ReInceptionE: Relation-Aware Inception Network with Joint Local-Global Structural Information for Knowledge Graph Embedding

Zhiwen Xie[1], **Guangyou Zhou**[2*], **Jin Liu**[1*], **Jimmy Xiangji Huang**[3]
[1]School of Computer Science, Wuhan University
[2]School of Computer Science, Central China Normal University
[3]School of Information Technology, York University
xiezhiwen@whu.edu.cn, gyzhou@mail.ccnu.edu.cn
jinliu@whu.edu.cn, jhuang@yorku.ca

Abstract

The goal of Knowledge graph embedding (KGE) is to learn how to represent the low-dimensional vectors for entities and relations based on the observed triples. The conventional shallow models are limited to their expressiveness. ConvE (Dettmers et al., 2018) takes advantage of CNN and improves the expressive power with parameter efficient operators by increasing the interactions between head and relation embeddings. However, there is no structural information in the embedding space of ConvE, and the performance is still limited by the number of interactions. The recent KBGAT (Nathani et al., 2019) provides another way to learn embeddings by adaptively utilizing structural information. In this paper, we take the benefits of ConvE and KBGAT together and propose a **Re**lation-aware **Inception** network with joint local-global structural information for knowledge graph **E**mbedding (ReInceptionE). Specifically, we first explore the Inception network to learn query embedding, which aims to further increase the interactions between head and relation embeddings. Then, we propose to use a relation-aware attention mechanism to enrich the query embedding with the local neighborhood and global entity information. Experimental results on both WN18RR and FB15k-237 datasets demonstrate that ReInceptionE achieves competitive performance compared with state-of-the-art methods.

1 Introduction

Knowledge graphs (KGs) are at the core of most state-of-the-art natural language processing solutions and have been spotlighted in many real-world applications, including question answering (Hao et al., 2017), dialogue generation (He et al., 2017; Madotto et al., 2018) and machine reading comprehension (Yang and Mitchell, 2017). Typically, KGs are directed graphs whose nodes denote the entities and edges represent the different relations between entities. The structured knowledge in KGs is organized in the form of triples (h, r, t), where h and t stand for the head and tail entities respectively, and r represents the relation from h to t. Although large-scale KGs (e.g., Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2015)) have already contained millions or even billions of triples, they are still far from complete since the emerging new knowledge appears. Knowledge graph embedding (KGE) is an effective solution to solve the incompletion problem.

KGE aims to learn the low-dimensional vectors (embeddings) for entities and relations based on the observed triples in KGs. Conventional models including TransE (Bordes et al., 2013) and its numerous extensions (e.g., TransD (Ji et al., 2015), TransR (Lin et al., 2015), DistMul (Yang et al., 2015), ComplEx (Trouillon et al., 2016), etc.) have been proposed. These shallow models are limited to their expressiveness (Dettmers et al., 2018). Recently, CNN-based methods have been proposed to capture the expressive features with parameter efficient operators. ConvE (Dettmers et al., 2018) takes advantage of CNN and uses convolution filters on 2D reshapings of the head entity and relation embeddings. Through this, ConvE can increase the interactions between head and relation embeddings. Empirical results have proved that increasing the number of interactions is beneficial to the KGE task, but ConvE is still limited by the number of interactions (Jiang et al., 2019; Vashishth et al., 2020).

Furthermore, ConvE does not consider the structural information. In contrast, graph-based methods are effective to aggregate neighborhood information to enrich the entity/relation representation (Schlichtkrull et al., 2018; Bansal et al., 2019; Nathani et al., 2019). Among them, KB-

[*] Corresponding author.

