Cross-Network Learning with Partially Aligned Graph Convolutional Networks

Meng Jiang
Department of Computer Science and Engineering, University of Notre Dame
Notre Dame, Indiana, USA
mjiang2@nd.edu

ABSTRACT

Graph neural networks have been widely used for learning representations of *nodes* for many downstream tasks on graph data. Existing models were designed for the nodes on a single graph, which would not be able to utilize information across multiple graphs. The real world does have multiple graphs where the nodes are often *partially aligned*. For examples, knowledge graphs share a number of named entities though they may have different relation schema; collaboration networks on publications and awarded projects share some researcher nodes who are authors and investigators, respectively; people use multiple web services, shopping, tweeting, rating movies, and some may register the same email account across the platforms. In this paper, I propose partially aligned graph convolutional networks to learn node representations across the models. I investigate multiple methods (including model sharing, regularization, and alignment reconstruction) as well as theoretical analysis to *positively* transfer knowledge across the (small) set of partially aligned nodes. Extensive experiments on real-world knowledge graphs and collaboration networks show the superior performance of our proposed methods on relation classification and link prediction.

CCS CONCEPTS

• **Theory of computation** → **Graph algorithms analysis**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Transfer learning; Graph neural network; Representation learning

ACM Reference Format:
Meng Jiang. 2021. Cross-Network Learning with Partially Aligned Graph Convolutional Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467282

1 INTRODUCTION

Graph learning represents or encodes graph structure to perform tasks such as classification, clustering, and regression on graphs. It is important and ubiquitous with applications ranging from knowledge graph completion to movie recommendation. It is originally studied as dimensionality reduction in adjacency matrix factorization methods [2]. Then network embedding methods aim at preserving particular structure, property, or side information in graph or networked data [4, 5]. Recently, graph convolutional networks (GCNs) have been introduced to specify model parameters (e.g., a set of weight matrices) on how to aggregate information from a node's local neighborhood [12, 20]. Unlike the network embedding methods, these parameters are shared across nodes and allow the approaches to generate node representations that do not necessarily rely on the entire graph [11, 33, 38].

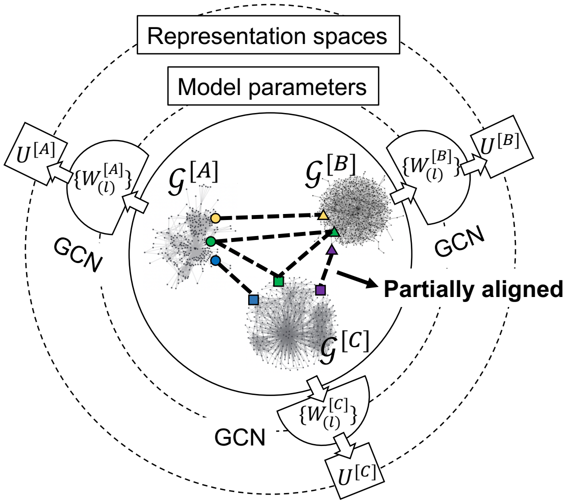


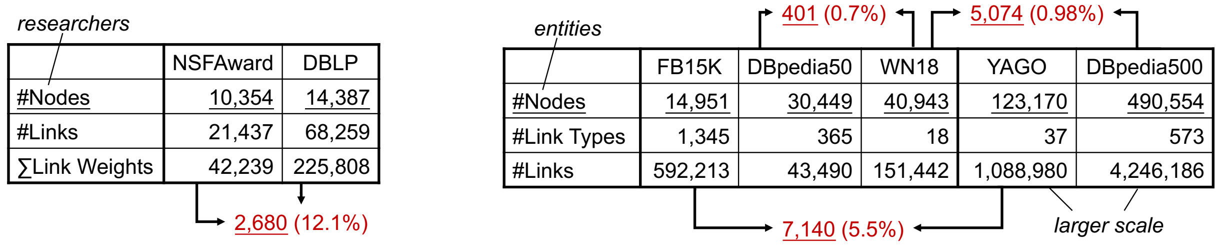
Figure 1: Cross-GCN learning uses network alignment information to bridge the separated model parameters and representation spaces and transfer knowledge across the models.

The sparsity problem in graph data is a major bottleneck for most graph learning methods. For example, knowledge graphs are incomplete, causing inaccuracy in downstream tasks such as fact retrieval and question answering, especially when knowledge is represented under the Open World Assumption [31]. In real-world recommender systems, users can rate a very limited number of items, so the user-item bipartite graph is always extremely sparse [8, 46]. The observed data that can be used for adjacency matrix factorization or neural net graph learning are radially insufficient.

Although we cannot fabricate more observations, we may borrow useful knowledge from graph data on different sources, domains, or platforms. Consider the following case: A new UberPool-like carpooling App has launched. Due to lack of usage at the beginning, machine learning on the very sparse user-user ride-sharing graph

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467282



	NSFAward	DBLP
#Nodes	10,354	14,387
#Links	21,437	68,259
∑Link Weights	42,239	225,808

researchers
2,680 (12.1%)

(a) Partially aligned **collaboration networks** for **link prediction**

	FB15K	DBpedia50	WN18	YAGO	DBpedia500
#Nodes	14,951	30,449	40,943	123,170	490,554
#Link Types	1,345	365	18	37	573
#Links	592,213	43,490	151,442	1,088,980	4,246,186

entities
401 (0.7%)
5,074 (0.98%)
7,140 (5.5%)
larger scale

(b) Partially aligned **Knowledge Graphs** for **relation classification**

Figure 2: Examples of partially aligned networks in real world for two different tasks.

cannot be effective for ride recommendation. Now suppose that a small number of the users register the App with their social media accounts, and we have social graph data available from the social media platform. Then we ask, though the percentage of the "aligned" users is very small (10%, 5%, or even less than 1%), can we use the alignment as a bridge and transfer knowledge from the social graph to the ride-sharing graph? Since ride sharing and social relationship are somewhat related, the *partially* aligned users on different graphs (platforms) can share similar behavioral patterns. Thus, learning across the graphs can be beneficial. When the graph learning models are state-of-the-art, learning across graph convolutional networks can transfer knowledge via the partially aligned nodes.

Figure 1 illustrates the idea of cross-GCN learning. Originally, the three GCN models are trained *separately* to generate low-dimensional representations of nodes $U^{[\cdot]}$ with a set of model parameters $\{W^{[\cdot]}_{(l)}\}$. We seek for effective methods to transfer knowledge across the GCN models through the partial alignment information between $\mathcal{G}^{[\cdot]}$ into their node representations.

In this paper, I study how to alleviate the sparsity problem in graph learning by transferring knowledge across multiple GCN models, when the graphs are partially aligned.

The contributions of this work can be described in three aspects:

(1) **Cross-GCN learning methods:** I design three different types of methods: (a) sharing model parameters, (b) aligning representation spaces with regularizations, and (c) simultaneously reconstructing the alignment and learning representations. I present how to generalize them for more than two GCNs and for relational graphs.

(2) **Theoretical foundations:** I present theories on (a) the choice of the number of dimensions of the representation spaces to force knowledge transfer and (b) the conditions of positive transfer across GCN models, regularizations, and alignment reconstruction.

(3) **Extensive experiments:** As shown in Figure 2, I perform experiments on two types of graphs datasets and graph learning tasks: (a) link prediction on weighted graphs (e.g., collaboration social networks) and (b) relation classification on knowledge graphs. Both graphs are partially aligned (see the number of aligned nodes in red). Cross-GCN learning performs consistently better than individual GCNs on all the datasets and tasks.

The rest of this paper is organized as follows. Section 2 gives definitions of basic concepts (e.g., graph convolutional network) and the proposed goal. Section 3 presents multiple kinds of methods to achieve the goal of cross-network learning. Section 4 provides theoretical analysis on choice of number of dimensions and positive transfer. Section 5 presents experimental results on knowledge graphs and collaboration networks for relation classification and link prediction, respectively. Section 6 reviews the related papers, and Section 7 concludes the work.

2 PRELIMINARIES

2.1 Graphs and Learning Tasks

We study on two types of graphs and related tasks about learning, classifying, or predicting links between nodes.

