Knowledge Graph Embedding with Atrous Convolution and Residual Learning

Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, Yujia Bai
School of Computer Science and Engineering, Northeastern University
Shenyang, 110169, China
renfeiliang@cse.neu.edu.cn

Abstract

Knowledge graph embedding is an important task and it will benefit lots of downstream applications. Currently, deep neural networks based methods achieve state-of-the-art performance. However, most of these existing methods are very complex and need much time for training and inference. To address this issue, we propose a simple but effective atrous convolution based knowledge graph embedding method. Compared with existing state-of-the-art methods, our method has following main characteristics. First, it effectively increases feature interactions by using atrous convolutions. Second, to address the *original information forgotten* issue and vanishing/exploding gradient issue, it uses the residual learning method. Third, it has simpler structure but much higher parameter efficiency. We evaluate our method on six benchmark datasets with different evaluation metrics. Extensive experiments show that our model is very effective. On these diverse datasets, it achieves better results than the compared state-of-the-art methods on most of evaluation metrics. The source codes of our model could be found at https://github.com/neukg/AcrE.

1 Introduction

Knowledge graph is a kind of valuable knowledge bases and it is important for many AI-related applications. Generally, a KG stores factual knowledge in the form of structural triplets like <*h, r, t*>, which means there is a kind of *r* relation from *h* (head entity) to *t* (tail entity). Nowadays, great achievements have been made in building large scale KGs. Usually a KG may contain millions of entities and billions of relational facts. However, there are still two major difficulties that prohibit the availability of KGs. First, although most existing KGs contain large amount of triplets, they are far from completeness. Second, most existing KGs are stored in symbolic and logical formations while applications often involve numerical computing in continuous spaces. To address these two issues, researchers proposed *knowledge graph embedding* (KGE) methods that aim to learn a kind of embedding representations for a KG's items (entities and relations) by projecting these items into some continuous low-dimensional spaces. Generally, different kinds of KGE methods mainly differ in how to view the role of relations in the projected spaces. For example, translation based methods (*TransE* (Bordes et al. , 2013), *TransH* (Wang et al. , 2014), *TransR* (Lin et al. , 2015a), *TransD* (Ji et al. , 2015), et al.) view the relation in a triplet as a translation operation from the head entity to the tail entity. Other KGE methods view relations as some kind of combination operators that link head entities and tail entities. For example, *HolE* (Nickel et al. , 2016) employs a circular correlation function as the combination operator in the project space. *ComplEx* (Trouillon et al. , 2016) makes use of complex valued embeddings and takes the matrix decomposition as the combination operator. *RT* (Wang et al. , 2019)uses Tucker decomposition for KGE. *RotateE* (Sun et al. , 2019) use the rotation operation in the complex space as the combination operator. Experimental results show these existing methods have strong feasibility and robustness in solving the mentioned two issues.

Recently, deep neural networks (DNN) based KGE methods (Dettmers, 2018; Nguyen et al. , 2018; Yao et al. , 2020; Vashishth et al. , 2020a; Vashishth et al. , 2020b) push the performance of KGE to a soaring height. Compared with previous methods, this kind of methods can learn more effective embeddings mainly due to the powerful learning ability inherent in the DNN models. However, as pointed out by Xu et al. (2020) that existing research did not make a proper trade-off between the model complexity (the number of parameters) and the model expressiveness (the performance in capturing semantic information). Thus deep convolutional neural networks (DCNN) based methods are achieving more and more research attention due to their simple but effective structure. However, Chen et al. (2018) point out that the DCNN based methods usually suffer from the reduced feature resolution issue that is caused by the repeated combination of max-pooling and down-sampling(“*striding*”) performed at consecutive layers of DCNNs. This will result in feature maps with significantly reduced spatial resolution when DCNN is employed in a fully convolutional fashion.

To address this issue, we propose an atrous convolution based KGE method which allows the model to effectively enlarge the field of view of filters almost without increasing the number of parameters or the amount of computations. To address the vanishing/exploding gradient issue inherent in the DNN based learning frame and the *original information forgotten* issue when more convolutions used, we introduce residual learning in the our method. We propose two learning structures to integrate different kinds of convolutions together: one is a serial structure, and the other is a parallel structure. We evaluate our method on six diverse benchmark datasets. Extensive experiments show that our method achieves better result than the compared state-of-the-art baselines under most evaluation metrics on these datasets.

