complexity is $O(|E|) \simeq O(|V|)$. However, as higher-order paths connect distant nodes, complexity becomes $O(|V^2|)$.

just direct neighbors, it is not possible to predict the true label of node D by label propagation. However, as the high-order paths from 2^{nd} order neighbors are considered, the label of D could be successfully predicted. Like this example, high-order paths are potentially useful to learn the underlying hierarchical roles such as advisor-student in a citation network and admin-member relationships in a University group on Facebook. When we just stack a GCN layer multiple times, it is also difficult to learn this kind of information because their aggregation is always from direct neighbors. Meanwhile, if a user is surrounded by nodes with diverse class labels, the magnitude of nearby labels can often mislead prediction. In this case, if latent representations are known/estimated, the model can put more importance on neighbors with similar representations. In Fig. 1b, we are trying to predict the label of node F, who is also a faculty member. Although the node F has more students or staffs as neighbors, node G and H are likely to have similar representations to the representation of node F in the latent space, so that they can be weighted more heavily when aggregating. Our aim is to have REGNN exploit both these ideas, by combining high-order paths with a role similarity-based attention layer.

2 Related Work

Semi-Supervised Node Classification Previous semi-supervised node classification algorithms learn a model to predict unknown class labels in a partially labeled graph. For example, LP [22] and ICA [15] estimate labels of unlabeled nodes using the local inference. Recently, graph embedding methods have been proposed to learn low-dimensional representations of nodes by leveraging many relational properties such as random walk (e.g., Node2Vec [8]), high-order paths (e.g., GraRep [2] and NEU [19]), structural similarities (e.g., Struc2Vec [13]).

Graph Neural Networks (GNN) In addition, graph neural networks architectures (e.g., GCN [10]) also have attracted a lot of attention. Recently, high-order path information was also incorporated into GCNs. HA-GCN [23] and N-GCN [1] proposed joint graph neural network architectures that take attributed high-order proximity matrices. However, while high-order GCNs show more robust performance on node classification, computing the high-order paths from the proximity matrices can be quite inefficient.

Attention-based GNNs Graphs are often complex and noisy, so many researchers have incorporated the concept of "attention" into semi-supervised classification. For example, GAT [18] proposed a self-attention-based graph neural network, which decides importance using an edge-wise weighted sum by leveraging rich attributes. However, this attention mechanism has been shown to not be very effective when the data contain attributes with low homophily or there are no attributes. Meanwhile, VAIN [9] proposed kernel-based attention mechanisms for multi-agent modeling. However, they exploit the similarity between nodes along with direct edges only in an attributed graph. In contrast, we design a novel similarity-based kernel based on the concept of *role equivalence* attention and extend it to incorporate neighbors at various distances (thus, high-order paths) in the setting of label propagation.

3 Role Equivalence

First, we include the mathematical definitions of structural equivalence [11] and regular equivalence [5] below. Let $\mathcal{N}(u)$ refer to the neighbors of node u.

Definition 1. *(Structural Equivalence) A pair of nodes u and v are* structurally *equivalent, if the neighbors of node u and v are the same. Thus, u and v are structurally equivalent iff $\mathcal{N}(u) = \mathcal{N}(v)$.*

Definition 2. *(Regular Equivalence) A pair of nodes u and v are* regularly *equivalent if the roles of their neighbors are the same. Let $r(i)$ be the role of node i. Thus, u and v are regularly equivalent iff $\{r(i) \,|\, i \in \mathcal{N}(u)\} = \{r(j) \,|\, j \in \mathcal{N}(v)\}$.*

Regular equivalence states that nodes play the same role if they have connections to nodes with similar roles [6]. There can be many valid ways of grouping nodes into equivalence role sets for a given graph, and regular equivalence is often defined recursively. Based on the above definitions, we can approximate the notion of regular equivalence based on positions in latent space.

Definition 3. *(Role Equivalence in latent (embedding) space) A pair of nodes u and v are* role *equivalent in latent space if their set of neighbors in latent space are the same. If neighbors are defined by distance in latent space, then u and v will have the same neighbors if their representations are equal. Let $f(i)$ be the latent representation of node i. Thus, u and v are role equivalent in latent space iff $f(u) = f(v)$.*

Using Definition 3, we propose a graph neural network architecture with the attention layer based on *role equivalence* among nodes in the following section. Note that the term, *role (or structural)-equivalence*, has been also used in many different ways (e.g., similarities in triangles, betweenness, k-paths, k-stars, k-cliques, subgraph patterns/graphlets, and feature-based MF). In this paper, we use the term *role-equivalence* to refer to Definition 3.