Definition 1 (Weighted graph). *A weighted graph, denoted as $\mathcal{G} = (\mathcal{U}, \mathcal{E} \subset \mathcal{U} \times \mathcal{U})$, contains a set of nodes $\mathcal{U}$ and a set of links $\mathcal{E}$, with a weight mapping function $w : \mathcal{E} \rightarrow \mathbb{R}$.*

For example, collaboration networks are weighted graphs. The weight is the frequency of collaborations between two person nodes. So, it is a positive integer when the link exists. The task of *link prediction* is to predict whether the value of $w(u_i, u_j)$ is positive or what the value is, given a pair of nodes $u_i, u_j \in \mathcal{U}$.

Definition 2 (Relational graph). *A relational graph, denoted as $\mathcal{G}_{rel} = (\mathcal{U}, \mathcal{E} \subset \mathcal{U} \times \mathcal{U} \times \mathcal{R}, \mathcal{R})$, contains a set of nodes $\mathcal{U}$, a set of relational links $\mathcal{E}$, and a relation schema $\mathcal{R}$. A relation link $e = (u_i, u_j, r) \in \mathcal{E}$ indicates that u_i and u_j have relation r.*

Knowledge graphs are typical relational graphs, where the relational links need to be learned and completed. The task of *relation classification* is to predict the type of relation between two nodes (i.e., entities) u_i, u_j from the relation schema $\mathcal{R}$ so that a link $e = (u_i, u_j, r)$ is likely to exist in the graph.

The fundamental problem of both tasks is *node representation learning* that learns low-dimensional vectors of nodes to preserve the graph's information. Take *link prediction* as an example – formally, it is to learn a mapping function: $\vec{u} = f(u, \mathcal{G}, \Theta) \in \mathbb{R}^d$, where Θ is the node representation learning model's parameters and d is the number of dimensions of node representations. Then the value of $w(u_i, u_j)$ is assigned as $\sigma(\vec{u_i}^\top \vec{u_j})$, where σ is sigmoid. Simple predictive functions can be specified for the other two tasks.

2.2 Graph Convolutional Network

Graph convolutional network (GCN) is one of the most popular models for node representation learning [20]. Let us continue using *link prediction* as the example. We denote the number of nodes in graph $\mathcal{G}$ by $n = |\mathcal{U}|$. Suppose $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $\mathcal{G}$ and $X \in \mathbb{R}^{n \times m}$ has the nodes' raw attribute values, where m is the number of the raw attributes. A two-layer GCN generates the matrix of node representations as follows:

$$U = \text{softmax}\left(\hat{A}\ \text{ReLU}(\hat{A}\ X\ W_{(1)})\ W_{(2)}\right) \in \mathbb{R}^{n \times d}, \qquad (1)$$

where $\Theta = \{W_{(1)} \in \mathbb{R}^{m \times d}, W_{(2)} \in \mathbb{R}^{d \times d}\}$ has layer-specific weight matrices. We choose two layers as most of existing studies suggest. The pre-processing step includes (1) $\tilde{A} = A + I_n$, (2) $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and (3) $\hat{A} = \tilde{D}^{\frac{1}{2}}\ \tilde{A}\ \tilde{D}^{\frac{1}{2}}$. Graph reconstruction is used to inform the GCN to learn Θ to generate effective U:

$$g(X, A, W_{(1)}, W_{(2)}) = \sigma(UU^{\top}) \in [0, 1]^{n \times n}. \qquad (2)$$

The loss function can be written as follows:

$$\begin{aligned} f(W_{(1)}, W_{(2)}) &= \mathcal{L}\left(g(X, A, W_{(1)}, W_{(2)}),\ A\right) \\ &= -\sum_{i,j \in \{1 \dots n\}} A_{ij} \log\left(\sigma(\vec{u_i}^{\top} \vec{u_j})\right) \qquad (3) \end{aligned}$$

where $\vec{u_i}$ and $\vec{u_j}$ are the i-th and j-th rows in matrix U. It is worth noting that one can easily extend our study from two-layer GCN to l-layer GCN ($l > 2$).

2.3 Cross-Network Learning

One GCN can be built for one graph. What if we have multiple graphs which overlap each other to perform the same type of tasks? The overlapping parts, referred as *network alignment* serve as bridge for transferring knowledge across multiple GCNs so that they can mutually enhance each other. Without loss of generality, we focus on the cases that have two graphs $\mathcal{G}^{[A]}$ and $\mathcal{G}^{[B]}$. We denote by $A^{[A,B]} \in \{0, 1\}^{n_A \times n_B}$ the adjacency matrix of network alignment between the two graphs: $A^{[A,B]}_{ij} = 1$, if node $u^{[A]}_i$ and node $u^{[B]}_j$ are aligned; where $n_A = |\mathcal{U}^{[A]}|$ and $n_B = |\mathcal{U}^{[B]}|$. Usually the graphs are *partially aligned*, leading to missing alignment links and the sparsity of the alignment matrix.

Definition 3 (Separated GCNs). *Traditionally, the network alignment was ignored in building GCNs. So, two separated GCNs would be built for the tasks on graphs $\mathcal{G}^{[A]}$ and $\mathcal{G}^{[B]}$:*

$$\begin{aligned} f(W^{[A]}_{(1)}, W^{[A]}_{(2)}) &= \mathcal{L}\left(g(X^{[A]}, A^{[A]}, W^{[A]}_{(1)}, W^{[A]}_{(2)}),\ A^{[A]}\right), \\ f(W^{[B]}_{(1)}, W^{[B]}_{(2)}) &= \mathcal{L}\left(g(X^{[B]}, A^{[B]}, W^{[B]}_{(1)}, W^{[B]}_{(2)}),\ A^{[B]}\right), \qquad (4) \end{aligned}$$

where no model parameters (between $\Theta^{[A]}$ and $\Theta^{[B]}$) is shared and no alignment information is used.

Definition 4 (Cross-network learning). *Given two graphs $\mathcal{G}^{[A]}$ and $\mathcal{G}^{[B]}$ as well as the network alignment information $A^{[A,B]}$, build cross-network learning models that have the properties below:*

- *generate node representations $U^{[A]} \in \mathbb{R}^{n_A \times d_A}$ and $U^{[B]} \in \mathbb{R}^{n_B \times d_B}$ through graph convolutions, where d_A and d_B are the numbers of dimensions, respectively – $d_A \ll m_A$, $d_B \ll m_B$, and they do not have to be equivalent;*
- *perform better than separated GCNs by sharing model parameters and/or incorporating the network alignment information.*

3 PROPOSED METHODS

In this section, we present multiple types of methods for transferring knowledge across GCNs. The first type trains two GCNs by sharing their model parameters (Sections 3.1 and 3.2). The second uses the network alignment information to regularize the output representation spaces (Section 3.3). And the third type uses network alignment reconstruction as an extra task for knowledge transfer (Section 3.4). Generalizations to multiple GCNs and particular types of graphs are discussed (Sections 3.5 to 3.6).

3.1 Sharing Model Parameters across GCNs

Recall that a two-layer GCN on graph $\mathcal{G}^{[A]}$ has parameters $\Theta^{[A]} = \{W^{[A]}_1 \in \mathbb{R}^{m_A \times d_A}, W^{[A]}_2 \in \mathbb{R}^{d_A \times d_A}\}$. Same goes for graph $\mathcal{G}^{[B]}$.

Suppose $d_A \neq d_B$. None of the parameters can be directly shared.

Suppose $d_A = d_B$. The second-layer weight parameters can be shared: $W_2 = W^{[A]}_2 = W^{[B]}_2$. So, we can perform joint GCN training:

$$\begin{aligned} f(W^{[A]}_{(1)}, W^{[B]}_{(1)}; W_{(2)}) = {} & \alpha^{[A]} \cdot \mathcal{L}\left(g(X^{[A]}, A^{[A]}, W^{[A]}_{(1)}, W_{(2)}),\ A^{[A]}\right) \\ & + \alpha^{[B]} \cdot \mathcal{L}\left(g(X^{[B]}, A^{[B]}, W^{[B]}_{(1)}, W_{(2)}),\ A^{[B]}\right), \qquad (5) \end{aligned}$$

where $\alpha^{[A]}$ and $\alpha^{[B]}$ are per-graph weight of the training procedure. When we have two graphs, $\alpha^{[A]} + \alpha^{[B]} = 1$.