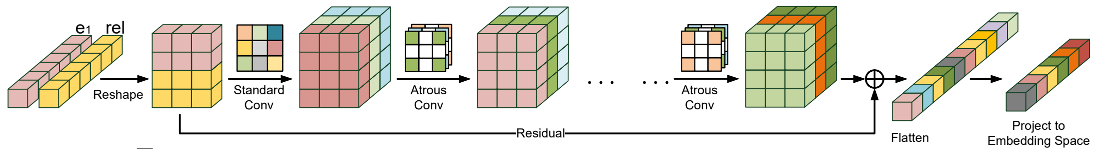
2 Related Work

Translation based KGE methods view the relation in a triplet as a translation operation from the head entity to the tail entity. These methods usually define a score function (or energy function) that has a form like $||\mathbf{h} + \mathbf{r} - \mathbf{t}\ ||$ to measure the plausibility of a triplet. During training, almost all of them minimize a margin based ranking loss function over the training data. *TransE* (Bordes et al. , 2013) is a seminal work in this branch. It directly takes the embedding space as a translation space. Formally, it tries to let $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ if $<h, r, t>$ holds. *TransH* (Wang et al. , 2014) models a relation as a hyperplane together with a translation operation on it. *TransR* (Lin et al. , 2015a) models entities and relations in distinct spaces, i.e., the entity space and multiple relation spaces. *TransD* (Ji et al. , 2015) models each entity or relation by two vectors. *TranSparse* (Ji et al. , 2016) mainly considers the heterogeneity property and the imbalance property in KGs. *PTransE* (Lin et al. , 2015b) integrates relation paths into a TransE model. *ITransF* (Xie et al. , 2017) uses a sparse attention mechanism to discover hidden concepts of relations and to transfer knowledge through the sharing of concepts. Recently, researchers also employ the methods of combining different distance functions together for KGE. For example, Sadeghi et al. (2019) proposed a multi distance embedding (*MDE*) model, which consists of several distances as objectives.

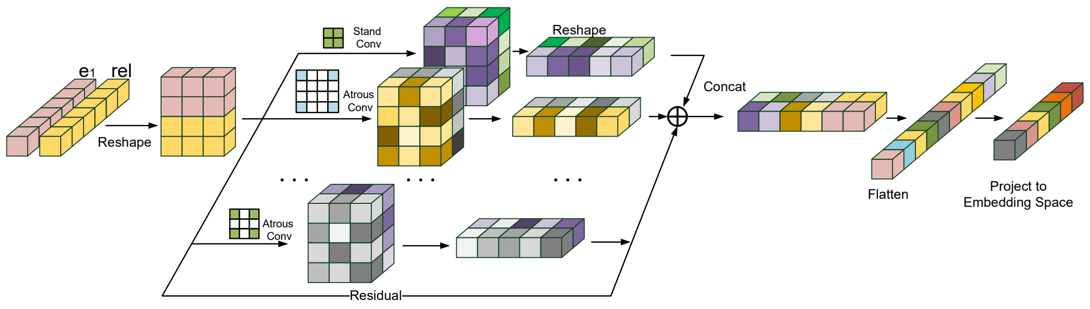
Bilinear KGE models use different kinds of combination operators other than the translation. For example, *HolE* (Nickel et al. , 2016) employs a circular correlation as the combination operator. *ComplEx* (Trouillon et al. , 2016) makes use of complex valued embedding and takes matrix decomposition as the combination operator. Similar to *ComplEx*, *RotatE* (Sun et al. , 2019) also use a complex space where each relation is defined as a rotation from the source entity to the target entity. Xu and Li (2019) proposed DihEdral for KG relation embedding. By leveraging the desired properties of dihedral group, their method could support many kinds of relations like symmetry, inversion, etc. Wang et al. (2019) propose the *Relational Tucker3(RT)* decomposition for multi-relational link prediction in knowledge graphs.

Other work, *KG2E* (He et al. , 2015) uses a density-based method to model the certainty of entities and relations in a space of multi-dimensional Gaussian distributions. *TransG* (Xiao et al. , 2016) mainly addresses the issue of multiple relation semantics.

Recently, researchers begin to explore the DNN based methods for KGE and achieve state-of-the-art results. For example, *ConvE* (Dettmers, 2018) uses 2D convolution over embeddings and multiple layers of nonlinear features to model KGs. *ConvKB* (Nguyen et al. , 2018) also use convolutional neural network for KGE. *ConMask* (Shi and Weniger, 2018)uses relationship-dependent content masking, fully convolutional neural networks, and semantic averaging to extract relationship-dependent embeddings from the textual features of entities and relations in KGs. More recently, Guo et al. (2019) studied the path-level KG embedding learning and proposed recurrent skipping networks(*RSNs*) to remedy the problems of using sequence models to learn relational paths. Yao et al. (2020) integrate BERT (Devlin et al.



(a)



(b)

Figure 1: (a): Structure of *Serial AcrE*; (b): Structure of *Parallel AcrE*.

, 2019) into the KGE model. Wang et al. (2020) propose CoKE which uses Transformer (Vaswani et al. , 2017). Vashishth et al. (2020a) extend *ConvE* by increasing the interactions with feature permutation, feature reshaping, and circular convolution.

Most recently, graph based neural network(*GNN*) methods are achieving more and more attentions. Schlichtkrull et al. (2018) propose *R-GCN*, which is a graph based DNN model that uses neighboring information of each entity. Bansal et al. (2019) propose *A2N*, an attention-based model based on graph neighborhood. Shang et al. (2019) propose a weighted graph convolutional network based method that mainly utilizes learnable relational specific scalar weights during GCN aggregation. Ye et al. (2019) propose *VR-GCN*, which is an extention of graph convolutional networks for embedding both nodes and relations. Shang et al. (2019) propose *SACN* that takes the benefit of both GCN and *ConvE*. Vashishth et al. (2020b) propose *CompGCN* which jointly embeds both nodes and relations in a relational graph.

However, as pointed out by Xu et al. (2020) that most of these existing DNN-based or GNN-based KGE methods are very complex and time-consuming, which prevents them be used in some on-line or real-time application scenarios.

3 AcrE Model

We denote our model as *AcrE* (the abbreviation of ***A***trous ***C***onvolution and ***R***esidual ***E***mbedding). In this study, we design two structures to integrate the standard convolution and atrous convolutions together. One is a serial structure as shown in Figure 1 (a), and the other is a parallel structure as shown in Figure 1 (b). We will introduce them one by one in the following.