4 REGNN Architecture

Problem Formulation and Notation Given an undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. A is an adjacency matrix of G. V is composed of V_L (labeled vertices) and V_U (unlabeled vertices). Y_L is constructed as a $|V| \times C$ class label matrix. Again, for each labeled node $i \in Y_L$ with class label $y_i = c$, we set $Y_L[i, c] = 1$ and $Y_L[i, \cdot] = 0$ otherwise. If node j is in V_U, we set $Y_L[j, :] = 0$. This Y_L will be fed to our REGNN with the adjacency matrix A. Thus, the goal of REGNN is to estimate the class labels of V_U from Y_L and A, which is a transductive learning setting.

4.1 k^{th} Order GCN Layers for Label Propagation

To learn high-order path-based GCN, we initially construct K different GCN layers. For the layers, adjacency matrices, $A, A^2, ..., A^K$, which have different orders, are fed as input. A^k is obtained by self-multiplying the adjacency matrix A k-times. Then the (i, j) entry of A^k is the number of k-hop paths from i to

j. With these high-order adjacency matrices, we can define $K \times M$ high-order convolution operators. The node representations in m^{th} layer with an adjacency matrix A^k are formulated as

$$H_k^{(m+1)} = \text{ReLU}\left(\hat{A}^k H_k^{(m)} W_k^m\right), \tag{1}$$

where $\hat{A}^k = min\left(\tilde{D}_k^{-1/2}\left(A^k + I\right)\tilde{D}_k^{-1/2}, 1\right)$. In Equation (1), $H_k^{(1)} = Y_L$, and Y_L represents known class labels, which are fed to the first GCN layers. W_k^m is a trainable weight matrix for the m^{th} layer in the k^{th} order GCN, and $\tilde{D}_k$ is the degree matrix of $A^k + I$. The symmetric normalizing trick in $\hat{A}^k$ takes the average of neighboring nodes' representation from each of high-order adjacency matrices. Note that $W_k^m \in \mathbb{R}^{s_{(m)} \times s_{(m+1)}}$ where $s_{(m)}$ is the input representation size at m^{th} layer and $s_{(m+1)}$ is the size of output representation for the next layer. Therefore, W_k^1 could be defined from $\mathbb{R}^{C \times s_{(2)}}$, where C is the number of class labels. This indicates that propagated labels are transformed by the matrix multiplication. The representation of the last GCN layer, $H_k^{(M+1)}$, is additionally passed through another softmax function to normalize the latent representations.

4.2 Role Equivalence Attention Layers

The last layers of REGNN play an important role in jointly learning multiple representations via different high-order paths. In this paper, our role equivalence attention uniquely merges their characteristics in the following ways:

Concatenation Layer Outputs from the previous high-order GCNs are concatenated before they are fed into the self-attention layer. There are K outputs, one from each of the high-order GCNs: H_1^{M+1}, ..., H_K^{M+1}. The outputs are concatenated corresponding to the axis of the representation column. Let $q_i^k \in \mathbb{R}^{s_{(M+1)}}$ be a latent representation of node i from H_k^{M+1}. Thus, $H_k^{M+1}[i,:] = q_i^k$. After the concatenation, q_i^{con} is $q_i^{con} = \Big\Vert_{k=1}^{K} q_i^k$.