Suppose $m_A = m_B$ and $d_A = d_B$. The weight parameters of both layers can be shared across GCNs. We omit the equation for space – it simply replaces $W^{[A]}_{(1)}$ and $W^{[B]}_{(1)}$ in Eq.(5) by $W_{(1)}$. However, the multiple graphs can hardly have the same raw attribute space. So, when $d = d_A = d_B$ but $m_A \neq m_B$, we assume that linear factorized components of $W^{[A]}_{(1)}$ and $W^{[B]}_{(1)}$ can be shared. We denote the shared component by $W_{(1)} \in \mathbb{R}^{\hat{m} \times d}$, where $\hat{m} = \min\{m_A, m_B\}$ or a smaller value. Then the loss function of joint training is as follows:

$$\begin{aligned} f(Q^{[A]}, Q^{[B]}; W_{(1)}, W_{(2)}) = {} & \alpha^{[A]} \cdot \mathcal{L}\left(g(X^{[A]}Q^{[A]}, A^{[A]}, W_{(1)}, W_{(2)}),\ A^{[A]}\right) \\ & + \alpha^{[B]} \cdot \mathcal{L}\left(g(X^{[B]}Q^{[B]}, A^{[B]}, W_{(1)}, W_{(2)}),\ A^{[B]}\right), \qquad (6) \end{aligned}$$

where $Q^{[A]} \in \mathbb{R}^{m_A \times \hat{m}}$ and $Q^{[B]} \in \mathbb{R}^{m_B \times \hat{m}}$ are linear transformation matrices on raw feature spaces for aligning the first-layer weight parameter matrices.

When the number of graphs is two, we can perform alternatives to save one linear transformation matrix. Alternative 1 is to use $Q^{[A \rightarrow B]} \in \mathcal{R}^{m_A \times m_B}$ to align $\mathcal{G}^{[A]}$'s raw attribute space to $\mathcal{G}^{[B]}$'s:

$$\begin{aligned} f(Q^{[A \rightarrow B]}; W^{[B]}_{(1)}, W_{(2)}) = {} & \alpha^{[A]} \cdot \mathcal{L}\left(g(X^{[A]}Q^{[A \rightarrow B]}, A^{[A]}, W^{[B]}_{(1)}, W_{(2)}),\ A^{[A]}\right) \\ & + \alpha^{[B]} \cdot \mathcal{L}\left(g(X^{[B]}, A^{[B]}, W^{[B]}_{(1)}, W_{(2)}),\ A^{[B]}\right), \qquad (7) \end{aligned}$$

And alternative 2 is to use $Q^{[B \rightarrow A]} \in \mathcal{R}^{m_B \times m_A}$ to align $\mathcal{G}^{[B]}$'s raw attribute space to $\mathcal{G}^{[A]}$'s.

3.2 Training Strategies across GCNs

The training strategy presented in the above section is *joint training* or referred as multi-task learning. The weights of the tasks, i.e., $\alpha^{[A]}$ and $\alpha^{[B]}$, are determined prior to the training procedure. The other popular training strategy is called *pre-training*, that is to define one task a time as the target task and all the other tasks as source tasks. Take the method of sharing $W_{(2)}$ as an example. The loss function of joint training was given in Eq.(5). Now if learning graph $\mathcal{G}^{[B]}$ is the target task, then learning graph $\mathcal{G}^{[A]}$ is the source task. The pre-training procedure has two steps. Their loss functions are:

$$f_1(W^{[A]}_{(1)}; W_{(2)}) = \mathcal{L}\left(g(X^{[A]}, A^{[A]}, W^{[A]}_{(1)}, W_{(2)}),\ A^{[A]}\right),$$
$$f_2(W^{[B]}_{(1)}; W_{(2)}) = \mathcal{L}\left(g(X^{[B]}, A^{[B]}, W^{[B]}_{(1)}, W_{(2)}),\ A^{[B]}\right). \quad (8)$$

Each step still minimizes Eq.(5): specifically, the first step uses $\alpha^{[A]} = 1$ and $\alpha^{[B]} = 0$; and the second step uses $\alpha^{[A]} = 0$ and $\alpha^{[B]} = 1$. The first step warms up the training of the shared $W_{(2)}$ on $\mathcal{G}^{[A]}$ so that the "fine-tuning" step on $\mathcal{G}^{[B]}$ can find more effective model parameters than starting from pure randomization. When the source and target are swapped, the two steps are swapped.

3.3 Aligning Representation Spaces with Regularizations

Besides sharing model parameters, an idea of bridging the gap between two GCNs is to align their output representation spaces. Given GCNs trained on two graphs $\mathcal{G}^{[A]}$ and $\mathcal{G}^{[B]}$, if two nodes $u^{[A]}_i$ and $u^{[B]}_j$ are aligned, we assume that their representations, $\vec{u}^{[A]}_i \in \mathbb{R}^{d_A}$ and $\vec{u}^{[B]}_j \in \mathbb{R}^{d_B}$ are highly correlated. For example, one named entity (person, location, or organization) in two different knowledge graphs should have very similar representations. The same goes for one researcher in two collaboration networks (as an investigators in projects or an author in papers).

We add the network alignment information as two types of regularization terms into the loss function. The first type is *Hard regularization*. It assumes that the aligned nodes have exactly the same representations in the two output spaces. This requires the two spaces to have the same number of dimensions: $d_A = d_B$. The term is written as:

$$h(W^{[A]}_{(1)}, W^{[A]}_{(2)}, W^{[B]}_{(1)}, W^{[B]}_{(2)}) = \left\| U^{[A]} - A^{[A,B]}\ U^{[B]} \right\|^2_{\mathrm{F}}. \quad (9)$$

The entire loss function is as below:

$$\begin{aligned} f(\Theta^{[A]}, \Theta^{[B]}) &= f(W^{[A]}_{(1)}, W^{[A]}_{(2)}, W^{[B]}_{(1)}, W^{[B]}_{(2)}) \\ = \quad & (1-\beta) \cdot \alpha^{[A]} \cdot \mathcal{L}\left(g(X^{[A]}, A^{[A]}, W^{[A]}_{(1)}, W^{[A]}_{(2)}),\ A^{[A]}\right) \\ & + (1-\beta) \cdot \alpha^{[B]} \cdot \mathcal{L}\left(g(X^{[B]}, A^{[B]}, W^{[B]}_{(1)}, W^{[B]}_{(2)}),\ A^{[B]}\right) \\ & + \beta \cdot h(W^{[A]}_{(1)}, W^{[A]}_{(2)}, W^{[B]}_{(1)}, W^{[B]}_{(2)}), \end{aligned} \quad (10)$$

where β is the weight of the regularization task.

The second type is *Soft regularization* which is designed for more common cases that $d_A \neq d_B$. It assumes that the aligned nodes' representations from one GCN can be linearly transformed into the ones in the other GCN's output space:

$$h(\Theta^{[A]}, \Theta^{[B]}, R^{[A \to B]}) = \left\| U^{[A]} R^{[A \to B]} - A^{[A,B]}\ U^{[B]} \right\|^2_{\mathrm{F}}, \quad (11)$$

where $R^{[A \to B]} \in \mathbb{R}^{d_A \times d_B}$ is the transformation matrix. Note that $R^{[A \to B]}$ is treated as model parameters in the loss function.

3.4 Alignment Reconstruction

When the two graphs are partially aligned, the network alignment information is usually sparse and incomplete. To address this issue, we treat the alignment information as a bipartite graph $\mathcal{G}^{[A,B]}$ and learn to reconstruct and complete the graph. Given the node representations $U^{[A]} \in \mathbb{R}^{n_A \times d_A}$ and $U^{[B]} \in \mathbb{R}^{n_B \times d_B}$, can we reconstruct the adjacency matrix of observed alignment data $\mathcal{G}^{[A,B]}$, i.e., $I^{[A,B]} \odot A^{[A,B]} \in \mathcal{R}^{n_A \times n_B}$, where $I^{[A,B]}$ is the indicator matrix of the observations? We introduce $R^{[A,B]} \in \mathbb{R}^{d_A \times d_B}$ and minimize the following optimization term:

$$h = \left\| \sigma\left(U^{[A]}\ R^{[A,B]}\ U^{[B]\top}\right) - I^{[A,B]} \odot A^{[A,B]} \right\|^2_{\mathrm{F}}. \quad (12)$$

The entire loss function optimizes on three graph reconstruction tasks ($\mathcal{G}^{[A]}$, $\mathcal{G}^{[B]}$, and $\mathcal{G}^{[A,B]}$), by replacing h in Eq.(10) with the above equation. The learning procedure transfers knowledge across the three tasks to find effective representations $U^{[A]}$ and $U^{[B]}$.