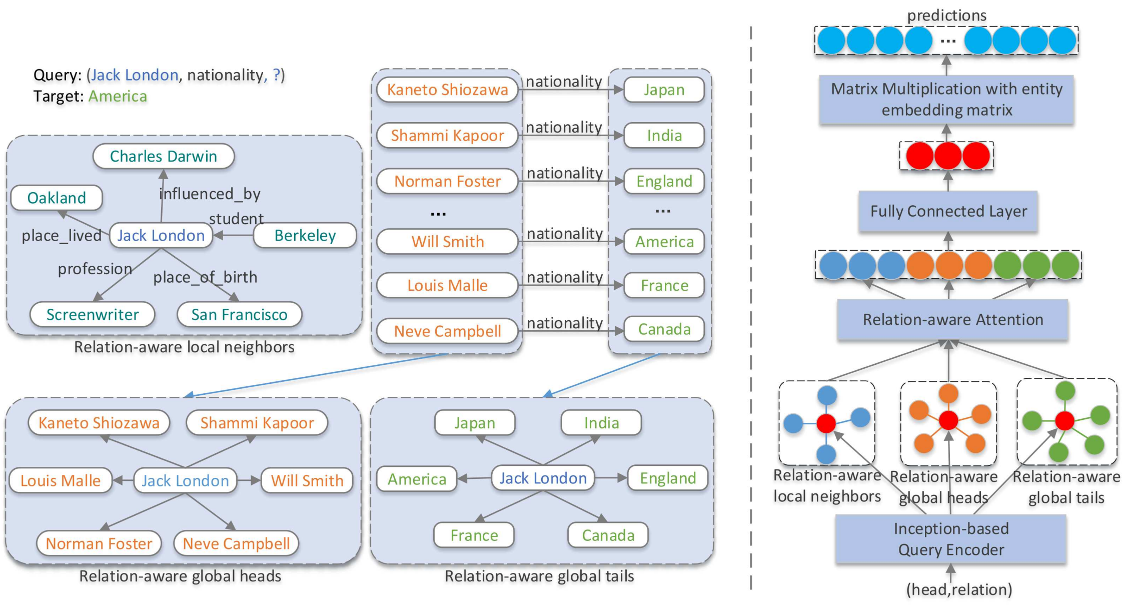

Figure 1: An example of relation-aware local and global information (left) and the general framework of our proposed ReInceptionE (right).

GAT (Nathani et al., 2019) achieves state-of-the-art performance on various benchmark datasets via using graph attention networks (GAT) (Velickovic et al., 2018). KBGAT learns embeddings for every entity by taking all possible relations into account, which requires multiple hops of reasoning. In contrast, it can be beneficial to learn embeddings from a query-relevant subgraph of the local neighborhood and global entities. As an example shown in Figure 1, given a query (*Jack London*, *nationality*, *?*) for *Jack London*, we can gather the relation-aware local neighbor (*place_lived*, *Okaland*). The local neighbor allows us to project *Jack London* into the *Okaland* region of the embedding space, which can lead to a high score for predicting the target *America*, as *Okaland* and *America* are close in embedding space. Besides, we also note that a specific relation can be acted as the "bridge" to link the related entities. Considering the relation *nationality*, the related head entities $\{$ *Kaneto Shiozawa*, *Shammi Kapoor*, *Will Smith*, $\cdots$ $\}$ and tail entities $\{$ *America*, *Canada*, *Japan*, $\cdots$ $\}$ tend to be a set of **person names** and **countries**. These related entities act as a strong signal to judge whether a triple is valid or not.

Based on the above observations, we take the benefits of ConvE and KBGAT together and propose a **Re**lation-aware **Inception** network with joint local-global structural information for knowledge graph **E**mbedding, and we name it **ReInceptionE**. In ReInceptionE, we first adapt Inception network (Szegedy et al., 2015, 2016) – a high performing convolutional neural network with carefully designed filters, to increase the interactions using multiple convolution filters with different scales, while at the same time to keep parameter efficient. Then, we construct a local neighborhood graph and a global entity graph by sharing the head and relation respectively for a given query. With the constructed graphs, we apply a relation-aware attention mechanism to aggregate the local neighborhood features and gather the global entity information to enrich the head/relation representation. Finally, we aggregate the joint local-global structural information using a fully connected layer to predict the missing links.

In summary, we make the following three contributions: (1) It is the first to explore Inception network to learn query embedding which aims to further increase the interactions between head and relation embeddings; (2) We propose to use a relation-aware attention mechanism to enrich the query embedding with the local neighborhood and global entity information; (3) We conduct a series of experiments to evaluate the performance of the proposed method. Experimental results demonstrate that our method obtains competitive performance in comparison to these state-of-the-art models on both WN18RR and FB15k-237.

The rest of this paper is structured as follows.

Section 2 describes our proposed method for KGE. In Section 3, the experimental results are presented. We make a conclusion in Section 4.

2 Our Approach

In this section, we first describe the background and definition in Subsection 2.1, and Inception-based query encoder in Subsection 2.2. Then, we introduce the relation-aware local attention and global attention in Subsection 2.3 and 2.4, respectively. Finally, we describe the joint using of them in Subsection 2.5.

2.1 Background and Definition

Definition 3.1 **Knowledge Graph** $\mathcal{G}$: A knowledge graph $\mathcal{G} = \{(h, r, t)|(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ denotes a collection of triples, where $\mathcal{E}$ and $\mathcal{R}$ indicate entities and relations, respectively, $h, t \in \mathcal{E}$ represent the head entity and tail entity, and $r \in \mathcal{R}$ denotes the specific relation linking from the head entity h to tail entity t.

Definition 3.2 **Knowledge Graph Embedding**: Knowledge graph embedding aims to learn embeddings of entities and relations with the valid triples in $\mathcal{G}$, and then predict the missing head entity h given query $(?, r, t)$ or tail entity t given query $(h, r, ?)$ with the learned entity and relation embeddings.

The framework of the proposed ReInceptionE is shown in Figure 1 (right). ReIncetionE consists of four modules: (1) Inception-based query encoder (InceptionE), which is used to transform the input query $q = (h, r, ?)$ into a k-dimensional vector $\mathbf{v}_q$; (2) relation-aware local attention and (3) relation-aware global attention are used to capture the local neighborhood information and the global entity information; and (4) joint relation-aware attention is used to aggregate the different structural information using a fully connected layer. Finally, we compute the score for the given triple (h, r, t) based on the query embedding and the tail entity embedding.