Attention Layer The self-attention layer measures the degree of *role equivalence* (Def. 3) among nodes to place more importance on structurally similar neighbors. The intermediate representations of the last high-order GCN layers are used for defining the role, thus $f(i) := q_i^{con}$. In this layer, by considering role equivalence, we can incorporate structural information into node classification. To measure the degree of role equivalence, we additionally define a quantitative measure of role equivalence in latent space with $\mathcal{RE}\big(f(i), f(j)\big)$, and use it in the attention layer below. Our self-attention layer takes inputs from the concatenation layer and produces a new vector $q'_i \in \mathbb{R}^{K \cdot s_{(M+1)}}$ as:

$$q'_i = \text{ReLU}\left(\sum\nolimits_{j \in \mathcal{N}(i)} \mathcal{RE}\big(q_i^{con}, q_j^{con}\big) \cdot q_j^{con}\right) \tag{2}$$

where $\mathcal{RE}\big(f(i), f(j)\big) = (1/Z)e^{\beta cos\left(f(i), f(j)\right)}$, cos refers to cosine similarity, $Z = \sum_{j \in \mathcal{N}(i)} e^{\beta cos\left(f(i), f(j)\right)}$, and β is a hyperparameter that moderates attention, which we estimate during learning. Note that we do not consider self-loops when computing similarity. $\mathcal{RE}\big(f(i), f(j)\big)$ measures how close nodes i and j are to

being role equivalent (i.e., if the latent representations are unit vectors, then the two are equal when their cosine similarity is 1).

Final Softmax Layer To predict node class labels, a final softmax function is used. Here, $\hat{y}_i$ is the output of the softmax function, and each dimension of $\hat{y}$ represents the predicted probability of the corresponding labels for the class given inputs. Note that $W_{\text{final}} \in \mathbb{R}^{(K \cdot s_{(M+1)}) \times C}$. For learning, as in the original GCN, we use a categorical cross-entropy loss.

$$\mathcal{L}_{\text{batch}}(L, Y) = -\sum_{V_L} \sum_{j=0}^{C-1} y_j log(\hat{y}_j), \text{where } \hat{y}_i = \text{softmax}\left(W_{\text{final}} {q'}_i + b_{\text{final}}\right) \quad (3)$$

In Eq. (3), C is the number of class labels. Since all activation functions are differentiable, learning is simple via back-propagation—all Ws (W_k^m and W_{final}), b_{final}, and β are trained. $\hat{y}_i$ is used to predict class labels for unlabeled nodes V_U.

4.3 Importance Sampling for Scalability

Calculating Eq. (3), requires the loss to be summed over all the nodes labeled together. A batch algorithm cannot handle large-scale datasets due to the difficulty of fitting the full graph in GPU memory and slow convergence. Even worse, the dense neighbors from high-order paths reduce the scalability of the model with respect to both time and space. To overcome the limitation, we propose an efficient sampling-based learning method. Consider the representation of a node u in the m^{th} GCN layer with k^{th} order paths from Eq. (1):

$$(\hat{A}^k H_k^{(m)})_u = |V| \sum\nolimits_{v=1}^{|V|} \frac{1}{|V|} \hat{A}^k[u, v] H_k^{(m)}[v, :] \quad (4)$$

We use the same importance distribution as in [3] to approximate Eq.(4) with $|S|$ samples for node u as follows:

$$(\hat{A}^k H_k^{(m)})_u \approx \frac{|V|}{|S|} \sum\nolimits_{v=1}^{|S|} \frac{1}{q(v)} \tilde{A}^k[u, v] H_k^{(m)}[v, :] \quad (5)$$

with the importance distribution $q(v) = ||\hat{A}[:, v]||^2 / \sum_{v' \in V} ||\hat{A}[:, v']||^2$. Note the distribution q is only calculated once (i.e., before training) given the normalized aggregated graph, and the input label matrix, $H_k^{(1)}$, should be updated according to S via $H_k^{(1)} = H_k^{(1)}[S, :]$. Our overall mini-batched training procedure is described in Algorithm 1. At every epoch, all nodes are randomly divided to create a mini-batch set B, which is composed of multiples of γ nodes. We set $\gamma = 1024$. B provides a candidate node set for sampling $|S|$ later. When it comes to a new *mini-batch*, $\tilde{A}^k$ is induced from $\hat{A}^k$ according to S. Similarly, when REGNN gets neighboring nodes at the attention layer for $\mathcal{N}(i)$ in Eq.(2), $\tilde{A}^{k=1}$ is used. As a result, the new loss function for the mini-based training will be $\mathcal{L}_{\text{mini-batch}}(L, Y) = -\sum_{S_L} \sum_{j=0}^{C-1} y_j log(\hat{y}_j)$. S_L is defined from $\{v \in S \cap V_L\}$.