3.1 Serial AcrE Model

In the *Serial AcrE* model, the standard convolution and atrous convolutions are organized in a serial manner. As shown in Figure 1 (a), the output of one convolution will be taken as the input of its subsequent adjoining convolution. In this model, the embeddings of an entity and its relation are first reshaped into a 2-Dimension representation, then a standard convolution and several atrous convolutions are performed serially. Next, the outputted embeddings of the last atrous convolution and the initial embeddings are combined by a residual learning based method. The combined embeddings are flattened into a vector. This vector is then used as features to get the probability distributions for the entity candidates.

2D Embedding Representation For a triplet $<h, r, t>$, we denote $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$ as their corresponding embedding representations. *ConvE* points out that a 2-Dimension (*2D* for short) convolution operation is better than a *1D*-convolution operation because a *2D*-convolution increases the expressiveness of a CNN model through additional points of interaction between embeddings. Thus follow *ConvE*, we also use a *2D* convolution in our model. To this end, the embedding concatenation of an entity and its linked relation is reshaped into a *2D* embedding representation.

Specifically, we use τ to denote a *2D* reshaping function and use $\mathbf{e}$ to denote the embedding of an entity *e*. If $\mathbf{e}, \mathbf{r} \in \mathbb{R}^m$, $\tau([\mathbf{e}; \mathbf{r}]) \in \mathbb{R}^{n_1 \times n_2}$ where $2 \times m = n_1 \times n_2$. In this study, we use $[\mathbf{e}; \mathbf{r}]$ to denote the concatenation of $\mathbf{e}$ and $\mathbf{r}$.

Standard Convolution based Learning After the 2D reshaping process, a standard convolution operation is performed with Equation 1.

$$\mathbf{C}_0^i = \omega_0^i \star \tau([\mathbf{e}; \mathbf{r}]) + \mathbf{b}_0^i \tag{1}$$

where $\star$ denotes convolution operation, $\omega_0^i \in \mathbb{R}^{k \times k}$ is the i-th filter and $\mathbf{b}_0^i$ is the i-th bias vector.

Then the outputs of these filters are stacked to form the output of the standard convolution learning. We denote the final output of this standard convolution learning as $\mathbf{C_0}$, which could be simply written as $\mathbf{C}_0 = \left[\mathbf{C}_0^1 : \mathbf{C}_0^2 : \mathbf{C}_0^3 : ... : \mathbf{C}_0^F\right]$ and F is the number of filters used.

It should be noted that we don't perform a max-pooling operation that is often used in traditional CNN models. This is because the input of our model is always an entity and a relation. Thus the length of the convolution output is fixed. It is unnecessary to use a max-pooling to generate a new length-fixed representation. Our in-house experiments show that there is no obvious performance difference between with and without a max-pooling operation.

Atrous Convolution based Learning Atrous convolution, also called as dilated convolution, inserts some holes (zeros) in the input during convolution. Given an input vector $\mathbf{x}$ with a filter vector $\mathbf{w}$ of length *K*, the output vector $\mathbf{y}$ of an atrous convolution is computed with Equation 2.

$$y_i = \sum_{k=1}^{K} x_{i+l \times k} \times w_k \tag{2}$$

Here *l* (the atrous rate parameter) means the stride with which we sample the input. Obviously, the standard convolution is a special case of the atrous convolution when *l* is set to 1.

Specifically, in the *Serial AcrE* model, an atrous convolution takes the output of its previous convolution as input, and output a new result with Equation 3.

$$\mathbf{C_t} = \omega_\mathbf{t} \star \mathbf{C}_{t-1} + \mathbf{b_t} \tag{3}$$

where $\mathbf{C_{t-1}}$ is the output of previous convolution operation, $\omega_\mathbf{t}$ and $\mathbf{b_t}$ are the filter and bias vector respectively in the *i-th* convolution.

Feature Vector Generation In the *Serial AcrE* model, different kinds of convolutions are performed one by one. Each convolution will extract some interaction features from the output of its previous convolution. Thus the mined features would "*forget*" more and more original input information as convolutions performed. However, the original information is the foundation of all mined features, so "*forget*" them will increase the risk that the mined features are actually irrelevant to what are needed. We call this phenomenon as *original information forgotten* issue. Besides, there is an inherent vanishing/exploding gradient issue in the deep networks. Here we use the residual learning method (He et al. , 2016) to add original input information back so as to address both issues. Then the result of residual learning is flattened into a feature vector. Specifically, the whole process is defined with Equation 4.

$$\mathbf{o} = Flatten(\mathrm{ReLU}(\mathbf{C}_T + \tau([\mathbf{e}; \mathbf{r}]))) \tag{4}$$

where $\mathbf{C}_T$ is the output of last atrous convolution, and T is the number of atrous convolutions.

Score Function With the generated feature vector $\mathbf{o}$, we define the following function to compute a score to measure the degree of an entity candidate t can form a correct triplet with the input <*h*,*r*>.

$$\psi(h, r, t) = (\mathbf{oW} + \mathbf{b})\, \mathbf{t}^\top \tag{5}$$

where $\mathbf{W}$ is a transformation matrix and $\mathbf{b}$ is a bias vector. Then a sigmoid function is used to get the probability distribution over all candidate entities.

$$p\,(t|h, r) = sigmoid(\psi(h, r, t)) \tag{6}$$

3.2 Parallel AcrE Model

In the *Parallel AcrE* model, the standard convolution and atrous convolutions are organized in a parallel manner. As shown in Figure 1 (b), different kinds of convolutions are performed simultaneously, then their results are combined and flattened into a vector. Similar to the *Serial AcrE* model, this vector is used as features to get the probability distributions for the entity candidates.