2.2 Inception-Based Query Encoder

ConvE (Dettmers et al., 2018) is the first model to apply CNN for KGE, which uses 2D convolution operation to model the head and relation in a query. However, ConvE is limited by the number of interactions between the head and relation embeddings (Jiang et al., 2019; Vashishth et al., 2020). In this paper, we propose to employ the Inception network

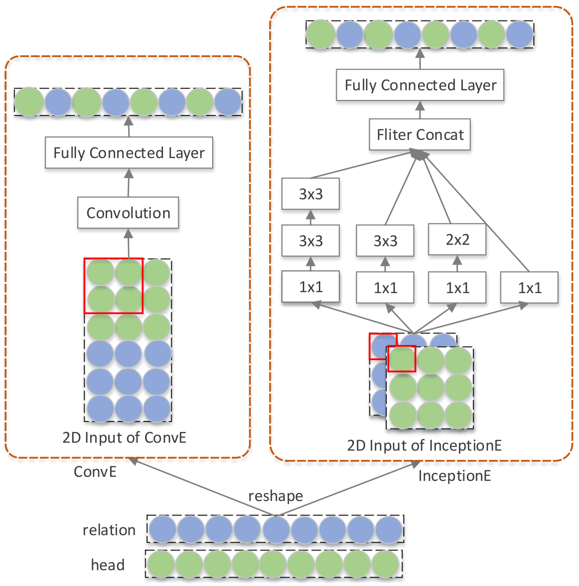

Figure 2: The structures of ConvE (left) and the proposed Inception-based query encoder (right). The red squares denote the slide windows of convolution filters.

(Szegedy et al., 2015, 2016), a high performing convolutional neural network with carefully designed filters, to increase the interactions by taking the head and relation as two channels of the input. Figure 2 shows the differences between InceptionE (right) and ConvE (left). Obviously, ConvE cannot capture full interactions between the head and relation embeddings since the convolution operations in ConvE only slides on the entity or relation 2D matrices independently. On the contrary, InceptionE can increase the interactions between the head and relation embeddings using multiple convolution filters with different scales, while at the same time keep parameter efficient.

As shown in Figure 2, given a query $q = (h, r, ?)$, we first reshape the head and relation embeddings as 2D matrices denoted as $\mathbf{v}_h$ and $\mathbf{v}_r$. Then, the 2D embeddings are viewed as two channels of the input for the Inception network. Thus, the entries at the same dimension of $\mathbf{v}_h$ and $\mathbf{v}_r$ are aligned over the channel dimension, which enables the convolution operations to increase the interactions between the head and relation embeddings. Specifically, We first use 1×1 convolutions to capture the direct interactions at the same dimension, which can be formulated as:

$$\mathbf{v}_{1\times1} = \text{Relu}([\mathbf{v}_h||\mathbf{v}_r] * \omega_{1\times1}) \tag{1}$$

where Relu (Glorot et al., 2011) is a non-linear activation function, $||$ denotes the concatenation operation, $*$ denotes the convolutional operation and

$\omega_{1\times1}$ is the parameter of convolution filters with 1×1 size, $\mathbf{v}_{1\times1}$ denotes the interaction features of the first 1×1 convolutional layer.

Then, filters with different sizes, such as 2×2 and 3×3, are applied to capture high-level interaction features in various scales. Thus, we can get interaction features of the 2×2 and 3×3 convolutional layers, denoted by $\mathbf{v}_{2\times2}$ and $\mathbf{v}_{3\times3}$, respectively.

As suggested in (Szegedy et al., 2016), we use two 3×3 convolutions instead of a 5×5 convolution to capture interaction features in larger spatial filters, which is able to reduce the number of parameters. The two 3×3 convolutions are denoted as:

$$\mathbf{v}_{2(3\times3)} = \text{Relu}(\text{Relu}(\mathbf{v}_{1\times1}^{2(3\times3)} * \omega_{3\times3}^{1}) * \omega_{3\times3}^{2}) \quad (2)$$

where $\mathbf{v}_{1\times1}^{2(3\times3)}$ is the input interaction features, $\omega_{3\times3}^{1}$ and $\omega_{3\times3}^{2}$ are parameters of the two 3×3 convolution layers.

Finally, the output interaction features with different scales and levels are concatenated and a fully connected layer is applied to obtain the embedding of the given query. Formally, we define the Inception-based query encoder model as:

$$\begin{aligned}\mathbf{v}_q &= \mathit{Inception}(\mathbf{v}_h, \mathbf{v}_r) \\ &= \text{Relu}(\text{vec}([\mathbf{v}_{1\times1}||\mathbf{v}_{2\times2}||\mathbf{v}_{3\times3}||\mathbf{v}_{2(3\times3)}])\mathbf{W})\end{aligned} \quad (3)$$

where $\mathbf{W}$ is the parameter of the fully connected layer.