4.4 Relationship to Label Propagation (LP)

Assume that $Y_L \in \mathbb{R}^{|V| \times C}$ is a label input matrix. For each labeled node $i \in V_L$ with class label $y_i = c$, we set $Y_L[i, c] = 1$ and $Y_L[i, .] = 0$ otherwise. If node j is

Algorithm 1 REGNN's mini-batched training (one epoch)

Generate a mini-batch set B from V
for each *mini-batch* $\in B$ **do**
  Sample $|S|$ vertices, $v_1, ..., v_{|S|}$, according to distribution q from *mini-batch*
  Assign $H_k^{(1)} = H_k^{(1)}[S, :]$
  For $k = [1, K]$, assign $\tilde{A}^k = \hat{A}^k[S, S]$
  Compute the categorical cross-entropy in $\mathcal{L}_{\text{mini-batch}}(L, Y)$
  Update W_k^m, β, W_{final}, and b_{final}
end for

in V_U, we set $Y_L[j, :] = 0$. Let $\widehat{\mathbf{Y}}$ be a prediction matrix, and $\widehat{\mathbf{Y}}[i, :]$ for each node $i \in V_U$ will be used for actual prediction. The prediction is from $\arg\max_j \widehat{\mathbf{Y}}[i, j]$. According to [22], the prediction will converge as $\widehat{\mathbf{Y}}_{LP} = (I - \alpha(I - L))^{-1} Y_L$. α is a parameter in (0, 1) and specifies the relative amount of the information from its neighbors and the initial label information. Regarding an input graph, The normalized Laplacian matrix L of A is decomposed as $L = \Phi \Lambda \Phi^{-1}$ and could be modified using the frequency response [14] as $L' = \Phi p(\Lambda) \Phi^{-1}$ where $p(\cdot)$ is called the frequency response function of the graph. $p(\Lambda)$ can further be written as $diag(p(\lambda_1), ..., p(\lambda_n))$. The graph L' is linear shift-invariant, if and only if there exist a function $p(\cdot) : \mathbb{R} \to \mathbb{R}$. At last, the prediction of LP [22], $\widehat{\mathbf{Y}}_{LP}$, can be reformulated from the perspective of eigen-decomposition and is shown as:

$$\begin{aligned} \widehat{\mathbf{Y}}_{LP} &= (I - \alpha(I - L))^{-1} Y_L = ((1-\alpha)I + \alpha L)^{-1} Y_L \\ &= \Phi((1-\alpha)I + \alpha\Lambda)^{-1} \Phi^{-1} Y_L = F(p_{\text{LP}}(\Lambda), Y_L) \end{aligned} \tag{6}$$

In this $\widehat{\mathbf{Y}}_{LP}$, $p_{\text{LP}}(\lambda_i)$, the frequency response function of LP, is equal to $\frac{1}{(1-\alpha)+\alpha\lambda_i}$.

Similarly, we can reformulate the GCN. Denote $\tilde{D}_{ii} = \sum_j (A + I)_{ij}$. Then, $\hat{A} = \tilde{D}^{-1/2} (A + I) \tilde{D}^{-1/2} = I - \tilde{D}^{-1/2} \tilde{L} \tilde{D}^{-1/2} = I - \hat{L}_s$, where $\tilde{L}$ is the Laplacian matrix of $A+I$. We denote $\hat{L}_s = \Phi \hat{\Lambda} \Phi^{-1}$. Using the above notation, a two-layered GCN can be characterized as follows, where W^0 and W^1 are trainable weight matrices in the first and second GCN layers, respectively:

$$\begin{aligned} \widehat{\mathbf{Y}}_{GCN} &= \hat{A} \text{ ReLU } (\hat{A} Y_L W^0) W^1 \\ &\approx \hat{A}(\hat{A} Y_L W^0) W^1 = \hat{A}^2 (Y_L W^0) W^1 = (I - \hat{L}_s)^2 (Y_L W^0) W^1 \\ &= (\Phi(I - \hat{\Lambda})\Phi^{-1})^2 (Y_L W^0) W^1 = \Phi(I - \hat{\Lambda})^2 \Phi^{-1} Y_L (W^0 W^1) \\ &= F(p_{\text{GCN}}(\hat{\Lambda}), Y_L)(W^0 W^1), \end{aligned} \tag{7}$$