Compared with the *Serial AcrE* model, most of the components in the *Parallel AcrE* model have the same definitions except for the results integration and feature vector generation. We will introduce these two differences in the following part.

Results Integration Different from the *serial* structure, there will be multi results generated by different convolution operations. Accordingly, we need to integrate these results together. This process can be defined with following Equation 7.

$$\mathbf{C} = \mathbf{C_0} \oplus \mathbf{C_1} \oplus ... \oplus \mathbf{C_T} \tag{7}$$

where $\mathbf{C_0}$ is the result of standard convolution and $\mathbf{C_i}$ is the result of the *i-th* atrous convolution, and $\oplus$ means a result integration operation. There are different kinds of integration methods. In this study, we explore two widely used methods for this. One is an element-add operation based method, the other is a concatenation operation based method.

Feature Vector Generation As shown in Figure 1, the final output of the whole convolution learning is followed by a transformation operation. Then the results are flattened into the feature vector. Specially, the process can be written with Equation 8, where $\mathbf{W_1}$ is the transformation matrix.

$$\mathbf{c} = Flatten(\mathbf{W_1} Relu(\mathbf{C} + \tau([\mathbf{e}; \mathbf{r}]))) \tag{8}$$

3.3 Training

Different from other KGE methods that often use a max-margin loss function for training, most neural networks based KGE methods (like *ProjE*, *ConvE*, etc.) often use the following two kinds of ranking loss functions. One is a kind of binary cross-entropy loss that the ranking scores are calculated independently (*pointwise* ranking method), and the other is a kind of softmax regression loss that considers the ranking scores collectively (*listwise* ranking method). Both *ProjE* and *ConvE* show that the latter one achieves better experimental results. In *AcrE*, we define a same *listwise* loss function as used in *ConvE*.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[t_i \log p\left(t_i|h,r\right) + \left(1-t_i\right) \log\left(1 - p\left(t_i|h,r\right)\right) \right] \tag{9}$$

where $\mathbf{t}$ is a label vector whose elements are ones for relationships that exist and zero otherwise, and N is the number of entities in a KG. This loss function takes one *(h,r)* pair and scores it against all entities simultaneously. Thus our model is very fast for both training and inference.

4 Experiments and Analyses

4.1 Experiment Settings

Datasets We evaluate our method on six widely used benchmark datasets. The first two are WN18 (Bordes et al. , 2014) and FB15k (Bordes et al. , 2014). The second two are WN18RR and FB15k-237 (Dettmers, 2018), which are two variant datasets for WN18 and FB15k to avoid test leakage. The rest two are Alyawarra Kinship (Lin et al. , 2018) and DB100K (Ding et al. , 2018), both are new datasets proposed in recent years. Some statistics of these six datasets are shown in Table 1.

Evaluation Task We use *link prediction*, one of the most frequently used benchmark evaluation tasks for KGE methods, to evaluate our model. *Link prediction* is to predict the missing *h* or *t* for a correct triplet <*h, r, t*>, i.e., predict *t* given <*h, r*> or predict *h* given <*r, t*>. Rather than requiring one best answer, this task emphasizes more on ranking a set of candidate entities from the KG. *Hits@k* and *MRR* are often used as the evaluation metrics.

Dataset	#R	#E	#Triplet		
			Train	Valid	Test
DB100K	470	99,604	597,572	50,000	50,000
WN18	18	40,943	141,442	5,000	5,000
FB15k	1,345	14,951	483,142	50,000	59,071
WN18RR	11	40,943	86,835	3,034	3,134
FB15k-237	237	14,541	272,115	17,535	20,446
Kinship	25	104	8,544	1,068	1,074

Table 1: Dataset statistics.

Model	MRR	H@1	H@3	H@10
TransE(Bordes et al. , 2013)	0.111	1.6	16.4	27
DistMult(Yang et al. , 2015)	0.233	11.5	30.1	44.8
HolE(Nickel et al. , 2016)	0.26	18.2	30.9	41.1
ComplEx(Trouillon et al. , 2016)	0.242	12.6	31.2	44
Analogy(Liu et al. , 2017)	0.252	14.2	32.3	42.7
RUGE(Guo et al. , 2018)	0.246	12.9	32.5	43.3
ComplEx-NNE+AER(Ding et al. , 2018)	0.306	24.4	33.4	41.8
Sys-SEEK(Xu et al. , 2020)	0.306	22.5	34.3	46.2
SEEK(Xu et al. , 2020)	0.338	26.8	37	46.7
AcrE (Serial)	**0.399**	**30.4**	**45.3**	**57.0**
AcrE (Parallel)	**0.413**	**31.4**	**47.2**	**58.8**

Table 2: Experimental results on DB100k. All the compared results are taken from Xu et al. (2020).

In experiments, all the parameters, including initial embeddings, transformation matrices, and bias vectors, are randomly initialized. Hyper-parameters are selected by a grid search on the validation set. All the results are reported when 3 atrous convolutions used for both learning structures.

4.2 Experimental Results

Overall Results Table 2 and 3 show the experimental results on different datasets under different evaluation metrics. It should be noted that not all models report their results on all these six datasets, so the compared baselines on different datasets are different in these two tables. In subsequent part, all the experimental results for the compared baselines are taken from some latest published papers or their original papers. From these results we can draw following two conclusions.