2.3 Relation-Aware Local Attention

KBGAT learns embedding for every entity by taking all possible relations into account, and the embedding learning is impaired by the irrelevant neighbors. In contrast, it can be beneficial to learn embedding from a query-relevant neighborhood graph. In this subsection, we first construct a relation-aware neighborhood graph and then apply an attention mechanism to aggregate local graph structure information.

For the query $q = (h, r, ?)$, we denote its neighbors as $\mathcal{N}_q = \{n_i = (e_i, r_i) | (e_i, r_i, h) \in \mathcal{G}\}$. Note that, for each triple (h, r, t), we create an inverse triple (t, r^{-1}, h), which has also been used in (Lacroix et al., 2018; Dettmers et al., 2018). Thus, query $(?, r, t)$ can be converted to $(t, r^{-1}, ?)$. And the neighbors $\{(r_j, e_j) | (h, r_j, e_j) \in \mathcal{G}\}$ for head entity h can be converted to a format of $\{(e_j, r_j^{-1}) | (h, r_j, e_j) \in \mathcal{G}\}$. Thus, $\mathcal{N}_q$ contains both the outgoing and incoming neighbors for a query $q = (h, r, ?)$.

Each neighbor $n_i = (e_i, r_i) \in \mathcal{N}_q$ is also a query with a head entity e_i and a relation r_i. Thus, each entity and relation in neighbor $n_i = (e_i, r_i)$ can be encoded using the Inception-based query encoder:

$$\mathbf{v}_{n_i} = \mathit{Inception}(\mathbf{v}_{e_i}, \mathbf{v}_{r_i}) \quad (4)$$

where $\mathbf{v}_{e_i}$ and $\mathbf{v}_{r_i}$ are the 2D embedding vectors of entity e_i and relation r_i.

In practice, different neighbors may have different impacts for a given query. It is useful to determine the importance of each neighbor for a specific query. As an example in Figure 1, for the query (*Jack London*, *nationality*, *?*), it is reasonable to focus on the the neighbors related to the relation *nationality*, such as (*Jack London*, *place_lived*, *Oakland*). To this end, we use relation-aware attention mechanism to assign different importance for each neighbor and compute the relevant score for each neighbor using a non-linear activation layer:

$$s_i = \text{LeakyRelu}(\mathbf{W}_1[\mathbf{W}_2\mathbf{v}_q||\mathbf{W}_3\mathbf{v}_{n_i}]) \quad (5)$$

where $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$ are parameters to be trained and LeakyRelu (Maas et al., 2013) is the activation function.

We then normalize the relevant scores for different neighbors using a softmax function to make it comparable across the neighbors, which is denoted as:

$$\alpha_i = \frac{\exp(s_i)}{\sum_{n_j \in \mathcal{N}_q} \exp(s_j)} \quad (6)$$

Finally, we aggregate the neighborhood information according to their attention scores and apply a non-linear function to obtain the neighborhood vector. To keep more information of the original query embedding, we also apply a residual operation:

$$\mathbf{v}_n = \text{Relu}\left(\sum_{n_i \in \mathcal{N}_q} \alpha_i \mathbf{W}_3 \mathbf{v}_{n_i}\right) + \mathbf{W}_2\mathbf{v}_q \quad (7)$$

For simplification, we denote the above relation-aware attention operations as:

$$\mathbf{v}_n = \mathit{ReAtt}(\mathbf{V}_n, \mathbf{v}_q) \quad (8)$$

where $\mathbf{V}_n = \{\mathbf{v}_{n_i} | n_i \in \mathcal{N}_q\}$ is a set of local neighobrhood vectors.

2.4 Relation-Aware Global Attention

The number of relation-aware local neighbors for each node (entity) varies from one to another, making the neighbor graph very sparse. The sparse nature would affect the accuracy of the embedding. In fact, a specific relation can be acted as the "bridge" to link the related entities. In this subsection, we construct a relation-aware head graph and tail graph by gathering all entities for relation r in the given query $q = (h, r, ?)$. Intuitively, all head entities for relation r share some common type information. And the tail entities for relation r contain some implicit information about the type of the target entity t. For example in Figure 1, given the relation *nationality*, all heads { *Kaneto Shiozawa*, *Shammi Kapoor*, *Will Smith*, $\cdots$, } and tails { *America*, *Canada*, *Japan*, $\cdots$, } are the names of a person and a country, sharing the similar entity types. These relation-aware global heads and tails can provide some useful information for the KGE task. Thus, we construct relation-aware global head and tail graphs according to the head and tail entities of the relation.