where $p_{\text{GCN}}(\hat{\lambda}_i) = (1 - \hat{\lambda}_i)^2$. When LP [22] uses the following frequency response function, $p(\lambda_i) = (1 - \lambda_i)^2$, with two linear transformations, the new $\widehat{\mathbf{Y}}'_{LP}$ will be same as $\widehat{\mathbf{Y}}_{GCN}$. Thus, $\widehat{\mathbf{Y}}_{GCN} \approx \widehat{\mathbf{Y}}'_{LP}(W^0 W^1)$. In other words, GCN can approximate LP, and as such we expect better accuracy in label propagation with the help of additional linear transformations and a non-linear function.

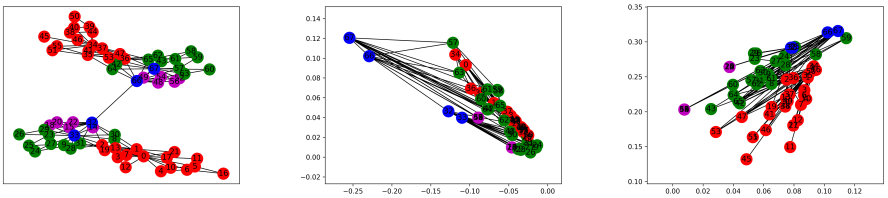
When the k^{th} order adjacency matrix A^k is fed to the GCN, the response function becomes $(1-\lambda_i)^{2k}$, which means that we can estimate labels from the different eigenvalue function by considering different paths. This analysis implies that high-order GCN layers in our REGNN can get label-wise representations of unknown nodes in latent space using different eigenvalue functions. Furthermore, they can learn a joint representation using our proposed *role equivalence* attention layer.

4.5 Complexity Analysis

When batch size is considered, the time complexity of learning REGNN (before importance sampling) is $O(|E^1| + ... + |E^K|) \approx O(|E^K|)$ due to the edge-wise aggregations. We assume that the sizes of hidden nodes in REGNN are constants and $|E^K| \geq |E^1|$. Each $|E^k|$ denotes the numbers of edges from the k^{th} order matrix $\hat{A}^k$. After we apply importance sampling, the new time complexity is $O(|E_S^K|)$, where each E_S^K is the set of edges among the sampled $|S|$ vertices in $\hat{A}^K$. Thus, $|E_S^K| \ll |E^K|$. Regarding space complexity, we note that all high-order matrices are preprocessed and do not need to be stored in the main memory. Then $(|S| \times |S|)$ matrices are indexed based on the selected sample S. Therefore, the space complexity of (online) learning depends on at most $O(|S|^2)$, where $|S|$ is typically chosen from [32, 64, 128, 256].

5 Experimental Evaluation

Data and Experimental Setup We use six real-world network datasets for evaluation. Cora, Citeseer, PubMed, and NELL are publicly available citation network datasets [20]. Facebook ($|V|=4038, |E|=65794, C=2$) is drawn from the Purdue University network [12], and the data were randomly sampled to make its class labels' proportion to 50/50. For Friendster [16] ($|V|=43880, |E|=145407, C=4$), only 30% of the training data is used to make the label ratio equal. Additionally, we generated a mirrored-Karate network to verify how roles are learned in our model. LR (Logistic Regression), LP [22], GhostEdge [7], graph neural networks (GCN, N-GCN [1], and GAT [18]), and graph embedding methods (Node2Vec [8], GraRep [2], NEU [19], Struc2Vec [13], and VERSE [17]) are used for comparison. We train the models on training/validation sets and report results on the test set. Every reported result is the average of 10 trials using randomly shuffled node sets, and 10% of the nodes are used for testing and validation, respectively. The number of nodes for training is varied. For all neural network models, we set max epochs=1000, dropout_rate=0.2, learn_rate = 0.01, and optimizer=*adam*. For GCN-based models, the size of hidden nodes ($s_{(m)}$ if $m>1$ for W_k^m) in GCNs is searched over [C, 8, 16, 32, 64], and all layers have the same size of hidden nodes. REGNN considers $K = [2, ..., 5]$ and selects the best using validation loss. For importance sampling, $|S|$ is chosen from [32, 64, 128, 256, 512]. For all embedding models, the size of representation was searched in [32, 64, 128], and other parameters are set to their defaults. We use GhostEdgeNL and N-GCN$_{fc}$. All experiments were executed on an Intel Xeon Gold 6126 CPU@2.60GHz server with 192GB RAM.



