ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230615

Multi-Aspect Enhanced Convolutional Neural Networks for Knowledge Graph Completion

Fu Zhang[a;†], **Pengpeng Qiu**[a;†], **Tong Shen**[a] **and Jingwei Cheng**[a;*]

[a]School of Computer Science and Engineering, Northeastern University, China

Abstract. Knowledge graph completion (KGC, also referred to as link prediction) aims at predicting missing entities and relations in knowledge graphs (KGs). Knowledge graph embedding (KGE) techniques have been proven to be effective for link prediction. Currently, a series of convolutional neural networks (CNNs) based models (e.g., ConvE and its extended models) have attained excellent results for link prediction. However, several aspects that are important for link prediction using CNNs have not been considered and enhanced simultaneously, which significantly limit the performance of these models. In this paper we explore an effective KGE model based on CNNs. We investigate and discover four extremely important aspects that have a strong influence on ConvE: *entity and relation embeddings*, *entity-to-relation interaction* approaches, CNN *structure*, and *loss function*. Based on the optimization of the above four aspects, we propose a novel KGE method called ConvEICF. Through extensive experiments, we find that ConvEICF outperforms the previous state-of-the-art link prediction baselines on FB15k-237 and WN18RR datasets. In particular, ConvEICF achieves a Hits@10 score that is 11.2% and 6.5% better than ConvE on FB15k-237 and WN18RR datasets respectively. Additionally, through in-depth experiments we observe an interesting phenomenon and important finding that the very common 1-N scoring technique in KGE can be considerably improved by just adding a dropout operation. Our code is available at https://github.com/NEU-IDKE/ConvEICF.

1 Introduction

Knowledge graphs (KGs) store huge amounts of structured data, with projects such as Freebase [3], YAGO [12] and DBpedia [9]. A KG usually contains a large number of factual triples: (head entity, relation, tail entity) abbreviated as (h, r, t). In recent years, KGs are widely used in many areas, such as semantic searching [21], question answering [7], and recommender systems [21]. However, KGs, which are often *incomplete*, still miss a lot of valid triples. The incompleteness of KGs will lead to poor performance of downstream applications.

Since the incompleteness of KGs, *knowledge graph completion* (KGC, also referred to as *link prediction*) is thus proposed to predict missing facts in the existing KGs to make these KGs more comprehensive and accurate. Many *knowledge graph embedding* (KGE) models have achieved remarkable results for link prediction. KGE methods embed entities or relations in a low-dimensional and continuous vector space. The goal of KGE is to learn appropriate embedding vectors for entities and relations by performing some operations (e.g., addition and multiplication) on these vectors. The typical KGE models include TransE [4], DistMult [20], and etc., where TransE uses the addition operation and DisMult employs the multiplication operation. Based on the above two methods, a series of effective KGE models are derived such as RotatE [13] and TuckER [1].

Recently, a series of convolutional neural networks (CNNs) based models have elicited sufficient attention. ConvE [5] solves KG incompleteness by utilizing the 2D convolution and its performance is excellent. However, ConvE simply applies external 2D convolutional filters on the input matrix concatenating entities and relations with limited interactions, and only takes a channel as the convolution input. SACN [11] proposes to enhance entity and relation embeddings by graph convolutional networks (GCN), and integrates ConvE and TransE to improve the interactions between entities and relations. HypER [2] takes entities as input matrix and relations as convolutional filters to enhance the interactions between entities and relations. AcrE [10] modifies the CNN structure of ConvE to improve the performance for link prediction, by adopting convolutional filters with different sizes. Although these models optimize CNNs and result in significant improvements, their performance is still limited because several aspects that are important for link prediction using CNNs have not been considered and enhanced simultaneously. We investigate and discover the following *four* extremely important *aspects* that have a strong influence on the link prediction, which need to be further improved: enhancing entity and relation embeddings, improving entity and relation interactions, optimizing the CNN structure for feature extraction, and employing a loss function with more powerful optimization ability.

Currently, most ConvE-based models only improve one or two of the aforementioned four aspects while leaving the others unaltered as will be detailed in Section 2. Compared to ConvE, the performance of these models for link prediction is improved but the improvement is still limited. Based on ConvE, we devise a novel KGE model called **ConvEICF**, which significantly improves ConvE based on four aspects: Entity and Relation **E**mbeddings, Entity-to-Relation **I**nteraction, **C**NN Structure Optimization, and Loss **F**unction Selection. Our contributions in this paper are as follows:

- A unified KGE framework ConvEICF is proposed that innovatively improves and integrates four aspects mentioned above.
- We employ a simple and high efficiency framework LTE [22] to enhance entity and relation embeddings on ConvEICF. Further, we go deeper to investigate effects of the linear layer, BatchNorm, and dropout operations involved in LTE on the enhanced embeddings. Surprisingly, we observe from experiments an interesting