Let $\mathcal{H}_r = \{e_i | (e_i, r, e_j) \in \mathcal{G}\}$ and $\mathcal{T}_r = \{e_j | (e_i, r, e_j) \in \mathcal{G}\}$ denote a set of head and tail entities for relation r, respectively. For each head entity $h_{ri} \in \mathcal{H}_r$, we first represent it as an embedding vector $\mathbf{v}_{h_{ri}}$. Then, we use relation-aware attention mechanism to capture the relevant information from all the relation-aware head entities, which is denoted as:

$$\mathbf{v}_{rh} = ReAtt(\mathbf{V}_{rh}, \mathbf{v}_q) \tag{9}$$

where $\mathbf{V}_{rh} = \{\mathbf{v}_{h_{ri}} | h_{ri} \in \mathcal{H}_r\}$ is a set of entity vectors for relation-aware global entities.

Similarly, we use relation-aware attention mechanism to capture global tail informations, which is computed as:

$$\mathbf{v}_{rt} = ReAtt(\mathbf{V}_{rt}, \mathbf{v}_q) \tag{10}$$

where $\mathbf{V}_{rt} = \{\mathbf{v}_{t_{ri}} | t_{ri} \in \mathcal{T}_r\}$ is a set of entity embeddings for relation-aware global tails.

2.5 Joint Relation-Aware Attention

Once obtained the relation-aware local neighborhood information $\mathbf{v}_n$ and global head and tail vectors $\mathbf{v}_{ht}$ and $\mathbf{v}_{rt}$, we concatenate these vectors and merge them by using a linear feed-forward layer:

$$\mathbf{v}'_q = \mathbf{W}_4[\mathbf{v}_n || \mathbf{v}_{rh} || \mathbf{v}_{rt}] + \mathbf{b} \tag{11}$$

Dataset	WN18RR	FB15k-237
#Entities	40,943	14,541
#Relations	11	237
#Training	86,835	141,442
#Validation	3,034	17,535
#Test	3,134	20,466

Table 1: Statistics of the datasets.

where $\mathbf{W}_4$ and $\mathbf{b}$ are the parameters of the feed-forward layer.

Finally, we compute the score for each triple (h, r, t) by applying a dot product of the query embedding $\mathbf{v}'_q$ and the tail embedding $\mathbf{v}_t$:

$$f(h, r, t) = \mathbf{v}'^T_q \mathbf{v}_t \tag{12}$$

To optimize the parameters in our model, we compute the probability of the tail t using a softmax function:

$$p(t|h, r) = \frac{\exp(\lambda f(h, r, t))}{\sum_{(h,r,t') \in \mathcal{G}' \cup \{(h,r,t)\}} \exp(\lambda f(h, r, t'))} \tag{13}$$

where λ is a smoothing parameter, and $\mathcal{G}'$ is a set of invalid triples created by randomly replacing the tail t with an invalid entity t'.

We train the model by minimizing the following loss function:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|} \sum_{i=0}^{|\mathcal{E}|} \log p(t_i | h_i, r_i) \tag{14}$$

where $(h_i, r_i, t_i) \in \mathcal{G}$ is a valid triple, and $|\mathcal{E}|$ is the number of valid triples in $\mathcal{G}$.

3 Experiments

3.1 Experimental Setup

Datasets: We conduct experiments for KGE on two widely used public benchmark datasets : WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015). WN18RR is a subset of WN18 (Bordes et al., 2013) while FB15k-237 is a subset of FB15k (Bordes et al., 2013). Since WN18 and FB15k contain a large number of inverse relations, making the triples in the test set can be obtained simply by inverting triples in the training set. To address the above problem, both WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015) are generated by removing the inverse relations from WN18 and FB15k. In recent two

Models	WN18RR			FB15k-237		
	MR	MRR	Hits@10	MR	MRR	Hits@10
TransE (Bordes et al., 2013)*	2300	0.243	0.532	323	0.279	0.441
DistMult (Yang et al., 2015)*	5110	0.430	0.490	512	0.281	0.446
ComplEx (Trouillon et al., 2016)*	5261	0.440	0.510	546	0.278	0.450
R-GCN+ (Schlichtkrull et al., 2018)	-	-	-	-	0.249	0.417
CACL (Oh et al., 2018)	3154	0.472	0.543	235	0.349	0.487
ConvE (Dettmers et al., 2018)	4187	0.430	0.520	244	0.325	0.501
NKGE (Wang et al., 2018)	4170	0.450	0.526	237	0.330	0.510
TransMS (Yang et al., 2019)	6523	-	0.460	249	-	0.445
AnyBURL (Meilicke et al., 2019)	-	0.470	0.552	-	0.310	0.486
SACN (Shang et al., 2019)	-	0.470	0.540	-	0.350	**0.540**
A2N (Bansal et al., 2019)	-	0.450	0.510	-	0.317	0.486
GRank (Ebisu and Ichise, 2019)	-	0.470	0.539	-	0.322	0.489
ConvR (Jiang et al., 2019)	-	0.475	0.537	-	0.350	0.528
MuRE (Balazevic et al., 2019b)	-	0.475	0.554	-	0.336	0.521
RotatE (Sun et al., 2019)	3340	0.476	0.571	177	0.338	0.533
QuatE (Zhang et al., 2019)	3472	0.481	0.564	176	0.311	0.495
InteractE (Vashishth et al., 2020)	5202	0.463	0.528	**172**	**0.354**	0.535
ConvKB (Nguyen et al., 2018)^b	3433	0.249	0.524	309	0.243	0.421
CapsE (Nguyen et al., 2019)^b	**718**	0.415	0.559	403	0.150	0.356
KBGAT (Nathani et al., 2019)^b	1921	0.412	0.554	270	0.157	0.331
ReInceptionE (ours)	1894	**0.483**	**0.582**	173	0.349	0.528
ConvKB (Nguyen et al., 2018)^a	2554	0.248	0.525	257	0.396	0.517
CapsE (Nguyen et al., 2019)^a	719	0.415	0.560	303	0.523	0.593
KBGAT (Nathani et al., 2019)^a	1940	0.440	0.581	210	0.518	0.626

