TransGCN:Coupling Transformation Assumptions with Graph Convolutional Networks for Link Prediction

Ling Cai, Bo Yan, Gengchen Mai, Krzysztof Janowicz, Rui Zhu
STKO Lab, University of California, Santa Barbara
{ling.cai,boyan,gengchen_mai,jano,ruizhu}@geog.ucsb.edu

ABSTRACT

Link prediction is an important and frequently studied task that contributes to an understanding of the structure of knowledge graphs (KGs) in statistical relational learning. Inspired by the success of graph convolutional networks (GCN) in modeling graph data, we propose a unified GCN framework, named TransGCN, to address this task, in which relation and entity embeddings are learned simultaneously. To handle heterogeneous relations in KGs, we introduce a novel way of representing heterogeneous neighborhood by introducing transformation assumptions on the relationship between the subject, the relation, and the object of a triple. Specifically, a relation is treated as a transformation operator transforming a head entity to a tail entity. Both translation assumption in TransE and rotation assumption in RotatE are explored in our framework. Additionally, instead of only learning entity embeddings in the convolution-based encoder while learning relation embeddings in the decoder as done by the state-of-art models, e.g., R-GCN, the TransGCN framework trains relation embeddings and entity embeddings simultaneously during the graph convolution operation, thus having fewer parameters compared with R-GCN. Experiments show that our models outperform the-state-of-arts methods on both *FB15K-237* and *WN18RR*.

CCS CONCEPTS

• **Computing methodologies** → **Learning latent representations**; *Knowledge representation and reasoning*.

KEYWORDS

Knowledge Graph Embedding, Link Prediction, Graph Convolutional Network, Transformation Assumption, Neighborhood

1 INTRODUCTION

Knowledge graphs (KGs) such as DBpedia and Freebase that encode statements about the world around us have attracted growing attention from multiple fields, including question answering [7, 18], knowledge inference [19], recommendation systems [32], and so on. By their very nature KGs are far from complete as the state of the world evolves constantly. This has motivated work on automatically predicting new statements based on known statements. Among these inference tasks link prediction has become a main focus of statistical relational learning (SRL) [14].

A KG encodes structural information about entities and the abundant relations among them as a directed labeled multigraph, where entities are represented as nodes and relations between them as labeled, directed edges. Accordingly, in the Semantic Web context a statement in a KG can be represented as a triple (h, r, t), where h is the head entity, r the relation, and t the tail entity, respectively. The connectivity among triples in KGs provides the basis for link prediction.

Since the symbolic representations of KGs prohibit them from directly being incorporated in many machine learning tasks, recently many studies have proposed to embed entities and relations of a KG into low-dimensional vector spaces [1, 17, 22, 26, 30, 31], which can be further unitized in multiple downstream tasks, e.g., the aforementioned link prediction. Along this line, there are two main branches [27]: (1) translation-based methods, which predict the existence of a triple by measuring the distance between the head entity and the tail entity after a translation enforced by the corresponding relation, such as TransE [1], TransD [9], and TransR [17] and (2) Semantic Matching Energy based methods, which measure the existence of a triple as the compatibility of two entities and their relation in latent vector space, e.g., RESCAL [21], DistMult [31], ComplEx [26]. More recently, there are some other ideas. For example, rather than defining a relation as a translation from the subject to the object, Sun et al. [24] thought of a relation as a rotation from the subject to the object in the complex vector space and proposed RotatE, which was the first model that can handle symmetry/antisymmetry, inversion, and composition relations simulatenoiusly. Their experiments demonstrated the effectiveness of this assumption. More details about these methods can be found in Section 4.

Although there are multiple successful stories in both branches, these aforementioned models are all trained on individual triples independently regardless of their local neighborhood structures. Noticing this downside, Schlichtkrull et al. [22] state that explicitly modeling local structure can be an important supplement to help recover missing statements in KGs. Inspired by the success of graph convolutional networks (GCN) [13] in modeling structured neighborhood information of unlabeled and undirected graphs with convolution operations, the authors proposed a GCN-based method to model knowledge graphs (R-GCN). In R-GCN, which is an encoder, the embedding of each entity is learned based on its up to n degree neighboring entities by using n graph convolution layers. Then the encoder is trained jointly with a task-specific decoder, e.g., a DistMult-like decoder, to predict links.

The experimental results of applying R-GCN demonstrate the importance of integrating neighborhood information in knowledge graph embedding models. R-GCN aims at learning entity embeddings even though it utilizes relation-specific weight matrices. The relation embeddings are learned in the task-specific decoder while the learned relation-specific matrices in the encoder are discarded. Consequently, without a task-specific decoder for learning relation embeddings, R-GCN cannot directly support tasks such as

link prediction. Even if an extra decoder is available, the encoder-decoder framework runs into another problem of repeated introduction of relation-specific parameters in both the encoder side (relation-specific weight matrices) and the decoder side (relation embeddings). As a result, the number of parameters increases.