3.5 Beyond Two: Extend to Multiple GCNs

Suppose we have $K > 2$ GCNs. All the above methods (i.e., loss functions) can be easily extended from 2 to K GCNs. For example, the loss of joint multi-GCN training with *shared* W_2 can be written as below (extended from Eq.(5)):

$$f(\{W^{[i]}_{(1)}\}|^K_{i=1}; W_{(2)}) = \textstyle\sum^K_{i=1} \alpha^{[i]} \mathcal{L}\left(g(X^{[i]}, A^{[i]}, W^{[i]}_{(1)}, W_{(2)}),\ A^{[i]}\right). \quad (13)$$

And the loss of joint multi-GCN training with *shared* W_1 *and* W_2 can be written as below (extended from Eq.(6)):

$$f(\{Q^{[i]}\}|^K_{i=1}; W_{(1)}, W_{(2)}) = \textstyle\sum^K_{i=1} \alpha^{[i]} \mathcal{L}\left(g(X^{[i]}Q^{[i]}, A^{[i]}, W_{(1)}, W_{(2)}),\ A^{[i]}\right). \quad (14)$$

When using the network alignment information, the *soft regularization* term is written as below (extended from Eq.(11)):

$$\begin{aligned} & h\left(\{\Theta^{[i]}\}|^K_{i=1},\ \{R^{[i \to j]}\}|_{i<j;\ i,j \in \{1 \dots n\}}\right) \\ = & \sum_{i<j;\ i,j \in \{1 \dots n\}} \gamma^{[i \to j]} \left\| U^{[i]} R^{[i \to j]} - A^{[i,j]}\ U^{[j]} \right\|^2_{\mathrm{F}}, \end{aligned} \quad (15)$$

where $\gamma^{[i \to j]}$ is the regularization weight of aligning representation spaces between graphs $\mathcal{G}^{[i]}$ and $\mathcal{G}^{[j]}$.

The optimization term for alignment reconstruction and completion can be written as below (extended from Eq.(16)):

$$h = \sum_{i<j} \gamma^{[i \to j]} \left\| \sigma\left(U^{[i]}\ R^{[i,j]}\ U^{[j]\top}\right) - I^{[i,j]} \odot A^{[i,j]} \right\|^2_{\mathrm{F}}. \quad (16)$$

3.6 Implementation on Relational Graphs: Relation Classification across GCNs

Schlichtkrull *et al.* proposed neural relational modeling that uses GCNs for relation classification on relational graphs [31]. The output node representations from two-layer GCN are given as below:

$$U = \mathrm{softmax}\left(\sum_{r \in \mathcal{R}} \frac{1}{c_r} A_{:,:,r}\ \mathrm{ReLU}\left(\sum_{r \in \mathcal{R}} \frac{1}{c_r} A_{:,:,r}\ X\ W_{r,(1)}\right)\ W_{r,(2)}\right). \quad (17)$$

Clearly, it can be derived from Eq.(1) by making these changes:

- The relational graph $\mathcal{G}_{rel} = \{\mathcal{U}, \mathcal{E}, \mathcal{R}\}$ is denoted as a three-way tensor $A \in \{0,1\}^{n \times n \times n_R}$, where $n = |\mathcal{U}|$ is the number of entity nodes and $n_R = |\mathcal{R}|$ is the number of relation types;
- The model parameters are two three-way tensors: $\Theta = \{W_{r,(1)}|_{r \in \mathcal{R}} \in \mathbb{R}^{m \times d}, W_{r,(2)}|_{r \in \mathcal{R}} \in \mathbb{R}^{d \times d}\}$;
- c_r controls the weight of relation type r.

Relation classification can be considered as relational link prediction if we apply the prediction on all the possible entity-relation triples. The full prediction matrix is:

$$g(X, A, W_{(1)}, W_{(2)}, D) = \sigma(U \otimes D \otimes U^{\top}) \in [0,1]^{n \times n \times n_R}, \quad (18)$$

where $D \in \mathbb{R}^{d \times d \times n_R}$ is a three-way core tensor. For each relation type r, $D_{:,:,r}$ is a $d \times d$-diagonal matrix. The loss function is below:

$$\begin{aligned} f(W_{(1)}, W_{(2)}, D) &= \mathcal{L}\left(g(X, A, W_{(1)}, W_{(2)}, D),\ A\right) \\ &= -\sum_{i,j \in \{1 \ldots n\};\ r \in \mathcal{R}} A_{i,j,r} \log\left(\sigma(\vec{u_i}^{\top}\ D_{:,:,r}\ \vec{u_j})\right) \quad (19) \end{aligned}$$

All the proposed methods for partially aligned GCNs in the previous sections can be applied here. The key differences are (1) a three-way tensor data for reconstruction and (2) additional parameters D.

4 THEORETICAL ANALYSIS

In this section, we present theoretical results for partially aligned GCNs. First, we study the choices of the number of dimensions of node representations that may perform no knowledge transfer across GCNs. Then, we study two factors of positive transfer across GCNs. Note that many details are given in Appendix to save space.

4.1 Choices of the Number of Dimensions d

The task of graph reconstruction is to use the node representations $U \in \mathbb{R}^{n \times d}$ to recover the adjacency matrix $A \in \mathbb{R}^{n \times n}$. With singular value decomposition, we have $A \simeq \hat{U}\ \Sigma\ \hat{U}^{\top}$, where $\hat{U} \in \mathbb{R}^{n \times \text{rank}(A)}$ and Σ is square diagonal of size $\text{rank}(A) \times \text{rank}(A)$. So, the loss function in Eq.(3) can be approximated as the squared loss:

$$f = \left\| U\ B - \hat{U}\ \sqrt{\Sigma} \right\|_{\text{F}}^{2}, \quad (20)$$

where $B \in \mathcal{R}^{d \times \text{rank}(A)}$. It transforms the $n \times n$ classification tasks (for link prediction) into $\text{rank}(A)$ regression tasks.

Proposition 1. *Suppose we have K GCNs. In a linear setting for every model in the partially aligned GCNs, the loss function becomes:*

$$f\left(\{\Theta_i = W_{(1)}^{[i]} W_{(2)}^{[i]} B^{[i]}\}|_{i=1}^{K}\right) = \textstyle\sum_{i=1}^{K} \left\| \hat{A}^{[i]\ 2} X^{[i]} \Theta_i - \hat{U}^{[i]}\ \sqrt{\Sigma^{[i]}}\ \right\|_{\text{F}}^{2}. \quad (21)$$

The optimal solution for the i-th GCN is

$$\Theta_i = \left(X^{[i]\top} \hat{A}^{[i]\ 4} X^{[i]}\right)^{\dagger} X^{[i]\top} \hat{A}^{[i]\ 2} \hat{U}^{[i]} \sqrt{\Sigma^{[i]}} \in \mathcal{R}^{m_i \times \text{rank}(A)}, \quad (22)$$

where $X^{\dagger}$ denotes X's pseudoinverse. Hence a capacity of $\text{rank}(A^{[i]})$ *suffices for the i-th GCN. If $d \geq \sum_{i=1}^{K} \text{rank}(A^{[i]})$, then $B^{[i]}$ can be an identity matrix and there is* no transfer *between any two GCNs. In that case, no matter $\{W_{(1)}^{[i]}\}|_{i=1}^{K}$ or $\{W_{(2)}^{[i]}\}|_{i=1}^{K}$ are shared or not, these exists an optimum $W_{(1)}^{[i]*} W_{(2)}^{[i]*} = \Theta_i$, for all $i = 1 \ldots K$.*

The illustration of this idea and extension to nonlinear settings are discussed in Appendix.