Table 2: Link prediction results on WN18RR and FB15k-237 test sets. * denotes that the results are taken from (Dettmers et al., 2018), the superscript a represents the results reported in the original papers while b represents the results are taken from (Sun et al., 2020), other results are directly taken from the corresponding papers. Both MRR and Hits@1 have a strong correlation, thus we do not report the results of Hits@1 since it does not give any new insight (Nguyen et al., 2019). The best results are in **bold** and the second best results are in underline.

years, WN18RR and FB15k-237 have become the most popular datasets for the KGE task. Table 1 shows the summary statistics of the datasets.

Implementations: For a test triple (h, r, t), the purpose of KGE task is to predict missing links, e.g. predict tail entity t given head entity h and relation r or predict head entity h given tail entity t and relation r. To evaluate our method, three metrics are used, including Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hit@10 (e.g. the accuracy in top 10 predictions). Please note that lower MR, higher MRR and Hits@10 indicate better performance. We follow the "Filtered" setting protocol (Bordes et al., 2013) to evaluate our model, i.e., ranking all the entities excluding the set of other true entities that appeared in training, validation and test sets. We initialize the embedding of entity and relation in our ReInceptionE model using the pre-trained embeddings with 100-dimension used in (Nguyen et al., 2019). We use Adam (Kingma and Ba, 2015) to optimize the model. The parameters of our model are selected via grid search according to the MRR on the validation set. We select the dropout rate from $\{0.1, 0.2, 0.4, 0.5\}$, the learning rate from $\{0.001, 0.0005, 0.0002, 0.0001\}$, the L_2 norm of parameters from $\{1e^{-3}, 1e^{-5}, 1e^{-8}\}$, the batch size from $\{32, 64, 128, 256, 512\}$ and the smoothing parameter λ in Equation 13 from $\{1, 5, 10\}$. Finally, the learning rate is set to 0.0002 for WN18RR and 0.0001 for FB15k-237. The L_2 norm of parameters is set to $1e^{-5}$. The batch size is set to 256. The dropout rate is set to 0.4 for WN18RR and 0.2 for FB15k-237. The smoothing parameter in Equation 13 is set to $\lambda = 5$. The number of filters for each convolution operation in the Inception module is set to 32. We

Models	WN18RR			FB15k-237		
	MR	MRR	Hits@10	MR	MRR	Hits@10
ConvE	4187	0.430	0.520	244	0.325	0.501
KBGAT	1921	0.412	0.554	270	0.157	0.331
InceptionE	2317	0.451	0.563	215	0.334	0.518
ReInceptionE w/o N	1942	0.449	0.573	185	0.348	0.525
ReInceptionE w/o E	**1809**	0.412	0.569	186	0.343	0.522
ReInceptionE	1894	**0.483**	**0.582**	**173**	**0.349**	**0.528**

Table 3: Impact of different modules contributes the KGE task.

observe that MRR performance increases slowly, starting to stagnate around 200 epochs. Finally, we train the model up to 200 epoches in the following experiments. The source codes are available at `https://github.com/JuneTse/ReInceptionE`.

3.2 Main Results

We compare our results with various state-of-the-art methods. Experimental results are summarized in Table 2. For all KGE models, a key step is to create the invalid triples to construct the negative samples. Most recently, Sun et al. (2020) investigated the inappropriate evaluation problem happened in ConvKB (Nguyen et al., 2018), CapsE (Nguyen et al., 2019) and KBGAT (Nathani et al., 2019). In fact, this issue comes from the unusual score distribution, e.g., the score function for some invalid triples gets the same values as the valid triples. Sun et al. (2020) also found that KBGAT removed the invalid triples when they appeared in the test set during negative sampling, suffering from the leakage of test triples. Therefore, we take the results (marked with the superscript b) from (Sun et al., 2020) for ConvKB, CapsE and KBGAT. Besides, we also list the results reported in the original papers (marked with the superscript a).

From Table 2, we can see that our proposed ReInceptionE obtains competitive results compared with the state-of-the-art methods. On WN18RR dataset, the ReInceptionE achieves the best results using Hits@10 and MRR, and the second-best results using MR. On FB15k-237 dataset, the ReInceptionE obtains the second-best results using MR, and comparable results using MRR and Hits@10.