To address the issue, we propose a novel model inspired by R-GCN [22], a GCN-based knowledge graph encoder framework which can learn entity embeddings and relation embeddings simultaneously by performing relation-specific transformations from head entity embeddings to tail entity embeddings, hence called TransGCN. In principle, any presumed transformation assumption from the subject to the object, such as translation assumption, rotation assumption, etc., can be exploited in the proposed framework. Take the translation assumption as an example, specifically in which translation operators acted by relations are resorted to connect entities in a KG. The basic idea of TransGCN is illustrated in Figure 1. In such a scenario, TransGCN first translates the embeddings of 1-degree neighbors of one center entity v_i with their specific relation embeddings. The resultant embeddings serve as the initial *embedding estimations* of the center entity v_i. Then a convolutional operation is performed over these initially *estimated* embeddings to derive a new embedding $\mathbf{v}_i'$ for each v_i, which encodes local structural information of the center entity. Similar to R-GCN, aggregated structural information and self-loop information of a node are combined for entity embedding updates. Moreover, we also define a novel relation embedding convolution process so that the entity and relation embeddings can be handled in a layer-based manner as GCNs do.

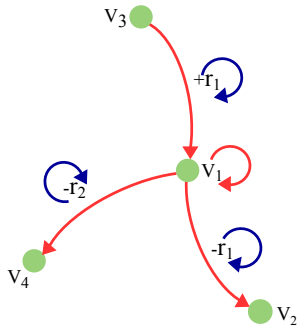


Figure 1: The basic idea of TransGCN. Red lines show updating rules of an entity v_1, where the neighborhood information is aggregated from neighbors with corresponding relations (+/- denotes incoming/outgoing relations) and then is combined with self-loop information (red self-loop arrow). Blue lines disclose the update phases of relations, which are achieved simply by transforming the previous relation embeddings.

The research contributions of our work are as follows:

(1) We propose that transformation assumptions in which relations are assumed as transformation operators transforming the subject entity to the object entity can be utilized to convert a heterogeneous neighborhood in a KG into a homogeneous neighborhood, which can be readily utilized within a GCN-based framework.
(2) We develop a novel GCN-based knowledge graph encoder framework called TransGCN which can encode entity and relation embeddings simultaneously. Compared with R-GCN, this method has less parameters and can be directly used for link prediction.
(3) Based on the transformation assumptions behind TransE and RotatE, respectively, we instantiate our GCN framework. Experimental results on *FB15K-237* and *WN18RR* show that two TransGCN models achieve substantial improvements against the state-of-the-art methods on both datasets.

The paper is structured as follows. In Section 2 we elaborate on the main idea of the TransGCN framework. Experimental details on *FB15K-237* and *WN18RR* are presented in Section 3. In Section 4, we introduce two branches of learning methods on graphs. One is the classic translation-based models and the other are GCN-based approaches. Section 5 concludes this work and suggests future research directions.

2 PROPOSED ARCHITECTURE

R-GCN model does not learn relation embeddings and thereby would not be directly utilizable for link prediction without a decoder. Moreover, R-GCN model repeatedly introduces relation-specific parameters in both the encoder side and the decoder side, which results in an increase in the number of parameters. We argue that the encoder alone for knowledge graph applications should encode entity and relation embeddings at the same time to reduce the number of parameters (thus helping alleviate the problem of overfitting) and thereby to improve training efficiency.

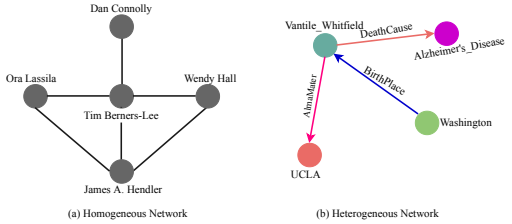
To address the issues, we propose a unified encoder framework based on GCN to learn entity and relation embeddings simultaneously, in which a presumed transformation assumption performed by relations is used to convert a heterogeneous neighborhood in a KG to a homogeneous one. This is subsequently used in a traditional GCN framework. Both entity embeddings and relation embeddings are learned in a convolutional layer-based manner. A knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes/entities and $\mathcal{E}$ is the set of labeled edges, contains statements in the form of a set of triples $(v_i, r_k, v_j) \in \mathcal{T}$, where v_i, r_k, and v_j represent the head entity, the relation, and the tail entity, respectively. In the following, we use the bold text to refer to embeddings and we will use (h, r, t) and (v_i, r_k, v_j) interchangeably.

2.1 Handling a Heterogeneous Neighborhood in a KG

Traditional GCNs [13] operate on an unlabeled undirected graph which consists of nodes of the same type and relations of the same type. This means that each edge has the same semantics and the neighborhood of a node is homogeneous. We call this a *homogeneous neighborhood*. Homogeneity makes it easier to aggregate the local neighborhood information around a node. For example, in an undirected unlabeled academic collaboration network shown in Figure 2(a), simply summing up information from *Wendy Hall*, *Dan Connolly*, *Ora Lassila* and *James A. Hendler* as messages transmitted to *Tim BernersLee* is reasonable. There is no need to consider the

differences in messages since their relations in such a graph are the same.

However, in a knowledge graph such as shown in 2(b), using such oversimplified summations would be problematic. Neighboring entities are linked to the center entity via different relations in different directions. For instance, the relation *DeathCause* is very different from the relation *BirthPlace* and their directions to *Vantile_Whitfield* are pointing in opposite directions. We call this a *heterogeneous neighborhood*. We argue that in order to make a KG be easily handled by a GCN-based framework, it is necessary to convert a heterogeneous neighborhood in a KG to a homogeneous one. In this work, we propose to approach this challenge by assuming relations in KGs are transformation operations which transform the head entity to the tail entity.