4.2 Positive Transfer across GCN Models

We apply the theory of *positive transfer* in multi-task learning (MTL) [39] into cross-GCNs learning. If the cross-GCNs training improves over just training the target GCN on one particular graph, we say the source GCNs on other graphs *transfer positively* to the target GCN. [39] proposed a theorem to quantify how many data points are needed to guarantee positive transfer between two ReLU model tasks. By the above section, it is necessary to limit the capacity of the shared model parameters to enforce knowledge transfer. Here we consider $d = 1$ and $m = m_A = m_B$ to align with the theorem in [39]. We have a theorem for positive transfer between two *one-layer* GCNs that share the first-layer model parameters $W_{(1)}$:

Theorem 1. *Let $(\hat{A}^{[i]} X^{[i]} \in \mathbb{R}^{n_i \times m}, \vec{\hat{u}}^{[i]} \sqrt{\lambda_1^{[i]}}) \in \mathbb{R}^{n_i}$ denote the data of graph $\mathcal{G}^{[i]}$ ($i = \{A, B\}$). Suppose the output is generated by $ReLU(\hat{A}^{[i]} X^{[i]}\ \theta^{[i]})$. And suppose $c \geq \sin(\theta^{[A]}, \theta^{[B]})/\kappa(\hat{A}^{[B]} X^{[B]})$. Denote by $W_{(1)}^{\star}$ the optimal solution of the one-layer setting of Eq.(6): $f(W_{(1)}) = \sum_{i \in \{A,B\}} \mathcal{L}\left(g(X^{[i]}, A^{[i]}, W_{(1)}),\ A^{[i]}\right)$. With probability $1 - \delta$ over the randomness of $(\hat{A}^{[A]} X^{[A]}, \vec{\hat{u}}^{[A]} \sqrt{\lambda_1^{[A]}})$, when*

$$n_i \geq \max\left(\frac{m \log m}{c^2}\left(\frac{1}{c^2} + \log m\right), \frac{\|\vec{\hat{u}}^{[B]} \sqrt{\lambda_1^{[B]}}\|^2}{c^2}\right), \quad (23)$$

we have that the estimation error is at most:

$$\sin(W_{(1)}^{\star}, \theta^{[A]}) \leq \sin(\theta^{[A]}, \theta^{[B]}) + O(c/\kappa(\hat{A}^{[B]} X^{[B]})). \quad (24)$$

Notations for the above theorem. $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ denotes the condition number of $A \in \mathbb{R}^{n \times m}$, where $\lambda_{\max}(A)$ denotes its largest singular value and $\lambda_{\min}(A)$ denotes its $\min\{n, m\}$-th largest singular value. As MTL is effective over "similar" tasks [41], the cross-GCNs learning can be effective across similar GCN models. A natural definition of GCN similarity is $\cos(\theta^{[A]}, \theta^{[B]})$: higher means more similar. So $\sin(W^{\star}, \theta) = \sqrt{1 - \cos^2(W^{\star}, \theta)}$ can denote the estimation error. Proof of the theorem can be referred in [39].

4.3 Positive Transfer with Network Alignment

Soft regularization: We transfer knowledge across two GCNs' representation spaces by optimizing Eq.(10) where h is specified as Eq.(11). The optimal solution is

$$\begin{aligned} U^{[A]\star} &= \left((1-\beta) \cdot \alpha^{[A]}\ I + \beta R^{[A \to B]} R^{[A \to B]\top}\right)^{-1} \cdot \\ &\cdot \left((1-\beta) \cdot \alpha^{[A]}\ \hat{U}^{[A]} \sqrt{\Sigma^{[A]}} + \beta\ A^{[A,B]} U^{[B]\star} R^{[A \to B]\top}\right) \quad (25) \end{aligned}$$

So, if $\beta = 0$, $U^{[A]\star} = \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}$ may overfit the graph data $\mathcal{G}^{[A]}$; if $\beta \in (0,1]$, the information transferred from $\mathcal{G}^{[B]}$ via the alignment $A^{[A,B]}$, i.e., $A^{[A,B]} U^{[B]\star} R^{[A \to B]}$, can have positive impact on link prediction on $\mathcal{G}^{[A]}$.

Alignment reconstruction: Suppose we optimize Eq.(10 where h is in Eq.(16). The optimal solution is

$$\begin{aligned} U^{[A]\star} &= \left((1-\beta) \cdot \alpha^{[A]}\ \hat{U}^{[A]} \sqrt{\Sigma^{[A]}} + \beta A^{[A,B]} U^{[B]\star} R^{[A,B]\top}\right) \\ &\cdot \left((1-\beta) \cdot \alpha^{[A]}\ I + \beta R^{[A,B]} U^{[B]\star\top} U^{[B]\star} R^{[A,B]\top}\right)^{-1}. \quad (26) \end{aligned}$$

So, if $\beta = 0$, $U^{[A]\star} = \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}$ may overfit the graph data $\mathcal{G}^{[A]}$; if $\beta \in (0,1]$, the information transferred from $\mathcal{G}^{[B]}$ via the alignment $A^{[A,B]}$, i.e., $A^{[A,B]} U^{[B]\star} R^{[A,B]\top}$, can have positive impact on link prediction on $\mathcal{G}^{[A]}$.

		Sharing model	Regularizations	Alignment reconstruction	Equation(s)	Validation				Test				Validation		Test	
						NSFAward (A)		DBLP (B)		NSFAward (A)		DBLP (B)		**Overall**			
						AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
VGAE		×	×	×	-	0.7657	0.7397	0.8159	0.8121	0.7522	0.7486	0.8143	0.7986	0.7900	0.7742	0.7820	0.7728
Separated		×	×	×	(4)	0.7725	0.7623	0.8414	0.8278	0.7691	0.7659	0.8395	0.8220	0.8055	0.7937	0.8028	0.7930
Cross-GCN learning	M1	×	Soft	×	(10), (11)	0.7727	0.7642	0.8411	0.8278	0.7739	0.7726	0.8394	0.8219	0.8055	0.7947	0.8053	0.7965
	M2	×	×	√	(10), (12)	0.8089	0.8131	0.8402	0.8293	0.8127	0.8240	0.8330	0.8220	0.8243	0.8211	0.8227	0.8230
	M3	$W_{(2)}$	×	×	(8): first A: then B:	0.7739 0.7373	0.7621 0.7148	0.5453 0.8535	0.5366 0.8451	0.7699 0.7264	0.7645 0.7154	0.5520 0.8535	0.5419 0.8399	- 0.7912	- 0.7745	- 0.7848	- 0.7727
	M4		×	×	(8): first B: then A:	0.5784 0.8098	0.6205 0.8085	0.8419 0.6954	0.8285 0.6470	0.5840 0.8056	0.6234 0.8095	0.8402 0.6953	0.8225 0.6479	- 0.7483	- 0.7188	- 0.7464	- 0.7197
	M5		×	×	(5): Joint	0.8176	0.8223	0.8626	0.8569	0.8122	0.8225	0.8602	0.8497	0.8395	0.8392	0.8355	0.8359
	M6		Hard	×	(5), (10), (9)	0.8246	0.8313	0.8572	0.8487	0.8266	0.8395	0.8569	0.8446	0.8406	0.8399	0.8415	0.8420
	M7		Soft	×	(5), (10), (11)	0.8333	0.8423	0.8675	0.8604	0.8334	0.8473	0.8653	0.8543	0.8501	0.8513	0.8491	0.8508
	M8		×	√	(5), (10), (12)	0.8420	0.8533	0.8758	0.8721	0.8402	0.8551	0.8727	0.8649	0.8586	0.8626	0.8561	0.8600
	M9	$W_{(1)}$ and $W_{(2)}$	×	×	(7): A → B	0.8020	0.8112	**0.8854**	**0.8797**	0.7988	0.8115	**0.8848**	**0.8792**	0.8416	0.8441	0.8396	0.8440
	M10		×	×	(7): B → A	0.8306	0.8343	0.8643	0.8584	0.8293	0.8325	0.8607	0.8504	0.8471	0.8462	0.8447	0.8414
	M11		×	×	(6): A and B	0.8311	0.8527	0.8644	0.8596	0.8335	0.8529	0.8610	0.8566	0.8474	0.8561	0.8470	0.8547
	M12		Soft	×	(6), (10), (11)	0.8521	0.8564	0.8759	0.8699	**0.8512**	0.8606	0.8757	0.8681	0.8638	0.8631	0.8633	0.8643
	M13		×	√	(6), (10), (12)	**0.8557**	**0.8683**	0.8810	0.8770	**0.8512**	**0.8659**	0.8799	0.8733	**0.8682**	**0.8726**	**0.8653**	**0.8696**

Table 1: Performance of VGAE [19], (Separated) GCNs [20], and 13 cross-GCN learning models on link prediction in NSFAward and DBLP academic graphs ($\mathcal{G}^{[A]}$ and $\mathcal{G}^{[B]}$). The models take various options on model sharing, regularizations, and alignment reconstruction. Equations can be found in Sections 2 and 3. Higher AUC or AP means better performance. We investigated the standard deviation of all the cells: all the values are smaller than 0.0016 and thus omitted for space. Underlined numbers are targeted values in pre-training strategies. Bolded numbers are the highest on the columns.