First, our model is very robust and it significantly outperforms the compared state-of-the-art results under all the evaluation metrics on all datasets except for WN18RR. Especially on DB100K, FB15k, and Kinship, both *AcrE (Serial)* and *AcrE (Parallem)* outperform the compared baselines by a large margin under almost all the evaluation metrics. As for WN18RR, our model still achieves very competitive results. Especially when compared with other DCNN-based KGE methods like *ConvE* and *ConvKB*, we can see that both the *Serial* and the *Parallel* models perform much better.

Second, *AcrE (Parallel)* performs better than *AcrE (Serial)* in most cases. We think this is mainly due to the reason that the *Serial* structure based method suffers more from the *original information forgotten* issue than the *Parallel* structure based method.

Detailed Results We conduct following two kinds of detailed experiments to further demonstrate the performance of our model. One is *Head and Tail Prediction*, and the other is *Prediction by Categories*.

In the first kind of detailed experiments, we compare the performance of our model with several representative state-of-the-art baselines on FB15k-237 for predicting missing head entities and predicting missing tail entities. The results are summarized into Table 4, from which we can see that our model

	FB15K				WN18			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE(Bordes et al. , 2013)	0.463	29.7	57.8	74.9	0.495	11.3	88.8	94.3
HolE(Nickel et al. , 2016)	0.524	40.2	61.3	73.9	93.8	93.0	94.5	94.9
ComplEx(Trouillon et al. , 2016)	0.692	59.9	75.9	84.0	0.941	93.6	94.5	94.7
SimplE(Kazemi et al. , 2018)	0.727	66.0	77.3	83.8	0.942	93.9	94.4	94.7
D4-Gumbel(Xu and Li, 2019)	0.728	64.8	78.2	86.4	0.728	64.8	78.2	86.4
D4-STE(Xu and Li, 2019)	0.733	64.1	80.3	87.7	0.733	64.1	80.3	87.7
ConvE(Dettmers, 2018)	0.657	55.8	72.3	83.1	0.942	93.5	94.7	95.5
R-GCN(Schlichtkrull et al. , 2018)	0.696	60.1	76.0	84.2	0.696	60.1	76.0	84.2
RotatE(Sun et al. , 2019)	0.797	74.6	83.0	88.4	0.949	94.4	**95.2**	**95.9**
RSNs(Guo et al. , 2019)	0.78	72.2	-	87.3	0.94	92.2	-	95.3
AcrE(Serial)	0.791	72.7	**83.8**	**89.6**	**0.950**	**94.6**	**95.3**	**95.9**
AcrE(Parallel)	**0.815**	**76.4**	**85.2**	**89.8**	0.948	94.3	**95.2**	95.7

	FB15K-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
ConvE(Dettmers, 2018)	0.312	22.5	34.1	49.7	0.43	40	44	52
ConvKB(Nguyen et al. , 2018)	0.243	15.5	37.1	42.1	0.249	0.057	41.7	52.4
R-GCN(Schlichtkrull et al. , 2018)	0.164	10	18.1	30	0.123	8	13.7	20.7
RotatE(Sun et al. , 2019)	0.338	24.1	37.5	53.3	0.476	42.8	49.2	**57.1**
D4-STE(Xu and Li, 2019)	0.320	23.0	35.3	50.2	0.480	**45.2**	49.1	53.6
SACN(Shang et al. , 2019)	0.35	26.0	39.0	54.0	0.47	43.0	48.0	54.0
HypER(Balazevic et al. , 2019b)	0.341	25.2	37.6	52.0	0.465	43.6	47.7	52.2
ConvR(Jiang et al. , 2019)	0.350	26.1	38.5	52.8	0.475	44.3	48.9	53.7
VR-GCN(Ye et al. , 2019)	0.248	15.9	27.2	43.2	-	-	-	-
RSNs (Guo et al. , 2019)	0.280	20.2	-	45.3	-	-	-	-
DK-STE (Xu and Li, 2019)	0.320	23.0	35.3	50.2	**0.480**	**45.2**	49.1	53.6
CompGCN(Vashishth et al. , 2020b)	0.355	26.4	39.0	53.5	0.479	44.3	**49.4**	54.6
AcrE(Serial)	0.352	26.0	38.8	53.7	0.463	42.9	47.8	53.4
AcrE(Parallel)	**0.358**	**26.6**	**39.3**	**54.5**	0.459	42.2	47.3	53.2

	Kinship			
	MRR	H@1	H@3	H@10
ComplEx(Trouillon et al. , 2016)	0.823	73.3	89.9	97.1
ConvE(Dettmers, 2018)	0.833	73.8	91.7	98.1
ConvKB(Nguyen et al. , 2018)	0.614	43.6	75.5	95.3
R-GCN(Schlichtkrull et al. , 2018)	0.109	3	8.8	23.9
SimplE(Kazemi et al. , 2018)	0.752	62.6	85.4	97.2
RotatE(Sun et al. , 2019)	0.843	76.0	91.9	97.8
HAKE(Zhang et al. , 2020)	0.852	76.9	92.8	98.0
InteractE(Vashishth et al. , 2020a)	0.777	66.4	87.0	95.9
CompGCN(Vashishth et al. , 2020b)	0.778	66.7	86.8	96.7
CoKE(Wang et al. , 2020)	0.793	69.3	87.8	95.4
AcrE(Serial)	**0.864**	**78.7**	**93.1**	**98.7**
AcrE(Parallel)	**0.864**	**78.5**	**93.9**	**98.4**

Table 3: Experimental results on the rest five benchmark datasets.