(a) Homogeneous Network (b) Heterogeneous Network

Figure 2: Neighbors in homogeneous and heterogeneous networks

Common ways of transformation between two entities include translation, rotation, reflection, and so on. In any such a transformation, a statement in KGs can be interpreted as that the head entity is transformed to the tail entity by a relation. More specifically, the tail entity in a statement may be the head entity after being translated/rotated. Accordingly, the embedding of a tail entity can be estimated by the head entity after a relation-specific transformation operation. For ease of generality, a statement $< h, r, t >$ following a transformation assumption can be written as:

$$\begin{cases} \mathbf{t} = \mathbf{h} \circ \mathbf{r} \\ \mathbf{h} = \mathbf{t} \star \mathbf{r} \end{cases} \tag{1}$$

where $\star$ and $\circ$ are defined as two transformation operators, which vary from different assumptions. We will specify them later in 2.3 and 2.4. The diversity of relation types and the direction of relations are two main characteristics of heterogeneity of heterogeneous graphs, e.g. KGs. Obviously, in this equation, the fact that each relation type is encoded differently takes care of the diversity of relation types and the transformation operators are usually specially designed to address the relation direction.

Based on the transformation assumption, we define the estimations of a central entity derived from connected entities with corresponding relations as the embeddings of neighbors of the entity. Take TransE as an example. Given an entity v_i with an outgoing triple (v_i, r_k, v_j), we define the estimation $(\mathbf{v_j} - \mathbf{r_k})$ of v_i based on (v_i, r_k, v_j) as the embedding of one neighbor of v_i. Similarly, for an incoming triple (v_l, r_m, v_i) of v_i, the embedding of another neighbor is $\mathbf{v_l} + \mathbf{r_m}$, which is another estimation of the central entity v_i. More concretely, in Figure 2(b), the estimations of the entity $v_{Vantile\_Whitfield}$ from incoming triples can be expressed as $\{\mathbf{v}_{Washington} + \mathbf{r}_{BirthPlace}\}$, while embedding estimations from outgoing triples can be expressed as $\{\mathbf{v}_{UCLA} - \mathbf{r}_{AlmaMater}, \mathbf{v}_{Alzheimer's\_Disease} - \mathbf{r}_{DeathCause}\}$.

Formally, under any transformation assumption, the embedding estimations of an entity v_i can be shown as follows:

$$\begin{aligned} \mathcal{T}(v_i) &= \mathcal{T}_{in}(v_i) \cup \mathcal{T}_{out}(v_i) \\ \mathcal{T}_{in}(v_i) &= \{(v_j, r_k, v_i) \mid \forall v_j, r_k \ (v_j, r_k, v_i) \in \mathcal{T}\} \\ \mathcal{N}_{in}(v_i) &= \{\mathbf{v_j} \circ \mathbf{r_k} \mid \forall (v_j, r_k, v_i) \in \mathcal{T}_{in}(v_i)\} \\ \mathcal{T}_{out}(v_i) &= \{(v_i, r_k, v_j) \mid \forall v_j, r_k \ (v_i, r_k, v_j) \in \mathcal{T}\} \\ \mathcal{N}_{out}(v_i) &= \{\mathbf{v_j} \star \mathbf{r_k} \mid \forall (v_i, r_k, v_j) \in \mathcal{T}_{out}(v_i)\} \end{aligned} \tag{2}$$

where $\mathcal{T}(v_i)$ denotes all the triples associated with v_i, consisting of $\mathcal{T}_{in}(v_i)$ as incoming triples and $\mathcal{T}_{out}(v_i)$ as outgoing triples, and $\mathcal{N}_{in}(v_i)$ and $\mathcal{N}_{out}(v_i)$ both are the sets of the estimated embeddings derived from incoming and outgoing neighbors, respectively.

After these transformation operations along different triple paths, the resultant estimated embeddings for the center entity should have the same semantics to the true center entity, by which the heterogeneous neighborhood in a KG is converted to a homogeneous one that can be easily handled by the GCN framework.

2.2 Model Formulation

Our model can be regarded as an extension of R-GCN [22]. In the following, we introduce how our model learns entity and relation embeddings at the same time. Like other existing GCN models [11, 22], our model can be formulated as a special case of Message Passing Neural Networks (MPNN) [5], which provide a general framework for supervised/semi-supervised learning on graphs.

In general, MPNN defines two phases: a message passing phase for nodes and a readout phase for the whole graph. Since in this paper we care about nodes and relations instead of the whole graph, we focus only on the message passing phase. Basically, this message passing phase of a node is executed L times to aggregate multi-hop neighborhood information and is composed of message passing functions $M^{(l)}$ and node update functions $U^{(l)}$, where l denotes the l-th hidden layer. $M^{(l)}$ mainly aggregates messages from local neighbors, while $U^{(l)}$ combines $M^{(l)}$ with self-loop information in the previous step. Both of these two functions are differentiable. In addition, Gilmer et al. [5] indicated that one could also learn edge features by introducing similar functions for all the edges in a graph, but so far only Kearnes et al. [11] have implemented this idea. To fit it into our task, we instantiate $M^{(l)}$ and $U^{(l)}$ for message propagation and entity embedding update for each entity v_i, and additionally introduce the update rule for a relation.