5 EXPERIMENTS

In this section, I perform cross-network learning on three types of graph learning tasks. For each task, I present datasets (stats can be found in Figure 2), experimental settings, results, and analysis.

5.1 Link Prediction

5.1.1 Datasets. Below are two collaboration networks I use that can be considered as weighted graphs between researcher nodes.

- NSFAward: It is publicly available on the National Science Foundation (NSF) Awards database from 1970 to 2019.[1] Nodes are investigators of the awards. Two investigator nodes have a link if they are participated in the same award(s) and the weight of the link is the number of their shared award(s).
- DBLP: I expand the DBLP-4Area [15] from 5,915 authors to 14,387 authors by adding their co-authors in the graph. The weight of the link between two author nodes is the number of their co-authored papers.

5.1.2 Experimental settings. For each graph, I randomly split the observed links into training, validation, and test sets by 8:1:1. Then I randomly select unobserved links as negative examples in the validation and test sets. Positives and negatives are balanced. I use Area Under the ROC Curve (AUC) and Average Precision (AP) as evaluation metrics. I perform the evaluation 10 times, calculate mean value and standard deviation. I also report the "overall" performance by calculating the harmonic mean of the AUC/AP on the two datasets. All models are developed by PyTorch and one NVIDIA GeForce RTX 2080 Ti graphic card. I set $d_A = d_B = 64$. I use grid search on the hyperparameters in $\{0.1, 0.2, \ldots, 0.9\}$ and find that the best combination is $\alpha^{[A]} = \alpha^{[B]} = \beta = 0.5$.

5.1.3 Results. Table 1 presents the performance of as many as 13 cross-GCN learning models I implemented. Models M3–M8 share $W_{(2)}$ across the NSFAward and DBLP academic graphs. Models M9–M13 share both $W_{(1)}$ and $W_{(2)}$. M6 uses hard regularization; M1, M7, and M12 use soft regularization; and M2, M8, and M13 use alignment reconstruction. I have the following observations:

O1. *Cross-network models perform better than baselines and the best is M13:* M13 shares $W_{(1)}$ and $W_{(2)}$ between two GCNs, and uses alignment reconstruction to bridge the two representation spaces. In terms of the overall AUC on the two test sets, M13 achieves 0.865, which is significantly higher than VGAE's 0.782 [19] and (Separated) GCNs' 0.803 [20]. The AP is improved from VGAE's 0.773 and GCN's 0.793 to 0.870. Clearly, the knowledge is successfully transferred across the GCN models on two academic graphs. Performance on test sets is worse than but very close to that on validation sets, which indicates no or very little overfitting in the GCN models. Both models need information beyond the corresponding graph data for effective learning and decision-making on the test data.

O2. *Sharing weight matrices is beneficial, and sharing those of both layers is beneficial:* Suppose we do not use the alignment information at all: the only possible bridge across two GCNs is weight matrices $W_{(1)}$ and $W_{(2)}$. M3–M5 share $W_{(2)}$. Compared with separated GCNs, M5 improves the overall AUC on test sets from 0.803

[1] NSF Award Search: https://www.nsf.gov/awardsearch/download.jsp

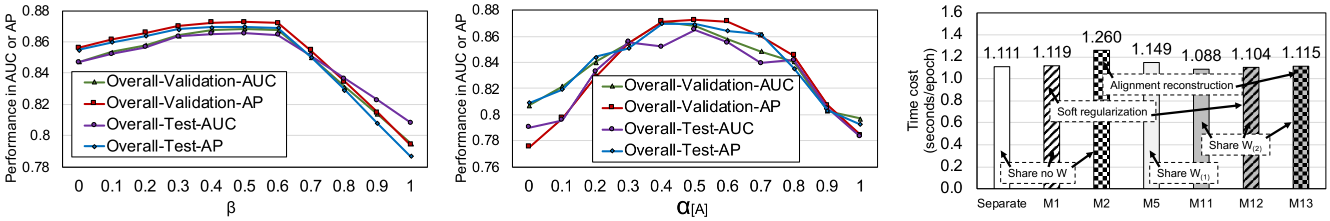


Figure 3: Sensitivity analysis on M13 (sharing $W_{(1)}$ and $W_{(2)}$, plus alignment reconstruction): (a) on the weight of the alignment reconstruction task β, (b) on the weight of learning on graph $\mathcal{G}^{[A]}$: $\alpha^{[A]}$. And (c) running time analysis in seconds per epoch.

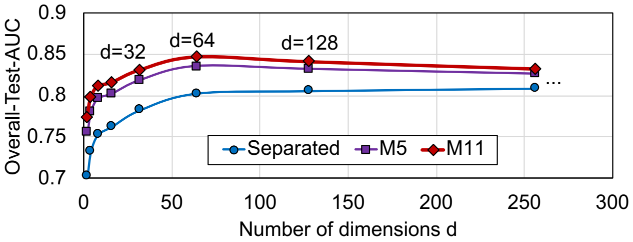


Figure 4: Empirical analysis on the choice of d.

to 0.836. And M9–M11 share both $W_{(1)}$ and $W_{(2)}$. Compared with M5, M11 improves the AUC from 0.836 to 0.847. When soft regularization is adopted, we can compare M7 and M12 with M1; when alignment reconstruction is adopted, we can compare M8 and M13 with M2. The conclusions are the same.

O3. *If $W_{(2)}$ is shared, joint training is slightly better than pre-training:* M3 pre-trains the model with $\mathcal{G}^{[A]}$ and then fine-tunes on $\mathcal{G}^{[B]}$; M4 switches the use of the graphs. So we focus on M3's performance on $\mathcal{G}^{[B]}$ and M4's on $\mathcal{G}^{[A]}$. Compared with separated GCNs, M3 improves AUC on the test set from 0.840 to 0.854; M4 improves it from 0.769 to 0.806. However, with a joint training strategy, M5 achieves an AUC on $\mathcal{G}^{[B]}$ of 0.860 over M3's 0.854 and achieves an AUC on $\mathcal{G}^{[A]}$ of 0.812 over M4's 0.806.

O4. *If $W_{(1)}$ is shared, learning linear transformation matrices on both graphs is slightly better than learning a matrix on one of them:* M11 learns $Q^{[A]} \in \mathbb{R}^{m_A \times \hat{m}}$ and $Q^{[B]} \in \mathbb{R}^{m_B \times \hat{m}}$. M9 learns $Q^{[A \rightarrow B]}$ only, and M10 learns $Q^{[B \rightarrow A]}$ only. In terms of the overall AUC on the two test sets, compared with M5 that shares $W_{(2)}$ only, M9 improves AUC from M5's 0.836 to 0.840, M10 achieves an AUC of 0.845, and M11 achieves an AUC of 0.847.

O5. *Regularizations, hard or soft, are beneficial; and alignment reconstruction is better:* (a) M6 adds hard regularization to M5. M6 improves the overall AUC on the two test sets from M5's 0.836 to 0.842. (b) M1, M7, and M12 add soft regularization to separated GCNs, M5, and M11, respectively. M1 improves the AUC from 0.803 to 0.805, M7 improves from 0.836 to 0.849, and M13 improves from 0.847 to 0.863. With more model parameters (e.g., weight matrices) shared, the improvements by regularizations become bigger. (c) M8 and M13 add alignment reconstruction to M5 and M11, respectively. M8 improves the AUC from 0.836 to 0.856. M13 improves from 0.847 to 0.865. The AUC scores are quite significantly improved by alignment reconstruction, slightly higher than regularizations.