	Predict Head				Predict Tail			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
ConvE(Dettmers, 2018)	0.211	13.2	23.1	36.8	0.416	32.3	45.7	60.1
SACN(Shang et al. , 2019)	0.241	15.8	26.0	40.9	0.446	35.2	49.0	63.1
RotatE(Sun et al. , 2019)	0.239	14.9	26.5	42.4	0.432	32.9	47.7	63.9
InteractE(Vashishth et al. , 2020a)	0.258	17.0	28.3	43.7	0.454	**35.8**	49.8	64.4
AcrE (Serial)	0.254	16.6	27.9	43.4	0.451	35.3	49.7	64.2
AcrE (Parallel)	**0.261**	**17.3**	**28.6**	**44.2**	**0.455**	**35.8**	**49.9**	**64.7**

Table 4: Head and Tail Predictions on FB15k-237. All the compared results are the best results we can achieve by running the source codes provided by the original papers.

Model	Prediction Head (Hits@10)				Prediction Tail (Hits@10)			
	1-to-1	1-to-n	n-to-1	m-to-n	1-to-1	1-to-n	n-to-1	m-to-n
TransE(Bodes et al., 2013)	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH(Wang et al., 2014)	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransD(Ji et al.,2015)	86.1	95.5	39.8	78.5	85.4	50.6	94.4	81.2
CTransR(Lin et al., 2015a)	81.5	89	34.7	71.2	80.8	38.6	90.1	73.8
KG2E(He et al., 2015)	92.3	94.6	66	69.6	92.6	67.9	94.4	73.4
TranSparse(Ji et al., 2016)	87.5	95.9	44.4	81.2	87.6	57	94.5	83.7
STransE(Nguyen et al., 2016)	82.8	94.2	50.4	80.1	82.4	56.9	93.4	83.1
TransG(Xiao et al., 2016)	93.0	96	62.5	86.8	92.8	68.1	94.5	88.8
ComplEx(Trouillon et al. , 2016)	**93.9**	**96.9**	69.2	89.3	**93.8**	82.3	95.2	91.0
Jointly(Xu et al., 2017)	83.8	95.1	21.1	47.9	83	30.8	94.7	53.1
RotatE(Sun et al. , 2019)	92.9	96.7	60.2	89.3	92.3	71.3	**96.1**	**92.2**
AcrE(Serial)	91.0	**96.9**	**70.5**	89.1	90.7	**85.2**	95.9	**92.2**
AcrE(Parallel)	92.5	**97.0**	**69.5**	**89.5**	92.1	**84.0**	**96.1**	**92.7**

Table 5: Predictions by Categories on FB15k. The compared results are taken from their original papers.

outperforms the compared baselines again under all the evaluation metrics.

In the second kind of detailed experiments, we compare the performance of our model with several representative state-of-the-art baselines on FB15k for predicting by different categories. The results are shown in Table 5. We can see that *ArcE* does much better than other compared baselines on almost all types of relations except the 1-to-1 relations. This merit is much important for real application scenarios where the complexer relations often take up large proportions. For example, in FB15k, one of the largest available KGs, the triplets of 1-to-1 are about 1.4%, 1-to-n are about 8.9%, n-to-1 are about 14.6%, and m-to-n are about 75.1%.

Ablation Results Table 6 shows the ablation experiments of our model on FB15k and FB15k-237. We can see that there is a large different between the performance of "with/without" residual learning in most cases. As analyzed above, the more serial convolutions used, the more original information would be *forgotten*. While a residual learning adds the original information back. Accordingly, the mentioned issue is alleviated greatly. Since *AcrE (Serial) forgets* more original information than *AcrE (Parallel)*, it achieves more performance gains from residual learning.

From Table 6 we can also observe that the integration method plays important role in *AcrE (Parallel)*. Usually, the concatenation based integration method is superior to an element-add based integration method in most cases. Here we do not use some complexer integration methods like *gate control* based methods for we do not want to make the model too complex.

Besides, the atrous rate and the number of atrous convolutions used also affect the performance. Here we do not report the performance under different settings of these two hyper-parameters due to space

		FB15K				FB15k-237			
		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
AcrE(Serial)		0.791	72.7	83.8	89.6	0.352	26.0	38.8	53.7
-Residual		0.776	70.6	82.8	89.1	0.351	25.8	38.6	53.7
AcrE(Parallel)		0.815	76.4	85.2	89.8	0.358	26.6	39.3	54.5
-Residual		0.804	74.6	84.9	89.7	0.355	26.1	39.0	54.1
AcrE (Parallel)	*add*	0.803	74.4	84.6	89.7	0.356	26.5	38.9	54.1
	con	0.815	76.4	85.2	89.8	0.358	26.6	39.3	54.5

Table 6: Ablation experiments on FB15k and FB15k-237. "*add*" and "*con*" refer to the element-add and concatenation integration methods respectively.