Our proposed ReInceptionE is closely related to ConvE (Dettmers et al., 2018) and KBGAT (Nathani et al., 2019). Compared with ConvE, ReInceptionE achieves large performance gains on both WN18RR and FB15k-237 (ConvE vs. ReInceptionE). The reason is that instead of simply concatenating the head and relation embeddings, ReInceptionE takes head and relation as two channels of the input and applies the Inception network to capture the rich interactions, which is able to learn expressive features by using filters with various scales. Unlike KBGAT, the ReInceptionE takes the (entity, relation) pair as a query and utilizes the relation-aware attention mechanism to gather the most relevant local neighbors and global entity information for the given query. The results again verify the effectiveness of the relation-aware local and global information for KGE.

Some other methods have been proposed to address the KGE task, such as pLogicNet (Ou and Tang, 2019), RPJE (Niu et al., 2020), CoKE (Wang et al., 2019), TuckER (Balazevic et al., 2019a), D4-GUmbel (Xu and Li, 2019) and HAKE (Zhang et al., 2020). pLogicNet (Ou and Tang, 2019) and RPJE (Niu et al., 2020) leverage logic rules to improve the performance. CoKE (Wang et al., 2019) uses Transformer (Vaswani et al., 2017) to encode contextualized representations. HAKE (Zhang et al., 2020) embeds entities in the polar coordinate system to learn semantic hierarchies. D4-Gumbel (Xu and Li, 2019) uses the dihedral group to model relation composition. TuckER (Balazevic et al., 2019a) uses Tucker decomposition to learn tensor factorization for KGE. These methods take a series of different ways to model the KGE task. For example, logic rules play an important role to determine whether a triple is valid or not, we suspect that the performance of our proposed ReInceptionE can be further improved when taking the logic rules into account. We will leave the comparison and deep analysis in the future work.

3.3 Impact of Different Modules

We describe the experimental results in Table 3 to investigate the impact of different modules in ReInceptionE. In Table 3, "InceptionE" is the baseline

	Models	Predicting head				Predicting tail			
		1-1	1-N	N-1	N-N	1-1	1-N	N-1	N-N
WN18RR	ConvE	0.975	0.414	0.110	0.950	0.975	0.153	0.303	0.949
	InceptionE	**0.976**	**0.587**	0.128	0.957	0.952	0.231	0.482	0.957
	ReInceptionE	**0.976**	0.586	**0.152**	**0.961**	**0.976**	**0.272**	**0.494**	**0.958**
FB15k-237	ConvE	0.303	0.590	0.137	0.400	0.272	0.088	0.845	0.545
	InceptionE	0.573	0.624	0.175	0.452	0.557	0.124	0.865	0.557
	ReInceptionE	**0.609**	**0.651**	**0.185**	**0.473**	**0.594**	**0.149**	**0.872**	**0.603**

Table 4: Link prediction results for each relation category on the WN18RR and FB15k-237 test sets using Hits@10. Following (Bordes et al., 2013), we classify relations into four groups: one-to-one (1-1), one-to-many (1-N), many-to-one (N-1) and many-to-many (N-N).

Query and Target	Top Neighbors and Predictions
Query: (Jack London, nationality, ?) **Target**: America	**Top Neighbors**: (place_lived, Oakland) Prob: 0.415 (place_of_birth, San Francisco) Prob: 0.353 (Berkeley, student) Prob: 0.083 (influence_by, Friedrich Nietzsche) Prob: 0.042 (influence_by, Charles Darwin) Prob: 0.031 **Top Predictions**: America, United Kingdom, Canada, Australia, Germany
Query: (Jerry Lewis, languages, ?) **Target**: English Language	**Top Neighbors**: (place_of_birth, Newark) Prob: 0.197 (place_lived, Newark) Prob: 0.173 (Nutty Professor II, story_by) Prob: 0.105 (award_nomination, Razzie Award for Worst Actor) Prob: 0.089 (nominated_for, Law & Order: Special Victims Unit) Prob: 0.082 **Top Predictions**: English Language, Spanish Language, French Language, Italian Language, Japanese Language

Table 5: Two examples of top 5 attention neighbors and predictions for the given queries.

model without using relation-aware local neighbors and global entities. "ReInception w/o N" is the model without using relation-aware local neighbor information while "ReInception w/o E" is the model without using relation-aware global entity information. Besides, we also take two closely related models ConvE and KBGAT for fair comparison.

From Table 3, we can see that our baseline InceptionE outperforms the closely related CNN-based model ConvE. Compared with ConvE, InceptionE is more powerful because it can capture the rich interaction features by using filters with various scales. And the ReInceptionE, which incorporates relation-aware local neighborhood and global entity information, outperforms the related graph-based model KBGAT. Table 3 also shows that the ReInceptionE outperforms InceptionE, "ReInception w/o N" and "ReInception w/o E" by a large margin on both datasets, which reconfirms our observations that relation-aware local neighbors and global entities can play different contributions for KGE.