5.1.4 Hyperparameter sensitivity. In Figure 3(a) and (b) we investigate the effect of hyperparameters β and $\alpha^{[A]}$ on the performance of model M13. When $\beta = 0$, M13 is equivalent with M11. With β increasing, the performance becomes better by learning to reconstruct the network alignment. It is the best when $\beta \in [0.4, 0.6]$. Then a too big β will overfit the model on the alignment reconstruction, so the AUC/AP decreases quickly when β goes from 0.6 to 1. Given $\beta = 0.5$, a too big or too small $\alpha^{[A]}$ will overfit one of the graph data, $\mathcal{G}^{[A]}$ or $\mathcal{G}^{[B]}$. When $\alpha^{[A]}$ is around 0.5, the performance is the best. So, all the optimization terms are important.

5.1.5 Time complexity. Figure 3(c) presents the time cost per epoch of separated GCNs, M1, M2, M5, M11–M13. The models that share both $W_{(1)}$ and $W_{(2)}$ (M11), share $W_{(2)}$ only (M5), and share neither of them (Separated) do not show significant difference of time cost. Adopting soft regularizations (M1 vs Separated, M12 vs M11) increases the time cost slightly. And adopting alignment reconstruction (M2 vs M1, M13 vs M12) take slightly more time. The time complexity remains at the same level in the proposed models.

5.1.6 On the choice of d. Figure 4 presents the overall AUC on test sets of Separated GCNs, M5 (sharing $W_{(2)}$), and M11 (sharing $W_{(1)}$ and $W_{(2)}$), when the number of dimensions d is $\{2, 2^2, \ldots, 2^8\}$. When d becomes bigger, the improvements by model parameters sharing become smaller. As presented in theoretical analysis, small d would force the knowledge transfer across the GCN models; when d is too big, there might be no transfer. We choose $d = 64$ as the default number of dimensions of the latent spaces.

5.2 Relation Classification

5.2.1 Datasets. Below are four knowledge graphs I use that can be considered as relational graphs between entity nodes.

- FB15K: It is a subset of the relational database Freebase [31].
- WN18: It is a subset of WordNet containing lexical relations.
- YAGO: The open source knowledge base developed by MPI.
- DBpedia500: A subset of structured content in the Wikipedia.

I will investigate whether incorporating partially-aligned graphs YAGO and DBPedia500 can improve the performance of relation classification on FB15K and WN18, respectively. Entities are aligned if they have the same surface name. Resolving the inaccuracy of entity alignment is important but out of this work's scope.

5.2.2 Experimental settings. Both FB15K and WN18 have standard splits for training, validation, and test. I provide results using two common evaluation metrics: mean reciprocal rank (MRR) and Hits at k (Hits@k). I report both raw and filtered MRR (with filtered MRR typically considered more reliable) and filtered Hits@1 and Hits@3, following TransE [1]. Other baseline methods include:

		FB15K			WN18		
		MRR	Hits @		MRR	Hits @	
		Raw / Filtered	1	3	Raw / Filtered	1	3
TransE		0.221 / 0.380	0.231	0.472	0.335 / 0.455	0.089	0.823
HOlE		0.232 / 0.524	0.402	0.613	**0.616** / 0.938	0.930	0.945
DistMult		0.242 / 0.654	0.546	0.733	0.532 / 0.822	0.728	0.914
ComplEx		0.242 / 0.692	0.599	0.759	0.587 / **0.941**	**0.936**	0.945
R-GCN		0.262 / 0.696	0.601	0.760	0.562 / 0.819	0.697	0.929
CrossE		0.267 / 0.728	0.634	0.802	0.570 / 0.830	0.741	0.931
Cross- R-GCN learning (+YAGO / DBPedia500)	M1	0.263 / 0.698	0.603	0.762	0.563 / 0.821	0.699	0.931
	M2	0.268 / 0.711	0.614	0.776	0.574 / 0.828	0.712	0.939
	M5	0.271 / 0.720	0.622	0.786	0.581 / 0.830	0.721	0.942
	M11	0.274 / 0.729	0.629	0.796	0.588 / 0.836	0.730	0.948
	M12	0.279 / 0.741	0.639	0.809	0.598 / 0.841	0.742	0.954
	M13	**0.279 / 0.742**	**0.641**	**0.810**	0.599 / 0.843	0.743	**0.956**

Table 2: Performance of existing methods (e.g., RGCN [31]) and cross-R-GCN learning models on relation classification.

- HOlE [27]: it replaces vector-matrix product with circular correlation to compute knowledge graph embeddings.
- DistMult [42]: a neural embedding approach that learns representations of entities and relations.
- ComplEx [32]: it models asymmetric relations by generalizing DistMult to the complex domain.
- R-GCN [31]: it is the first work of relation graph convolutional network, considered the state of the art. And I use it as the basic GCN model for cross-GCN learning.
- CrossE [51]: its “interaction embeddings" are learned from both knowledge graph connections and link explanations.

I use the Adam optimizer with a learning rate of 0.01. Implementation details can be found in Appendix.

5.2.3 Results. Table 2 presents the results of the baselines and six cross-GCN learning models (M1, M2, M5, M11–M13). Model M5 shares $W_{(2)}$ across two knowledge graphs. M11–M13 share both $W_{(1)}$ and $W_{(2)}$. M1 and M12 use soft regularization; and M2 and M13 use alignment reconstruction. Our main observation is that *cross-network models perform better than baselines and the best model is M13.* With the large-scale YAGO, M13 improves the filtered MRR from 0.696 to 0.742 and the filtered Hits@1 from 0.601 to 0.641 for relation classification on FB15K, compared to a sole R-GCN model. With DBPedia500 which is bigger than YAGO, M13 improves the filtered MRR from 0.819 to 0.843 and the filtered Hits@1 from 0.697 to 0.743 on WN18, compared to R-GCN. Both M12 and M13 that share two-layers parameters and alignment information achieve higher performance than the best baseline CrossE on all the metrics.

6 RELATED WORK

6.1 Graph Neural Networks

Graph neural network (GNN) is a type of neural network that operates on graph structure to capture the dependence of graphs via message passing between the nodes of graphs [40, 52]. Kipf *et al.* proposed graph convolutional networks that use a spectral graph convolution to capture the spatial dependency [20]. Velivckovic *et al.* considered the interaction between nodes in the graph convolution measured as graph attention weight [33]. Hamilton *et al.* implemented the graph convolution as an efficient aggregation and sampling algorithm for inductive node representation learning [11].

Recently, GNN techniques have been applied for learning representations on relational graphs, specifically knowledge graphs [29, 30, 48]. Relational GNNs perform significantly better than conventional knowledge graph embeddings [31]. Most techniques suffer from the issue of data sparsity and our proposed methods address it with knowledge transfer across networks.

In this paper, we discussed weighted graph, relational graph, and bipartite graphs; however, heterogeneous graphs are ubiquitous and have unique essential challenges for modeling and learning [22, 24, 35–37]. We have been seeing heterogeneous GNNs [5, 9, 23, 47] and heterogeneous attention networks for graphs [13, 14, 38, 43, 44]. Extending our proposed cross-network learning to heterogeneous graphs and enabling inductive learning in the framework are very important. These two points are out of the scope of our work but worth being considered as future directions.

6.2 Network Alignment and Modeling

The alignment of nodes between two or more than two graphs has been created and/or utilized in at least three categories of methods [3, 10, 16, 17, 28, 34]. First, Kong *et al.* inferred anchor links between two social networks by matching the subgraphs surrounding the linked nodes [21]. Emmert *et al.* wrote a comprehensive survey on the algorithms of graph matching [7]. Second, Zhang *et al.* used graph pattern mining algorithms for node alignment on incomplete networks [50] and multi-level network [49]. Nassar *et al.* leveraged the multimodal information for network alignment [25]. The third line of work learned node representations and trained the supervised models with node pairing labels (if available) [6, 26, 45]. A matrix factorization-based model has demonstrated that cross-network or cross-platform modeling can alleviate the data sparsity on a single platform [18].

7 CONCLUSIONS

In this paper, I proposed partially aligned GCNs that jointly learn node representations across graphs. I provided multiple methods as well as theoretical analysis to positively transfer knowledge across the set of partially aligned nodes. Extensive experiments on real-world knowledge graphs and collaboration networks show the superior performance of the proposed models on relation classification and link prediction.

ACKNOWLEDGMENT

This research was supported by National Science Foundation award IIS-1849816.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013), 2787–2795.

[2] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. 2010. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence* 33, 8 (2010), 1548–1560.

[3] Zheng Chen, Xinli Yu, Bo Song, Jianliang Gao, Xiaohua Hu, and Wei-Shih Yang. 2017. Community-based network alignment for large attributed network. In

Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 587–596.