Models	ParaNum (Millions)
ConvE(Dettmers, 2018)	$\approx$ **4.96**
RotatE(Sun et al. , 2019)	$\approx$ 29.32
SACN(Shang et al. , 2019)	$\approx$ 9.63
InteractE(Vashishth et al. , 2020a)	$\approx$ 10.7
CompGCN(Vashishth et al. , 2020b)	$\approx$ 9.45
HAKE(Zhang et al. , 2020)	$\approx$ 29.79
CoKE(Wang et al. , 2020)	$\approx$ 10.19
AcrE (Serial)	$\approx$ 5.61
AcrE(Parallel)	$\approx$ 6.22

Table 7: Parameter efficiency on FB15k-237 ("ParaNum" refers to the number of parameters).

limitation. In fact, both of these two parameters are easily selected due to their small search spaces.
Parameter efficiency We also compare the parameter efficiency between our model and some state-of-the-art models on FB15k-237. For each method, we report the number of parameters associated with the optimal configuration that leads to the performance shown in Table 3. The comparision results are shown in Table 7, from which we can see that the number of parameters in *AcrE* is close with *ConvE*, but is far less than that in other compared baselines. This is in line with our expectation: using atrous convolutions would not increase the parameters greatly. These results show that our model is more parameter efficient, it achieves substantially better results with fewer parameters. Note that *AcrE (Parallel)* has more parameters than *AcrE (Serial)* because it has an extra transformation operation after the result integration.

Here we do not quantitatively compare the runtime of different models for it is difficult to provide a fair evaluation environment: coding tricks, hyper-parameter settings (like *batch-size*, *learning rate*), parallelization, lot of non-model factors affect the runtime. However, *AcrE* can be viewed as a variant of *ConvE*. Theoretically, it has the same time complexity as *ConvE* that has been proven to be faster than most existing state-of-the-art methods. Taking FB15k-237 as an example, when using a Titan XP GPU server, it takes about 220 and 100 seconds per epoch during training for *AcrE (Serial)* and *AcrE (Parallel)* respectively. As for inference, it only takes 14 and 6 seconds for *AcrE (Serial)* and *AcrE (Parallel)* respectively to finish the whole test set evaluation. While some latest GNN or DNN based methods often takes many hours even several days to complete the same work under the same experiment settings.

5 Conclusions

In this paper, we propose *AcrE*, a simple but effective DNN-based KGE model. We make comprehensive comparisons between *AcrE* and many state-of-the-art baselines on sis diverse benchmark datasets. Extensive experimental results show that *AcrE* is very effective and it achieves better results than the compared baselines under most evaluation metrics on six benchmark datasets. The main contributions of our method are summarized as follows. First, to our best knowledge, this is the first work that uses different kinds of convolutions for the KGE task. Second, we propose two simple but effective learning structures to integrate different kinds of convolutions together. Third, the proposed model has much better parameter efficiency than the compared baselines.

Acknowledgements

This work is supported by the National Key R&D Program of China (No.2018YFC0830701), the National Natural Science Foundation of China (No.61572120), the Fundamental Research Funds for the Central Universities (No.N181602013 and No.N171602003), Ten Thousand Talent Program (No.ZX20200035), and Liaoning Distinguished Professor (No.XLYC1902057).

References

Afshin Sadeghi, Damien Graux, Hamed Shariat Yazdi, Jens Lehmann. 2019. *MDE: Multi Distance Embeddings for Link Prediction in Knowledge Graphs.*

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. *Translating embeddings for modeling multi-relation data. In NIPS*, pages 2787-2795.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N.Gomez, Lukasz Kaiser. 2017. *Attention Is All You Need. arXiv: 1706.03762v4.*

Baoxu Shi and Tim Weninger. 2017. *ProjE: embedding projection for knowledge graph completion.* In *AAAI*, pages 1236-1242, 2017.

Baoxu Shi and Tim Weninger. 2018. *Open-World Knowledge Graph Completion.* In *AAAI2018.*

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor,J. 2008. *Freebase: a collaboratively created graph database or structuring human knowledge.* In *SIGMOD*, pages 1247–1250.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. *Asemantic matching energy function for learning with multi-relational data. Machine Learning*. 94(2):233–259.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. *Improving knowledge graph embedding using simple constraints. ACL*, pages 110–121.

Canran Xu, Ruijing Li. 2019. *Relation Embedding with Dihedral Group in Knowledge Graph. ACL2019*, pp 263-272.

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*

Dat Quoc Nguyen, KairitSirts, Lizhen Qu, and Mark Johnson. 2016. *STransE: a novel embedding model of entities and relationships in knowledge bases.* In *NAACL HLT*, pages 460-466.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung. 2018. *A Novel Embedding Model for Knowledge Base Comple-tion Based on Convolutional Neural Network. NAACL2018*, pages 327-333.

Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung. 2019. *A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization.* In*NAACL*, pages 2180–2189.

Geoffrey E.Hinton, Nitish Srivastava, Alex Krizhevsky, Illy Sutskever, and RuslanR. Salakhutdinov. 2012. *Improving neural net-works by preventing co-adaptation of feature detectors.*
newblock*arXiv: 1207.0580.*

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. *Knowledge graph embedding via dynamic mapping matrix.* In *ACL*, pages 687-696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. *Knowledge graph completion with adaptive sparse transfer matrix.* In *AAAI2016.*

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. *TransG: a generative model for knowledge graph embedding.* In *ACL*, pages 2316-2325.

Han Xiao, Minlie Huang, LianMeng, and Xiaoyan Zhu. 2017. *SSP: semantic space projection for knowledge graph embedding with text descriptions.* In *AAAI2017*.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. *Analogical inference for multi-relational embeddings.* In *ICML*, pages 2168–2178.