$$\begin{aligned} \mathbf{m}_i^{(l+1)} &= \sum_{(v_j, r_k, v_i) \in \mathcal{T}(v_i)} M^{(l)}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}, \mathbf{r}_k^{(l)}) \\ &= \frac{1}{c_i} \mathbf{W}_0^{(l)} \Big(\sum_{(v_j, r_k, v_i) \in \mathcal{T}_{in}(v_i)} (\mathbf{v}_j^{(l)} \circ \mathbf{r}_k^{(l)}) \\ &+ \sum_{(v_i, r_k, v_j) \in \mathcal{T}_{out}(v_i)} (\mathbf{v}_j^{(l)} \star \mathbf{r}_k^{(l)}) \Big) \end{aligned} \tag{3}$$

$$\mathbf{v}_i^{(l+1)} = U^{(l)}(\mathbf{m}_i^{(l+1)}, \mathbf{v}_i^{(l)}) = \sigma(\mathbf{m}_i^{(l+1)} + \mathbf{v}_i^{(l)}) \tag{4}$$

where $\mathbf{v}_i^{(l)} \in \mathbb{C}^{d^{(l)}}$ denotes the hidden representation of entity v_i in the l-th layer with a dimensionality of $d^{(l)}$. $\mathbf{W}_0^{(l)} \in \mathbb{C}^{d^{(l+1)} \times d^{(l)}}$ is a layer-specific matrix. $\mathcal{T}_{in}$ and $\mathcal{T}_{out}$ are defined in Eq. 2. c_i is an

entity-related normalization constant that could be the total degree of v_i. σ is the activation function, e.g., *ReLU*.

Basically, there are two terms in Eq. 3 that are used to encode local structural information for entity update representing messages from incoming relations and outgoing relations, respectively. The messages from incoming/outgoing relations are first accumulated by an element-wise summation and then are passed through a linear transformation. Then in the next step (Eq. 4), these messages are combined with self-loop information by simply adding them up to update entities. This idea is inspired by the skip-connections in ResNet [8] so that our model can perform at least as well as the simple transformation-based model instantiated in this framework. Figure 3 illustrates the computation graph for an entity. Typically, Eq. 3 considers the first-order neighbors of entities. One could simply stack multiple layers to allow for multi-hop neighbors. In

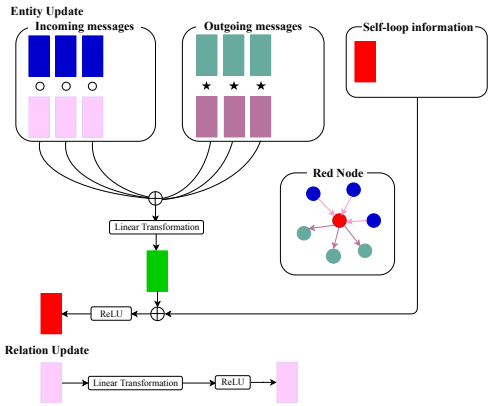


Figure 3: Diagram for the update of entity/node and relation/edge in the proposed TransGCN model.

addition, we realize that every update of entity embeddings in Eq. 3 and Eq. 4 may transform the original vector space. Consequently, the relationships between relations and entities would be affected, which makes it impossible to perform presumed transformation operations between them in the next layer. To address this problem, instead of applying a similar message passing mechanism for relations as in Eq. 3 and Eq. 4, for ease of efficiency, we introduce a transformation matrix $\mathbf{W}_1^{(l)}$ operated on relation embeddings for each layer. We assume that the introduced matrix can project relation embeddings into a vector space that has the same relation to the new entity vector space as they have before. Note that this is a soft restriction on the vector space; one could choose other more strict restrictions as well. For example, enforce constraints on the basis vectors of entities and relations so that these two vector spaces are ensured to be the same. Following the soft restriction, the update rule of relations in each layer is formed as follows:

$$\mathbf{r}_k^{(l+1)} = \sigma(\mathbb{W}_1^{(l)}\mathbf{r}_k^{(l)}) \tag{5}$$

where $\mathbf{r}_k^{(l)} \in \mathbb{C}^{d^{(l)}}$ is the hidden state of relation r_k in the l-th layer with a dimensionality of $d^{(l)}$. $\mathbf{W}_1^{(l)} \in \mathbb{C}^{d^{(l+1)}\times d^{(l)}}$ is a linear transformation across relations in the l-th layer.

In the following subsections, we instantiate our TransGCN framework by using two different transformation assumptions. One is based on the translation assumption and TransE is selected owe to its simplicity and popularity, while the other follows the rotation assumption and RotatE is chosen to achieve this assumption.

2.3 TransE-GCN model

Under the translation assumption, the relation is assumed to serve as a translation from the head entity to the tail entity. For an entity v_i, Eq. 1 can be instantiated as follows:

$$\mathbf{v}_i^{(l)} = \begin{cases} \mathbf{v}_j^{(l)} + \mathbf{r}_k^{(l)}, & (v_j, r_k, v_i) \in \mathcal{T}_{in}(v_i) \\ \mathbf{v}_j^{(l)} - \mathbf{r}_k^{(l)}, & (v_i, r_k, v_j) \in \mathcal{T}_{out}(v_i) \end{cases} \tag{6}$$

where $\circ$ and $\star$ are $+$ and $-$, respectively.

Like in TransE, the score function for a triple (v_i, r_k, v_j) is defined according to:

$$f_{r_k}(v_i, v_j) = -\left\|\mathbf{v}_i^{(l)} + \mathbf{r}_k^{(l)} - \mathbf{v}_j^{(l)}\right\| \tag{7}$$

where $\mathbf{v}_i^{(l)}$, $\mathbf{r}_k^{(l)}$ and $\mathbf{v}_j^{(l)}$ are the embeddings of v_i, r_k and v_j in the last layer, respectively.