3.4 Evaluation on different Relation Types

In this subsection, we present the experimental results on different relation types on WN18RR and FB15k-237 using Hits@10. We choose the closely related model ConvE, as well as InceptionE as the baselines. Following (Bordes et al., 2013), we classify the relations into four groups: one-to-one (1-1), one-to-many (1-N), many-to-one (N-1) and many-to-many (N-N), based on the average number of tails per head and the average number of heads per tail. Table 4 shows the link prediction results for each relation category. From Table 4, we find that InceptionE achieves better performance than ConvE for all relation types, indicating that increasing the number of interactions between head and relation embeddings is indeed beneficial to KGE task. Furthermore, our proposed ReInceptionE significantly outperforms ConvE and InceptionE for all relation types. In particular, ReInceptionE obtains larger improvements for complex relations, such as one-to-many, many-to-one and many-to-many. This again verifies our observations that increasing the interactions and taking the local-global structural information allows the model to capture more complex relations.

3.5 Case Study

In order to further analyze how relation-aware neighbors contribute to KGE task, we give two examples in Table 5. For the query (*Jack London*, *nationality*, *?*), ReInceptionE assigns the highest attention scores for neighbors (*place_lived*, *Oakland*), since *Oakland* and *America* are close to each other in embedding space because of other relations between them. And the top predictions for the query are a set of entities with the type of country. For the second example (*Jerry Lewls*, *languages*, *?*), ReInceptionE assigns the very high score for neighbor (*place_of_birth*, *Newark*). This can allow us to project (*place_of_birth*, *Newark*) into the *Jerry Lewis* region of the embedding space, which can lead to a high score for predicting the target *English Language*. These examples give clear evidence of how our proposed ReInceptionE benefits the KGE task.

4 Conclusions

In this paper, we propose a novel relation-aware Inception network for knowledge graph embedding, called ReInceptionE. ReInceptionE takes the benefits of ConvE and KBGAT together. The proposed method first employs Inception network to learn the query embedding, with the aim of increasing the interaction between head and relation embeddings, while at the same time to keep the parameter efficient. Then, we gather the relation-aware local neighborhood and global entity information with an attention mechanism and enrich the query embedding with the joint local-global structural information. Empirical studies demonstrate that our proposed method obtains comparative performance compared with the state-of-the-art performance on two widely used benchmark datasets WN18RR and FB15k-237.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 61972290 and 61972173, and also supported by the National Key R&D Program of China under Grant 2018YFC1604000. This research was also supported in part by the research grant from Natural Sciences and Engineering Research Council (NSERC) of Canada and York Research Chairs (YRC) program. We thank anonymous reviewers for their thorough review comments on this paper.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019a. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the EMNLP-IJCNLP*.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019b. Multi-relational poincaré graph embeddings. In *Proceedings of the NeurIPS*.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the ACL*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the SIGMOD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the NIPS*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI*.

Takuma Ebisu and Ryutaro Ichise. 2019. Graph pattern entity ranking model for knowledge graph completion. In *Proceedings of the NAACL*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the AISTATS*.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the ACL*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the ACL*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the ACL*.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adative convolution for multi-relational learning. In *Proceedings of the NAACL-HLT*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the ICLR*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the ICML*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI*.

Maas, Andrew L, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2Seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. In *Proceedings of the ACL*.

Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the IJCAI*.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the ACL*.

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 NAACL*.

Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the NAACL*.

Guanglin Niu, Yongfei Zhang, Bo Li, Peng Cui, Si Liu, Jingyang Li, and Xiaowei Zhang. 2020. Rule-guided compositional representation learning on knowledge graphs. In *Proceedings of the AAAI*.

Byungkook Oh, Seungmin Seo, and Kyong-Ho Lee. 2018. Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. In *Proceedings of the CIKM*.

Meng Ou and Jian Tang. 2019. Probabilistic logic neural networks for reasoning. In *Proceedings of the NeurIPS*.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the ESWC*.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the ICLR*.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha P. Talukdar, and Yiming Yang. 2020. A re-evaluation of knowledge graph completion methods. In *Proceedings of the ACL*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the CVPR*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the CVPR*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the EMNLP*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of ICML*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the NIPS*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the ICLR (Poster)*.

Kai Wang, Yu Liu, Xiujuan Xu, and Dan Lin. 2018. Knowledge graph embedding with entity neighbors and deep memory network. *CoRR*, abs/1808.03752.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. CoKE: Contextualized knowledge graph embedding. *CoRR*, abs/1911.02168.

Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. In *Proceedings of the ACL*, pages 263–272.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the ACL*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*.

Shihui Yang, Jidong Tian, Honglun Zhang, Junchi Yan, Hao He, and Yaohui Jin. 2019. Transms: Knowledge graph embedding for complex relations by multidirectional semantics. In *Proceedings of the IJCAI*.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embedding. In *Proceedings of the NeurIPS*.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI*.