[4] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* 31, 5 (2018), 833–852.

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.

[6] Xingbo Du, Junchi Yan, and Hongyuan Zha. 2019. Joint Link Prediction and Network Alignment via Cross-graph Embedding.. In *IJCAI*. 2251–2257.

[7] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. 2016. Fifty years of graph matching, network alignment and network comparison. *Information sciences* 346 (2016), 180–197.

[8] 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.

[9] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*. 2331–2341.

[10] Shawn Gu and Tijana Milenković. 2020. Data-driven network alignment. *PloS one* 15, 7 (2020), e0234978.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1857–1867.

[14] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*. 2704–2710.

[15] Zhipeng Huang, Yudian Zheng, Reynold Cheng, Yizhou Sun, Nikos Mamoulis, and Xiang Li. 2016. Meta structure: Computing relevance in large heterogeneous information networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1595–1604.

[16] Meng Jiang, Peng Cui, Xumin Chen, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2015. Social Recommendation with Cross-Domain Transferable Knowledge. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 27, 11 (2015), 3084–3097.

[17] Meng Jiang, Peng Cui, Fei Wang, Qiang Yang, Wenwu Zhu, and Shiqiang Yang. 2012. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 1422–1431.

[18] Meng Jiang, Peng Cui, Nicholas J Yuan, Xing Xie, and Shiqiang Yang. 2016. Little Is Much: Bridging Cross-Platform Behaviors through Overlapped Crowds. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*. 13–19.

[19] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv:1611.07308* (2016).

[20] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[21] Xiangnan Kong, Jiawei Zhang, and Philip S Yu. 2013. Inferring anchor links across multiple heterogeneous social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 179–188.

[22] Siwei Liu, Iadh Ounis, Craig Macdonald, and Zaiqiao Meng. 2020. A Heterogeneous Graph Neural Model for Cold-Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2029–2032.

[23] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2077–2085.

[24] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3213–3223.

[25] Huda Nassar and David F Gleich. 2017. Multimodal network alignment. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 615–623.

[26] Huda Nassar, Nate Veldt, Shahin Mohammadi, Ananth Grama, and David F Gleich. 2018. Low rank spectral network alignment. In *Proceedings of the 2018 World Wide Web Conference*. 619–628.

[27] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[28] Kyle K Qin, Flora D Salim, Yongli Ren, Wei Shao, Mark Heimann, and Danai Koutra. 2020. G-CREWE: Graph CompREssion With Embedding for Network Alignment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1255–1264.

[29] Hongyu Ren and Jure Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. *Advances in Neural Information Processing Systems* 33 (2020).

[30] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems*. 15347–15357.

[31] 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.

[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *International Conference on Learning Representations*.

[34] Vipin Vijayan and Tijana Milenković. 2017. Multiple network alignment via multiMAGNA++. *IEEE/ACM transactions on computational biology and bioinformatics* 15, 5 (2017), 1669–1682.

[35] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous Graph Neural Networks for Extractive Document Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[36] Shen Wang, Zhengzhang Chen, Ding Li, Zhichun Li, Lu-An Tang, Jingchao Ni, Junghwan Rhee, Haifeng Chen, and Philip S Yu. 2019. Attentional heterogeneous graph neural network: Application to program reidentification. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 693–701.

[37] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[38] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[39] Sen Wu, Hongyang R Zhang, and Christopher Ré. 2019. Understanding and Improving Information Transfer in Multi-Task Learning. In *International Conference on Learning Representations*.

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[41] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multitask learning for classification with dirichlet process priors. *Journal of Machine Learning Research* 8, Jan (2007), 35–63.

[42] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[43] Xu Yang, Cheng Deng, Tongliang Liu, and Dacheng Tao. 2020. Heterogeneous Graph Attention Network for Unsupervised Multiple-Target Domain Adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[44] Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous Graph Transformer for Graph-to-Sequence Learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7145–7154.

[45] Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. 2019. A Vectorized Relational Graph Convolutional Network for Multi-Relational Network Alignment.. In *IJCAI*. 4135–4141.

[46] 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*. 974–983.

[47] 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.

[48] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*. 2735–2745.

[49] Si Zhang, Hanghang Tong, Ross Maciejewski, and Tina Eliassi-Rad. 2019. Multilevel network alignment. In *The World Wide Web Conference*. 2344–2354.

[50] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2017. ineat: Incomplete network alignment. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1189–1194.

[51] Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 96–104.

[52] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).

A ADDITIONAL INFORMATION FOR THEORETICAL ANALYSIS

A.1 Choices of the Number of Dimensions d

The total capacities (i.e., sum of ranks of adjacency matrices) can be implemented as $\sum_{i=1}^{K} \mathrm{argmax}_p \, \lambda_p(A^{[i]}) \geq \epsilon$, the sum of the minimum positions of no-smaller-than-ϵ singular value. To illustrate the idea, suppose $\{W_{(2)}^{[i]}\}$ are shared across GCNs, which means $W_{(2)} = W_{(2)}^{[1]} = \cdots = W_{(2)}^{[K]}$. As long as the shared $W_{(2)}^{*}$ contains $\Theta_i|_{i=1}^{K}$ in its column span, there exits $W_{(1)}^{[i]*}$ such that $W_{(1)}^{[i]*} W_{(2)}^{*} = \Theta_i$, which is optimal for Eq.(21) with minimum error. But this means no transfer across any GCNs. This can hurt generalization if a GCN has limited data $\mathcal{G}^{[i]}$, in which its single-GCN solution overfits training data, whereas the cross-network solution can leverage other graphs' data to improve generalization.

Extension to the ReLU/softmax setting. If the capacity of the shared model parameters (i.e., the number of dimensions d) is larger than the total capacities, then we can share both $\{W_{(1)}^{[i]}\}|_{i=1}^{K}$ and $\{W_{(2)}^{[i]}\}|_{i=1}^{K}$. This remains an optimal solution to the joint training problem in the ReLU/softmax setting. Furthermore, there is *no transfer* between any GCNs through the shared model parameters.

A.2 Positive Transfer with Network Alignment

Soft regularization: The loss function can be re-written as:

$$\begin{aligned} f \quad = \quad & (1-\beta) \cdot \alpha^{[A]} \, \|U^{[A]} - \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}\|_{\mathrm{F}}^{2} \\ & + (1-\beta) \cdot \alpha^{[B]} \, \|U^{[B]} - \hat{U}^{[B]} \sqrt{\Sigma^{[B]}}\|_{\mathrm{F}}^{2} \\ & + \beta \, \|U^{[A]} R^{[A \rightarrow B]} - A^{[A,B]} U^{[B]}\|_{\mathrm{F}}^{2}. \end{aligned} \tag{27}$$

The gradient is

$$\begin{aligned} & \frac{\mathrm{d}\, f}{\mathrm{d}\, U^{[A]}} = 2 \, (1-\beta) \cdot \alpha^{[A]} \, (U^{[A]} - \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}) \\ & +2 \, \beta \, (U^{[A]} R^{[A \rightarrow B]} - A^{[A,B]} U^{[B]}) R^{[A \rightarrow B] \, \top}. \end{aligned} \tag{28}$$

Alignment reconstruction: The loss function is:

$$\begin{aligned} f \quad = \quad & (1-\beta) \cdot \alpha^{[A]} \, \|U^{[A]} - \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}\|_{\mathrm{F}}^{2} \\ & + (1-\beta) \cdot \alpha^{[B]} \, \|U^{[B]} - \hat{U}^{[B]} \sqrt{\Sigma^{[B]}}\|_{\mathrm{F}}^{2} \\ & + \beta \, \|U^{[A]} R^{[A,B]} U^{[B]\top} - A^{[A,B]}\|_{\mathrm{F}}^{2}. \end{aligned} \tag{29}$$

The gradient is

$$\begin{aligned} & \frac{\mathrm{d}\, f}{\mathrm{d}\, U^{[A]}} = 2 \, (1-\beta) \cdot \alpha^{[A]} \, (U^{[A]} - \hat{U}^{[A]} \sqrt{\Sigma^{[A]}}) \\ & +2 \, \beta \, (U^{[A]} R^{[A,B]} U^{[B]\top} - A^{[A,B]}) U^{[B]} R^{[A,B] \, \top}. \end{aligned} \tag{30}$$