Similar to previous studies [28], this model is trained with negative sampling. For each existing triple in a KG, a certain number of negative samples (e.g., one positive triple with 10 negative samples) are constructed by replacing either the head entity or the tail entity randomly. Positive samples are expected to have high scores while negative samples are expected to have low scores. A margin-based ranking function is written as the loss function for training:

$$\mathcal{L} = \sum_{(v_i, r_k, v_j)\in\mathcal{T}} \sum_{(v_i', r_k, v_j')\in\mathcal{T}'} \max(0, -f_{r_k}(v_i, v_j) + f_{r_k}(v_i', v_j') + \gamma) \tag{8}$$

where $\max(a, b)$ is used to obtain the maximum between a and b, γ is the margin, $\mathcal{T}$ is the set of observed triples in a KG, and $\mathcal{T}'$ is the set of negative samples associated with the positive sample (v_i, r_k, v_j). It is noteworthy that in this implementation, all the embeddings are in the real vector space.

2.4 RotatE-GCN model

Another assumption recently explored in knowledge graph embedding is rotation. Sun et al. assumed that the tail entity is derived from the head entity after being rotated performed by a relation in the complex vector space [24]. Accordingly, we can formalize the neighbors of an entity v_i:

$$\mathbf{v}_i^{(l)} = \begin{cases} \mathbf{v}_j^{(l)} \odot \mathbf{r}_k^{(l)}, & (v_j, r_k, v_i) \in \mathcal{T}_{in}(v_i) \\ \mathbf{v}_j^{(l)} \oplus \mathbf{r}_k^{(l)} = \mathbf{v}_j^{(l)} \odot \bar{r}_k^{(l)}, & (v_i, r_k, v_j) \in \mathcal{T}_{out}(v_i) \end{cases} \tag{9}$$

where $\circ$ and $\star$ are $\odot$ and $\oplus$, respectively. More specifically, $\odot$ is the element-wise product in the complex space and $\bar{r}_k$ is the complex conjugate of r_k. $\mid r_i \mid = 1$. Note that here the existence of r_k and $\bar{r}_k$ rather than different transformation operators guarantees the relation direction is considered naturally.

Similarly, the distance function serves as the score function:

$$f_{r_k}(v_i, v_j) = -\left\|\mathbf{v}_i^{(l)} \odot \mathbf{r}_k^{(l)} - \mathbf{v}_j^{(l)}\right\| \tag{10}$$

Table 1: Basic statistics of *FB15K-237* and *WN18RR*.

Dataset	FB15k-237	WN18RR
Entities	14,541	40,943
Relations	237	11
Training triples	272,115	86,835
Validation triples	17,535	3,034
Test triples	20,466	3,134

To keep consistent with RotatE, we adopt self-adversarial negative sampling to train the model rather than vanilla negative sampling. The main argument of self-adversarial negative sampling is that negative triples should have different probabilities of being drawn as training continues, e.g. many triples may be obviously false, thus not contributing any meaningful information. Therefore, a probability distribution p is used to draw negative samples according to the current embedding model.

$$p(v_i', r_k, v_j'|(v_i, r_k, v_j)) = \frac{exp(\alpha f_{r_k}(v_i', v_j'))}{\sum_{(v_i'', r_k, v_j'')\in\mathcal{T}'} exp(\alpha f_{r_k}(v_i'', v_j''))} \quad (11)$$

where α is a constant which controls the temperature of sampling and σ is the sigmoid function.

Then the above probability of a negative sample is treated as the weight of the sample to help construct the loss function. For a positive sample (v_i, r_k, v_j), the loss function can be written as follows:

$$\mathcal{L} = -log\big(\sigma(\gamma + f_{r_k}(v_i, v_j))\big) - \sum_{(v_i', r_k, v_j')\in\mathcal{T}'} p(v_i', r_k, v_j') log\big(\sigma(-f_{r_k}(v_i', v_j') - \gamma)\big) \quad (12)$$

where all the embeddings are in the complex vector space.

3 EXPERIMENT

To test the performance of our models, we evaluate our TransGCN models on the task of link prediction on two datasets: *FB15K-237* and *WN18RR*.

3.1 Datasets

In previous studies, the performance of link prediction methods was commonly evaluated on two datasets, namely FB15K from Freebase and WN18 from WordNet. However, there are inverse triples in both training and testing data, resulting in methods showing better performance on these datasets by means of memorizing these affected triples rather than having a better ability of prediction. Therefore, we use the two filtered data sets: *FB15K-237* and *WN18RR*, proposed in [25] and [2], respectively, in which all the inverse triplet pairs were removed. These two datasets have been shown to be more challenging for models to perform link prediction [22]. Table 1 shows basic statistics for these two datasets.

3.2 Experiment Setup

Evaluation metrics. In the testing phase, for each triple, we replace the head entity with all other entities in current KG, and calculate scores for those replaced triples and the original triple using the scoring function specified in section 2. Since some of the replaced triples might also appear in either training, validation or test set, we then filter these triples out and produce a filtered ranking which we denote as the *filtered* setting. Then those triples are ranked in a descending order of scores and the rank of the correct triple in this ranking list is used for evaluation. The whole procedure is repeated while replacing the tail entity instead of the head entity. Following previous studies [1], we adopt Mean Reciprocal Rank (MRR) and Hits@k as evaluation metrics. We report filtered MRR scores as well as Hits at 1, 3, and 10 for the *filtered* setting. For all the metrics, higher values mean better performance.