* Corresponding Author. Email: chengjingwei@mail.neu.edu.cn. †Equal contribution.

phenomenon and important finding that the common 1-N scoring technique in KGE can be considerably improved by just adding a dropout operation.
- A multi-channel interaction approach is proposed to allow entities and relations to fully interact from multiple levels.
- We optimize the CNN structure and propose a novel interactive feature extraction structure by using two convolutional layers and a convolutional attention layer. In this method, for the first time we innovatively apply a convolutional block attention module (CBAM) in computer vision [19] to further extract interactive features for link prediction.
- We also change the loss function from BCE to InfoNCE, which can make the model focus on hard negatives. We note that all of the ConvE-based models apply the same loss function − Binary Cross-Entropy (BCE) as ConvE does. We first introduce the InfoNCE loss [17] to CNN-based link prediction to achieve better performance.

We carry out extensive experiments to verify the superiority of our KGE method and the necessity of enhancing the aforementioned four aspects. The experimental results show that our ConvEICF achieves state-of-the-art (SOTA) performance for link prediction.

2 Related Work

This section mainly describes the related work involved in CNN-based models. We introduce commonly used methods related to entity and relation embeddings, entity-to-relation interaction techniques, CNN structure, and loss functions.

Firstly, for entity and relation embedding, we generally embed entities and relations to low-dimensional distributed representations based on existing triples in the KGs. This method is used widely in various KGE models (e.g., TransE and RotatE). However, the method of direct embedding leads to the incapacity to make full use of the structured information in the KGs. Recently, GCNs have been used to capture the structured information stored in the KGs and have been shown to achieve competitive performance on link prediction task. A large number of variants based on GCNs (e.g., WGCN [23] and CompGCN [16]) achieve better results for KGE. SACN [11] employs WGCN to enhance entity and relation embeddings and achieves better performance. However, GCNs need to process a large amount of data and require more computing resources. LTE [22] proposes to replace GCNs with a simple and high-efficiency framework. The LTE-based models achieve similar results as the GCN-based models, and do not require extensive computational resources and complex data processing.

Secondly, for entity-to-relation interaction techniques, previous CNN-based models (e.g., ConvE [5] and HypER [2]) suffer from weak interaction between entities and relations in the vector space, which affects the capacity of feature extraction. ConvE takes a simple concatenation of entities and relations and only has a channel as the convolution input. HypER puts forward a novel interaction way, which takes the entities as the input matrix and the relations as the convolutional filters so that the entities and relations can fully interact. InteractE [15] provides another interaction way, which is based on three key ideas: feature permutation, novel feature reshaping, and circular convolution. InteractE adopts a multi-channel interaction way, where its multi-channel is generated by feature permutation. Our ConvEICF proposes a multi-channel interaction approach to enable entity and relation interaction features as multi-channel input to CNNs.

Thirdly, for CNN structure, some models achieve good performance by setting different CNN structures. For example, AcrE [10] innovates the CNN structure, which uses a variety of different convolutional filters to extract the interaction features between entities and relations. Moreover, some convolutional attention mechanisms effectively improve convolutional models in the field of computer vision. SENet [8] proposes to explicitly model the interdependence between feature channels. However, SENet misses the spatial attention, which plays an important role in deciding “where" to focus. CBAM [19] in computer vision proposes to combine channel and spatial attention so that it can fully focus on important features. But such attention mechanisms (including SENet and CBAM) have never been used in the link prediction task.

Finally, for loss functions, the KGE models based on ConvE as mentioned above do not modify the loss function and still adopt the BCE loss function as the ConvE does. SimKGC [17] proposes to change the loss function from margin-based ranking loss to InfoNCE, which can make the model focus on hard negatives and the performance of SimKGC is greatly improved. Inspired by SimKGC, we intend to apply InfoNCE to our ConvEICF.

3 Methodology

In this section, we propose a novel KGE model ConvEICF based on CNNs. The architecture of ConvEICF is shown in **Figure 1**. ConvEICF consists of four main modules: embedding enhancement module, interaction module, interactive feature extraction module, and prediction module.

3.1 Notion

A KG $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a collection of valid factual triples in the form of (head entity, relation, tail entity), where $h,\ t \in \mathcal{E}$, $r \in \mathcal{R}$, $\mathcal{E} = \{e_1, e_2, \cdots, e_{|\mathcal{E}|}\}$ and $\mathcal{R} = \{r_1, r_2, \cdots, r_{|\mathcal{R}|}\}$ are the entity set and relation set in the KGs. $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the number of entities and relations respectively. The bold lowercase letters, i.e., $\boldsymbol{h}$, $\boldsymbol{r}$, and $\boldsymbol{t}$, are the corresponding embeddings of the head entity, relation, and tail entity, and $\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t} \in \mathbb{R}^d$.