(a) Input (b) GCN (c) REGNN (Our Model)

Fig. 2: Visualization of node representations from the mirrored Karate network.

5.1 Results: Synthetic Data

The Karate club network [21] is a graph that is composed of 34 members and 78 interactions among them. To interpret REGNN with respect to capturing roles, we construct a *mirrored* network, which is composed of two copies of the network connected between node 32 and 66, as in Fig. 2a. We can assume that each node has its own structural role, which connects between different communities. The colors in the graph are chosen according to community IDs after community detection [4]. Every node that has the same color (i.e., role equivalent) should have similar latent representation when their structural roles are properly captured. Fig. 2b and Fig. 2c show the learned representations of nodes from GCN and REGNN, respectively. Similar to the GCN experiment with the Karate network [10], a hidden layer of size 2 was inserted before the final softmax layer for visualization of the latent representations. They are visualized in Fig. 2b and Fig. 2c. Labels for training data were chosen from total 8 nodes (two nodes per a community). In the result, GCN fails to distinguish red and green nodes (i.e., the communities overlap), while REGNN separated the nodes from the two communities more effectively. This is evidence that that REGNN's attention layer successfully learned the structural roles by measuring role equivalence.

5.2 Results: Real-world Data

Tables 1-5 show REGNN node classification performance on the Citeseer, Cora, Facebook, PubMed, and Friendster data as proportion of labeled nodes is varied, compared to other baselines. For GCN, Node2Vec, GraRep, Struct2Vec, VERSE, and NEU, we directly obtain results from official implementations. Classification results of all methods are averaged at each proportion. Bold scores represent the corresponding model is significantly better than the others by paired t-tests (p-value < 0.05). In all datasets, REGNN has consistently good results across all label proportions. On the other hand, N-GCN is similar to GCN and LP in Citeseer and Cora, in particular. This indicates that the high-order path information did not help to find better node representations, but the attention over high-order paths helped to increase performance when only known labels are given. Node2Vec and GhostEdge exhibit similar results in most of the datasets, and both achieve good performance at lower label proportions. However, their relative performance often decreases when more labels are available (e.g., in Cora). Struct2Vec and VERSE are not as good as Node2Vec. Since Struct2Vec considers structural similarity only, it does not perform well on most of the

Table 1: Accuracy (%) on Citeseer

% Labeled	10	20	30	40	50
REGNN	**55.03**	**59.05**	**63.18**	**66.84**	**68.74**
N-GCN	51.92	56.84	60.62	64.23	66.07
GCN	51.47	57.40	61.75	65.03	67.36
LP	53.80	57.78	61.37	63.98	66.33
NEU	49.29	55.26	57.54	59.05	59.98
GraRep	50.08	51.97	52.59	52.87	53.48
VERSE	36.28	39.63	40.30	40.66	40.63
Struct2Vec	36.65	39.67	41.97	43.35	43.54
Node2Vec	52.64	54.50	56.05	56.87	57.49
GEdgeNL	50.12	53.94	56.26	58.39	59.49

Table 2: Accuracy (%) on Cora

% Labeled	10	20	30	40	50
REGNN	**76.04**	**80.04**	**82.37**	**84.19**	**85.45**
N-GCN	72.31	78.16	80.91	81.55	84.33
GCN	71.75	77.60	80.93	82.80	84.89
LP	73.55	77.91	80.32	82.81	84.31
NEU	72.28	76.16	79.72	81.55	83.15
GraRep	72.85	74.71	75.02	75.16	75.26
VERSE	57.02	61.53	63.44	63.82	64.32
Struct2Vec	53.81	58.34	61.24	63.38	63.95
Node2Vec	**76.44**	77.88	79.24	80.20	80.04
GEdgeNL	72.22	75.16	77.19	78.79	79.39