Baselines. Six baselines ($TransE$, $DistMult$, $ComplEx$, $R-GCN$, $ConvE$ and $RotatE$) are selected for the evaluation. $TransE$ is a standard translation-based model, which is simple but performs well on most datasets. This model is wrapped in our TransE-GCN model to achieve the conversion from heterogeneous neighbors to homogeneous neighbors. $DistMult$, as a factorization model, also shows promising performance on standard datasets. Furthermore, our model is compared with $ComplEx$ [26], one powerful state-of-the-art model for link prediction, and $R-GCN$ [22], a strong baseline of modeling directed labeled graph. $ConvE$ uses a multi-layer convolutional network to model the iterations between entities and relations[2]. $RotatE$ is the most recent KGE model, which is built on the rotation assumption [24]. This model is exploited in our RotatE-GCN model to derive homogeneous neighbors.

Implementation details. To optimize our TransGCN models, we used the Adam optimizer [12] and fixed the learning rate $\lambda = 0.001$. The best parameters were selected when filtered MRR achieved the best performance on respective validation sets. First, for both models, the embeddings of entities and relations produced by these two base models, i.e. TransE and RotatE, were used to initialize the embeddings needed in our models. For TransE, the embeddings pretrained by Nguyen et. al in [20] were utilized. Then, the number of layers in GCN was selected by comparing the experimental results on validation set. Finally, $L = 1$ was the best choice for both datasets. For RotatE, we trained this model by using the implementation provided by the authors to gain initial embeddings of entities and relations. Then most of the hyperparameter values of RotatE remained unchanged except that we ignored the batch size, since in GCN the batch size is achieved by setting graph batch size, which we leave as default. The only tuned parameter was the number of layers $L \in \{1, 2\}$. Finally, the best parameter settings in our experiment are $L = 2$ on *FB15K-237* and $L = 1$ on *WN18RR*.

3.3 Results

Main Results. The results for both datasets are reported in Table 2. Results on the baseline models $DistMult$, $TransE$, $ComplEx$, $ConvE$, and $RotatE$ are taken from [24], and R-GCN's results are taken from [22].

In Table 2, one important observation is that our TransE-GCN model and RotatE-GCN model both outperformed their base models, i.e. TransE and RotatE, on both datasets in terms of all the metrics by noticeable margins, which demonstrates the effectiveness of our proposed framework. Besides, the improvements restate the significance of explicitly incorporating local structural information in knowledge graph embedding learning. Moreover, compared with all the other baselines, the RotatE-GCN model was consistently better while the TransE-GCN model performed differently on the two datasets. To be specific, TransE-GCN performed better than ComplEx on *FB15K-237* while worse on the other dataset. This can be interpreted by the difference between TransE and ComplEx that

Table 2: Prediction results of different models on *FB15K-237* and *WN18RR*

	FB15K-237				WN18RR			
	MRR(Filtered)	Hit@1	Hit@3	Hit@10	MRR(Filtered)	Hit@1	Hit@3	Hit@10
DistMult	0.241	0.155	0.263	0.43	0.39	0.44	0.49	0.447
TransE	0.294	-	-	0.465	0.226	-	-	0.501
TransE-GCN	0.315	0.229	0.324	0.477	0.233	0.203	0.338	0.508
ComplEx	0.247	0.158	0.275	0.428	0.44	0.41	0.46	0.51
R-GCN	0.248	0.153	0.258	0.417	-	-	-	-
ConvE	0.325	0.237	0.356	0.501	0.43	0.40	0.44	0.52
RotatE	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571
RotatE-GCN	**0.356**	**0.252**	**0.388**	**0.555**	**0.485**	**0.438**	**0.51**	**0.578**

TransE is not good at dealing with relation types except 1-to-1 relations, as pointed out by researchers before. During the training process, for each triple (h,r,t), TransE enforces $\mathbf{h}+\mathbf{r}$ to be as close as possible to $\mathbf{t}$, which would be problematic when dealing with 1-to-N, N-to-1, and N-to-N relations. For example, given a 1-to-N relation r, we have two triples (h, r, t_1) and (h, r, t_2). If $\mathbf{h}+\mathbf{r}=\mathbf{t}$ holds, $\mathbf{t}_1$ and $\mathbf{t}_2$ should have the same vector representations. To meet this requirement, $\mathbf{h}+\mathbf{r}$ is close to the center of all the positive tails $\mathbf{t}$ at the end of training instead of a particular tail (which may be the correct prediction). Therefore, the performance of TransE dropped extremely on *WN18RR*, where there are four times more entities but 20 times less relations than those in *FB15K-237*. The superior performance of RotatE-GCN model over TransE-GCN model indirectly showed the importance of a base model used in our framework.