Ivana Balazevic, Carl Allen, Timothy M.Hospedales. 2019. *TuckER: Tensor Factorization for Knowledge Graph Completion. arXiv: 1901.09590v1.*

Ivana Balazevic, Carl Allen, Timothy M.Hospedales. 2019. *Hypernetwork knowledge graph embeddings.* In *International Conference on Artificial Neural Networks*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* In *NAACL*, pages 4171-4186.

Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. 2017. *Knowledge graph representation with jointly structural and textual encoding.* In *IJCAI*, pages 1318-1324.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2016. *Deep Residual Learning for Image Recognition.* In *CVPR2016*.

Liang-Chieh Chen, George Papandreou, Kevin Murphy and Alan L.Yuille. 2018. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Liang Yao, Chengsheng Mao, Yuan Luo. 2020. *KG-BERT: BERT for Knowledge Graph Completion.* In *AAAI*, 2020.

Lingbing Guo, Zequn Sun, Wei Hu. 2019. *Learning to Exploit Long-term Relational Dependencies in Knowledge Graphs. ICML2019.*

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. *Modeling Relational Data with Graph Convolutional Networks. ESWC.*

Maximilian Nickel, Lorenzo Rosasco, Tomaso Poggio. 2016. *Holo-graphic embeddings of knowledge graphs.* In *AAAI*, pages 1995-1961.

Miller, G. A. 1995. *Wordnet: a lexical database for english. Communications of the ACM*, 38(11):39–41.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. *An inter-pretable knowledge transfer model for knowledge base completion.* In *ACL*, pages 950-962.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang,Jing Liu, Yajuan Lyu, Yong Zhu, Hua Wu. 2020. *CoKE: Contextualized Knowledge Graph Embedding. AAAI.*

Rui Ye, Xin Li, Yujie Fang, Hongyu Zang, and Mingzhong Wang. 2019. *A vectorized relational graph convolutional network for multi-relational network alignment.* In *IJCAI*, pages 4135-4141.

Ruobing Xie, Zhiyuan Liu, JiaJia, Huanbo Luan, and Maosong Sun. 2016. *Representation learning of knowledge graphs with entity descriptions.* In *AAAI*, 2659-2665.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. *Representation learning of knowledge graphs with hierarchical types.* In *IJCAI*, pages 2965-2971.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Berg, R. v. d.; Titov, I.; and Welling, M. 2017. *Modeling Relational Data with Graph Convolutional Networks . arXiv:1703.06103.*

Seyed Mehran Kazemi and David Poole. 2018. *SimplE Embedding for Link Prediction in Knowledge Graphs.* In *NeurIPS*, 2018.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, Partha Talukdar. 2020. *InteractE: Improving Convolution-based Knowledge Graph Embeddings by Increasing Feature Interactions.* In *AAAI*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Partha Talukdar. 2020. *Composition-based multi-relational graph convolutional networks.* In *ICLR*, 2020.

Shizhu He, Kang Liu, Liang Guo, Jun Zhao. 2015. *Learning to Represent Knowledge Graph with Gaussian Embedding.* In *CIKM*.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. *Knowledge graph embedding with iterative guidance from soft rules.* In *AAAI*, pages 4816–4823.

Tim Dettmers. 2018. *Convolutional 2D Knowledge Graph Embeddings.* In *AAAI.*

Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier and Guillaume Bouchard. 2016. *Complex embeddings for simple link prediction.* In *ICML.*

Toutanova, K., and Chen, D. Observed Versus 2015. *Latent Features for Knowledge Base and Text Inference.* In Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compo-sitionality, pp57-66.

Trapit Bansal, Da-Cheng Juan, Sujith Ravi, Andrew McCallum. 2019. A2N: Attending to Neighbors for Knowledge Graph Inference. In *ACL*, 2019, pages 4387–4392.

Wentao Xu, Shun Zheng, Liang He, Bin Shao, Jian Yin, and Tie-Yan Liu. 2020. SEEK: Segmented Embedding of Knowledge Graphs. In arXiv:2005.00856v1.

Xiaocheng Feng, Jiang Guo, Bing Qin, Ting Liu, Yongjie Liu. 2017. *Effective Deep Memory Networks for Distant Supervised Relation Extraction.* In *IJCAI*, page 4002-4008.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. *Adaptive convolution for multi-relational learning.* In *NAACL.*

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping.. In *EMNLP.*

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. *Learning entity and relation embeddings for knowledge graph completion.* In *AAAI*, pages 2181-2187.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. *Modeling relation paths for representation learning of knowledge bases.* In *EMNLP*, pages 705-714.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016 *Knowledge repre-sentation learning with entities, attributes and relations.* In *IJCAI*, pages 2866-2872.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2017. *Neural Relation Extraction with Multi-lingual Attention.* In *ACL*, pages 34-43.

Yanjie Wang, Samuel Broscheit, Rainer Gemulla. 2019. *A Relational Tucker Decomposition for Multi-Relational Link Prediction. arXiv:1902.00898v1.*

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. *Embedding Entities and Relations for Learning and Inference in Knowledge Bases.* In *Proceedings of ICLR 2015.*

Yuval Pinter and Jacob Eisenstein. 2018. *Predicting Semantic Relations using Global Graph Properties. arXiv: 1808.08644v1.*

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, Jie Wang. 2020. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In *AAAI.*

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. *Knowledge graph embedding by translating on hyperplanes*. In *AAAI*, pages 1112-1119.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, Jian Tang. 2019. *RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. ICLR.*