Table 3: Accuracy (%) on Facebook

% Labeled	10	20	30	40	50
REGNN	**59.85**	**60.75**	**61.53**	**61.39**	**62.05**
N-GCN	58.27	59.87	60.31	60.49	61.62
GCN	55.72	56.47	59.06	59.17	59.87
LP	56.25	57.36	58.45	59.54	59.83
NEU	56.29	58.52	59.94	60.23	60.88
GraRep	57.48	58.09	59.73	59.50	59.71
VERSE	53.94	56.67	57.09	56.89	57.40
Struct2Vec	53.32	54.47	54.75	54.86	53.56
Node2Vec	57.20	58.07	59.95	59.70	60.36
GEdgeNL	56.28	57.54	58.99	59.61	59.83

Table 4: Accuracy (%) on Pubmed

% Labeled	10	20	40	60	80
REGNN	**79.95**	**82.00**	**83.21**	**83.30**	**84.10**
N-GCN	78.24	81.04	81.41	82.72	83.23
GCN	77.94	80.73	**83.33**	**83.77**	**84.36**
LP	78.97	80.62	82.12	82.75	83.28
NEU	75.59	76.71	77.52	77.94	77.86
GraRep	79.14	79.68	79.90	80.04	80.07
VERSE	**80.44**	81.01	81.15	81.29	81.18
Node2Vec	79.42	80.28	80.86	80.82	81.03

Table 5: Accuracy (%) on Friendster

% Labeled	10	15	20	25	30
REGNN	**34.62**	**36.18**	**36.7**	**36.93**	**37.02**
N-GCN	28.93	28.54	32.03	32.55	31.89
GCN	29.7	29.74	30.2	30.18	31.82
LP	27.13	26.32	25.74	24.43	24.43
NEU	30.28	30.75	31.09	31.13	31.4
GraRep	33.53	33.93	34.22	34.53	34.72
VERSE	32.41	33.33	34.01	33.9	34.32
Node2Vec	31.81	32.58	32.8	33.27	33.36

Table 6: Accuracy (%) on NELL

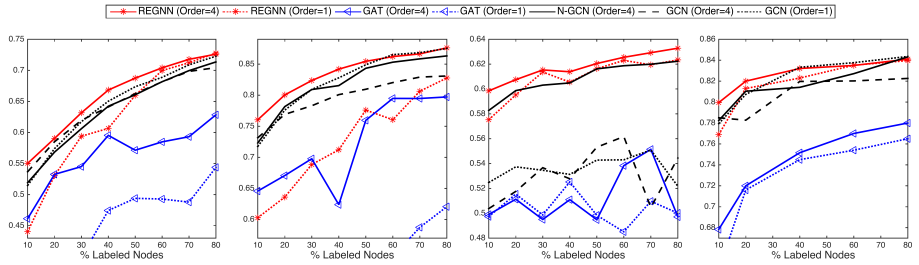
Method	Accuracy	Execution Time (Secs)
REGNN	**85.6**	740.45
N-GCN with IS	84.22	682.23
GCN	79.56	523.27
LP	82.67	1445.41
NEU	81.25	2787.72 (training only)
GraRep	79.25	2339.96
VERSE	**85.43**	1908.54
Node2Vec	84.41	2501.8 (training only)

datasets. VERSE also learns similarities from Personalized PageRank, which is not helpful for our citation and social network datasets. For PubMed and Friendster, due to the heavy computation cost on the large edges, Struct2Vec, and GhostEdgeNL are not included.

Table 6 shows classification performance on the NELL knowledge graph. The result is from the same train/test/validation sets as in [20]. REGNN shows the best performance but is almost on par with Node2Vec and VERSE. However, the execution time for training was much faster than Node2Vec and VERSE. In particular, Node2Vec and NEU incur a great deal of overhead to generate random walks (4,327.43 seconds), and their training time to learn embeddings after the generation was also slower than REGNN. We also tested N-GCN with our importance sampling but the accuracy was still lower than REGNN.

5.3 Effect of Attention Mechanism

REGNN uses role-based attention to leverage high-order paths. In this section, we report how high-order paths or role-based attention contributes to increasing REGNN's performance. Fig. 3 shows comparisons from an ablation study. We compare REGNN (Order=4), which is the best performing order chosen during



(b) Citeseer (c) Cora (d) Facebook (e) Pubmed

Fig. 3: Effect of attention mechanism (Y-axis: Accuracy). Best viewed in color.

parameter selection on the validation data, to REGNN (Order=1), which denotes a simplified REGNN that still use the role-based attention but does not consider high-order paths. N-GCN and GAT (Order=4) correspond to versions of our model where the *role equivalence* attention is replaced by the mixing layer used in [1] and the edge-wise attention of [18], respectively. For the mixing layer, column-wise concatenation is used. We also compared with the softmax attention of [1] in our experimental setting, but the concatenation-based mixing layer was more accurate.