Comparison with R-GCN. It is necessary to elaborate on the comparison between our models and the R-GCN model that inspired our work. The experimental results showed that our models (TransE-GCN, RotatE-GCN model) both consistently yielded better results with improvements of 10.8% and 6.7% in terms of $MRR(Filtered)$ on FB15K-237, respectively. We believe the improvements are attributed to two reasons. First, thanks to the idea of converting heterogeneous neighbors into homogeneous neighbors in KGs, proposed in this paper, it successfully captured both local structural information by considering entities and relations in the neighborhood and semantic information residing within transformation operators. Besides, by doing so, relations in a KG were just modeled once and simultaneously with entities, and relation-specific matrices in R-GCN being replaced by shared matrices potentially facilitated the encoding of more complex latent information. Thus, fewer parameters were needed to learn in our models, which helps alleviate the problem of overfitting. In total, our TransE-GCN model has $((B-1)\times L\times d^2+2\times B\times R\times L)$ fewer parameters than R-GCN in terms of basis decomposition regularization and $(2\times B\times R\times L\times(\frac{d}{B})^2-L\times d^2)$ fewer parameters in terms of block-diagonal decomposition, where B denotes the number of basis matrices, L denotes the number of layers, d denotes the dimension of a hidden layer, and R denotes the number of relations. As for our RotatE-GCN model, we followed the implementation proposed by [24]. They used real numbers to express complex numbers by treating the first half dimensions of entity embeddings as the real part and the last half as the imaginary part. Therefore, the dimensions of entities are doubled in the complex vector space. Finally, our RotatE-GCN model has $((B-5)\times L\times d^2+2\times B\times R\times L-E\times d)$ fewer parameters than R-GCN (basis decomposition) and $(2\times B\times R\times L\times(\frac{d}{B})^2-5\times L\times d^2-E\times d)$ fewer parameters than R-GCN (block-diagonal decomposition), respectively, in which E is the number of entities.

Performance on Entities of different degrees. Figure 4 depicts the performance of our models on FB15K-237 validation set as functions of the entity degree. It can be observed that in the beginning the performance of both models increased a lot with the increasing size of neighborhood, while after a threshold, it dropped significantly. We believe it showed that a few neighbors were only able to provide limited local structural information, thus leading to poor performance; by contrast, too many neighbors brought too much mixed information, which made models hard to optimize. In the future, more work should focus on how to deal with these two extreme conditions.

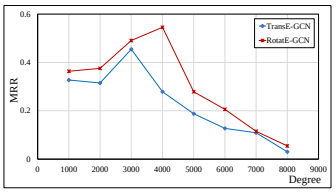


Figure 4: MRR for TransE-GCN and RotatE-GCN on FB15k-237 validation set with the entity degree

Table 3: Prediction results of our models on *FB15K-237* in terms of different hops

	MRR	Hit@10
TransE-GCN-1	0.315	0.474
TransE-GCN-2	0.297	0.453
TransE-GCN-3	0.273	0.421
RotatE-GCN-1	0.347	0.546
RotatE-GCN-2	0.356	0.555
RotatE-GCN-3	0.331	0.525

Analysis of multi-hop neighbors. Table 3 describes the prediction performance of our two models on *FB15K-237* in terms of multi-hop neighbors, namely 1-hop, 2-hop and 3-hop neighbors. TransE-GCN model favored 1-hop neighbors while RotatE was able to leverage more neighborhood information. The difference lies in that RotatE has a stronger ability to deal with complex relations and capture more accurate entity and relation information. But both models performed the worst when 3-hop neighbors were considered, which were even worse than the base models. We think this may be caused

by spectral convolutional filters, since it has been proven to have a smooth effect that could dilute the useful information[15].

4 RELATED WORK

Here we review previous work as it relates to our model.

4.1 Transformation-based Models

Until now, there exist two transformation assumptions in the literature - Translation and Rotation. A multitude of studies have explored these assumptions to achieve knowledge graph embedding learning for downstream tasks.

Translation-based models, also known as translational distance models, employed distance-based functions to model entities and relations in a KG. The key idea behind this kind of models is that for a positive triple (h, r, t), the head entity should be as close as possible to the tail entity through the relation, serving as a translation.

The most representative model is TransE [1] because of its simplicity and efficiency. It encodes the observed triples in a KG and projects entities and relations into the same vector space. TransE directly implemented the vanilla idea of translation, which enforces $\mathbf{h} + \mathbf{r} = \mathbf{t}$ when (h, r, t) holds. However, Wang et al. [28] argued that TransE cannot deal with N-to-1, 1-to-N and N-to-N relations and proposed a new model called TransH, which introduces a hyperplane H_r for each relation and requires that the projected head entity h' on H_r should be close to the projected tail entity t' on H_r after a translation r. TransR [17] follows a similar idea, but it introduced relation-specific translation spaces. In such a way, relations and entities can be represented in respective vector spaces. TransD [9] and TranSparse [10] are two other alternative approaches to simplifying TransR. In addition, another branch of improving TransE is to relax the strict restriction of $\mathbf{h} + \mathbf{r} = \mathbf{t}$, such as TransF [3]. For example, TransM assigned each triple with a relation-specific weight θ_r, and redefined the scoring function as $f_r(h, r) = -\theta_r \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$. For a comprehensive review of these methods, please refer to [27]. Our TransE-GCN model was based on this translation assumption and TransE from the first branch was exploited for carrying out translation operations.

Rotation assumption was recently exploited by Sun et al.[24]. Motivated by the Euler's identify that indicates a rotation in the complex plane can be achieved a unitary complex number, Sun et al. proposed a RotatE model, which projected both entities and relations into the complex vector space and treated each relation as a rotation from the head entity and the tail entity. The most attractive characteristics of RotatE is its ability to model and to infer multiple relation patterns, including symmetry/antisymmetry, inversion and composition. This model also adopted a distance-based score function to evaluate the compatibility of two entities and their relations, as shown in 10. Robust experimental results on benchmark datasets demonstrated the effectiveness of RotatE.

4.2 Graph Convolutional Networks

Our TransGCN framework is primarily motivated by plenty of works on modeling large-scale graph data using GCNs [13]. Generally, GCN can be classified into: (1) spectral-based approaches, which introduce spatial filters from the graph signal processing perspective [16, 23]; (2) spatial-based approaches, which simply interpret a graph convolutional operation as aggregating information from neighbors [4–6]. Although spectral-based methods seem appealing in that they can be supported by the spectral graph theory, in practice spatial-based methods perform better in terms of efficiency, generality and flexibility [29].