In the ablation experiments, REGNN (Order=4) again achieved the best results across all datasets. Specifically, it performed significantly better (assessed by paired t-tests) than REGNN (Order=1), GAT (Order=4), and GAT (Order=1). In this ablation study, before computing the attentive weights, high-order GCNs are used in the same way for the GAT for fair comparison, but the result is still worse than REGNN (Order=4). We tested different numbers of multi-headed attentions for GAT, but it did not help much. This means that our attention mechanism can identify more meaningful neighbors than the one used in GAT—at least in our application settings, which focus on label propagation in graphs with few attributes. In addition, when high-order GCNs are not used, REGNN (Order=1) is worse than the simple GCN (Order=1) in Citeseer and Cora. This indicates that it is more effective when REGNN combines its latent representations with high-order paths.

6 Conclusions

In this paper, we propose REGNN, a Graph Neural Network architecture that uses a novel *Role Equivalence* attention mechanism with higher-order paths. REGNN is able to exploit nodes roles to learn how to focus on relevant neighbors from high-order paths, in order to adaptively reduce the increased noise that occurs when higher-order structures are considered. In our experimental results, REGNN showed significant performance gains compared to state-of-the-art alternatives that use alternative attention mechanisms and/or higher-order paths.

Acknowledgments
This research is supported by NSF and AFRL under contract numbers: CCF-1918483, IIS-1618690, and FA8650-18-2-7879.

References

1. Abu-El-Haija, S., Kapoor, A., Perozzi, B., Lee, J.: N-gcn: Multi-scale graph convolution for semi-supervised node classification. In: Proc. of UAI (2019)
2. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proc. of CIKM (2015)
3. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. In: Proc. of ICLR (2018)
4. Clauset, A., Newman, M.E., Moore, C.: Finding community structure in very large networks. Physical review E **70**(6), 066111 (2004)
5. Everett, M.G., Borgatti, S.P.: Regular equivalence: General theory. Journal of Math. Soc. **19**(1), 29–52 (1994)
6. Everett, M.G., Boyd, J.P., Borgatti, S.P.: Ego-centered and local roles: A graph theoretic approach. Journal of Math. Soc. (1990)
7. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proc. of SIGKDD. pp. 256–264. ACM (2008)
8. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proc. of SIGKDD (2016)
9. Hoshen, Y.: Vain: Attentional multi-agent predictive modeling. In: Proc. of NeurIPS. pp. 2701–2711 (2017)
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proc. of ICLR (2017)
11. Lorrain, F., White, H.C.: Structural equivalence of individuals in social networks. The Journal of Math. Soc. **1**(1), 49–80 (1971)
12. Pfeiffer, III, J.J., Neville, J., Bennett, P.N.: Overcoming relational learning biases to accurately predict preferences in large scale networks. In: Proc. of WWW (2015)
13. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: Learning node representations from structural identity. In: Proc. of SIGKDD (2017)
14. Sandryhaila, A., Moura, J.M.: Discrete signal processing on graphs. IEEE trans. on signal processing **61**(7), 1644–1656 (2013)
15. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI mag. (2008)
16. Teixeira, L., Jalaian, B., Ribeiro, B.: Are graph neural networks miscalibrated? In: Proc. of Learning and Reasoning with Graph-Structured Representations Workshop in ICML 2019 (2019)
17. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proc. of WWW (2018)
18. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: Proc. of ICLR (2018)
19. Yang, C., Sun, M., Liu, Z., Tu, C.: Fast network embedding enhancement via high order proximity approximation. In: Proc. of IJCAI (2017)
20. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proc. of ICML (2016)
21. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of anthropological research **33**(4), 452–473 (1977)
22. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Proc. of NIPS (2004)
23. Zhou, Z., Li, X.: Graph convolution: A high-order and adaptive approach. In: Proc. of AAAI (2017)