Interestingly, Kipf and Welling [13] discovered that when approximated by the 1^{st} order Chebyshev polynomials, the graph convolution is localized in space. That is, to some degree spatial-based approaches are the same as spectral-based approaches. Based on this, they introduced a simple but efficient message propagation rule conditioned on nodes and adjacency matrix of a graph for the semi-supervised node classification task.

To extend the GCN model [13] to directed labeled graphs[22] proposed an R-GCN model, which is the first work that applied the GCN framework to knowledge graphs for link prediction. The main contribution of this work lies in the introduction of relation-specific weight matrices in each layer of a neural network such that relation-specific messages can be propagated over graphs for entity update. The message propagation method for node v_i is defined as follows:

$$\mathbf{v}_i^{(l+1)} = \sigma\Big(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} \mathbf{v}_j^{(l)} + W_0^{(l)} \mathbf{v}_i^{(l)} \Big) \tag{13}$$

where $W_r^{(l)}$ denotes a relation-specific weight matrix in the l-th layer, $W_0^{(l)}$ another layer-specific weight matrix, $\mathcal{R}$ the set of relation types and $\mathcal{N}_i^r$ the set of neighbors of node v_i in terms of relation r.

To perform the task of link prediction, R-GCN, as an encoder, must cooperate with a decoder, such as DistMult. Although this method achieves promising performance in this task, there are some limitations. This R-GCN model alone cannot learn relation embeddings, which are very important for knowledge graph applications, since they only define message propagation strategies for node update. On the other hand, despite the fact that R-GCN with an extra decoder can learn relation embeddings for link prediction task, relation information is repeatedly incorporated in both encoder side and decoder side. As a result, the number of parameters increases. In this paper, we concentrated on solving these issues by finding a more reasonable way to extend traditional GCN to KGs.

5 CONCLUSION

In this paper we proposed a unified GCN framework (TransGCN) to learn embeddings of relations and entities simultaneously.To handle the heterogeneous characteristics of knowledge graphs when using traditional GCNs, we came up with a novel way of converting a heterogeneous neighborhood into a homogeneous neighborhood by introducing transformation assumptions, e.g., translation and rotation. Under these assumptions, a relation is treated as a transformation operator transforming a head entity to a tail entity. Translation and rotation assumptions were explored and TransE and RotatE model were wrapped in TransGCN framework, respectively. Any other transformation-based method could work as transformation operations. By doing so, nearby nodes with associated relations were aggregated as messages propagated to the center node as traditional GCNs did, which benefited the entity embedding learning. In addition, we explicitly encoded relations in the same GCN framework so

that relation embeddings can be seamlessly encoded with entities at the same time. In this sense, our TransGCN framework can be interpreted as a new (knowledge) graph encoder which produces both entity embeddings and relation embeddings. This encoder can be further incorporated into an encoder-decoder framework for any other tasks. Experimental results on two datasets - *FB15K-237* and *WN18RR* showed that our unified TransGCN models, both TransE-GCN and RotatE-GCN models consistently outperformed the baseline - R-GCN model by noticeably large margins in terms of all metrics, which demonstrated the effectiveness of the conversion idea in dealing with heterogeneous neighbors. Additionally, both models performed better than their base models, i.e., TransE and RotatE, showing the significance of explicitly modeling local structural information in knowledge graph embedding learning.

In this paper, although relations are encoded and learned in our GCN framework, they are updated simply by being passed through a separated linear transformation. In the future, we plan to explore approaches to directly operating convolutions on relations so that the local structure of graphs could also play a role in relation embedding learning. In addition, a weighting mechanism should be studied to measure unequal contributions of neighbors.

REFERENCES

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

[2] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*. 1811–1818. https://arxiv.org/abs/1707.01476

[3] Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. 2016. Knowledge graph embedding by flexible translation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

[4] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1416–1424.

[5] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1263–1272.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[7] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Daniel Jurafsky, and Jure Leskovec. 2018. Embedding Logical Queries on Knowledge Graphs. In *NeurIPS*.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 687–696.

[10] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[11] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design* 30, 8 (2016), 595–608.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[14] Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, David Heckerman, Chris Meek, et al. 2007. *Introduction to statistical relational learning*. MIT press.

[15] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[16] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[17] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379* (2015).

[18] Gengchen Mai, Bo Yan, Krzysztof Janowicz, and Rui Zhu. 2019. Relaxing Unanswerable Geographic Questions Using A Spatially Explicit Knowledge Graph Embedding Model. In *Proceedings of 22nd AGILE International Conference on Geographic Information Science, Jun. 17-10, Limassol, Cyprus*.

[19] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*.

[20] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 327–333.

[21] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.

[22] 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.

[23] David Shuman, Sunil Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Processing Magazine* 30 (2013), 83–98.

[24] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HkgEQnRqYQ

[25] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. 57–66.

[26] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.

[27] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[28] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

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

[30] Bo Yan, Matthew Walker, and Krzysztof Janowicz. 2019. A Time-Aware Inductive Representation Learning Strategy for Heterogeneous Graphs. (2019).

[31] 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).

[32] 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 (KDD '18)*. ACM, New York, NY, USA, 974–983. https://doi.org/10.1145/3219819.3219890