3.2 Embedding Enhancement Module

Instead of GCNs as mentioned in Section 2, we use a simple and high-efficiency framework for enhancing entity and relation embeddings. Inspired by LTE [22], which achieves similar embedding performance compared to the models based on GCNs by exploiting three operations − linear layer, BatchNorm, and dropout, we adopt the similar framework and further go deeper to investigate effects of the three operations involved in LTE on the enhanced embeddings. The embedding representations for all entities and relations are learned, which are $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d}$ and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d}$, respectively. $\bar{\mathbf{E}}$ and $\bar{\mathbf{R}}$ denote the results of enhancing the entities embedding matrix $\mathbf{E}$ and the relations embedding matrix $\mathbf{R}$ by LTE, respectively. The transformation formula for $\bar{\mathbf{E}}$ and $\bar{\mathbf{R}}$ is as follows:

$$\begin{aligned} \bar{\mathbf{E}} &= \mathrm{Drop}(\mathrm{BN}(\mathbf{E}\mathbf{W}_{\mathbf{e}})) \\ \bar{\mathbf{R}} &= \mathrm{Drop}(\mathrm{BN}(\mathbf{R}\mathbf{W}_{\mathbf{r}})) \end{aligned} \tag{1}$$

where $\mathbf{W}_{\mathbf{e}}, \mathbf{W}_{\mathbf{r}} \in \mathbb{R}^{d \times d}$, $\bar{\mathbf{E}} \in \mathbb{R}^{|\mathcal{E}| \times d}$, and $\bar{\mathbf{R}} \in \mathbb{R}^{|\mathcal{R}| \times d}$. BN and Drop represent the BatchNorm operation and the dropout operation, respectively.

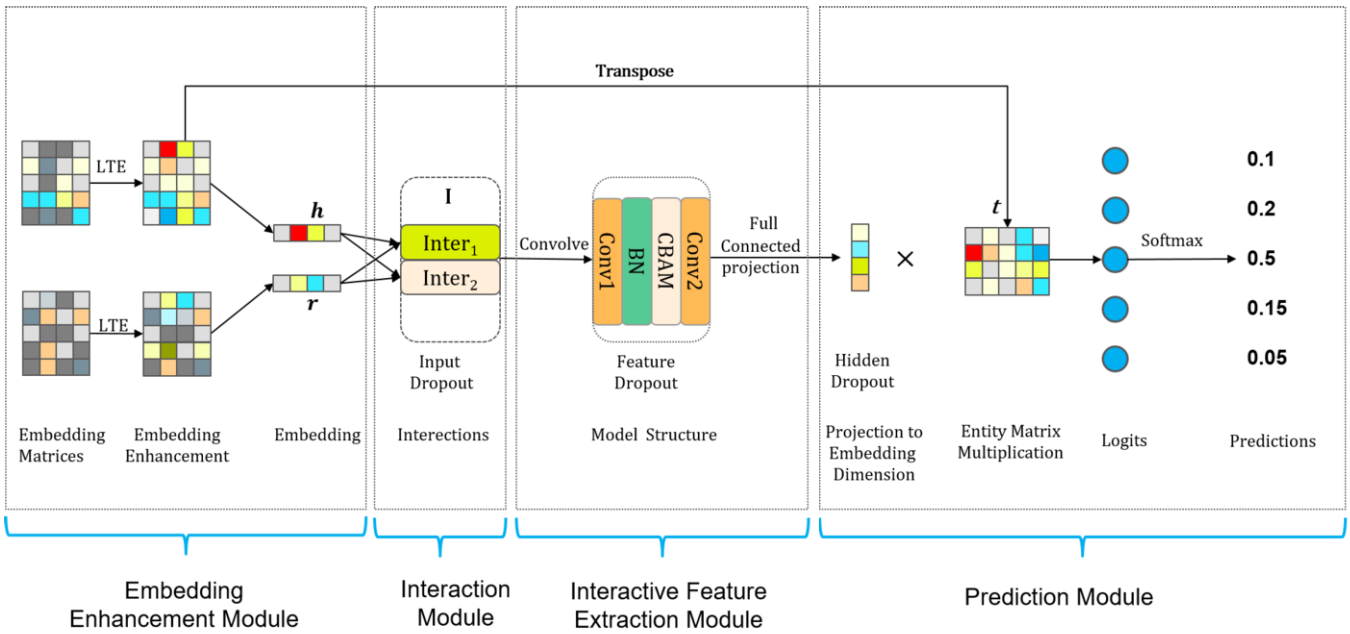

Figure 1. The framework of our proposed ConvEICF model consists of four modules: embedding enhancement module, interaction module, interaction feature extraction module, and prediction module. The input dropout, feature dropout, and hidden dropout are performed on the output result of the current operation respectively.

3.3 Interaction Module

The interaction module models the interaction of entities and relations in two approaches (i.e., 3D transformation and linear transformation). Each interaction generates a novel matrix as a convolutional input channel respectively. Specifically, the head entities and relations obtained from the enhanced embedding, i.e., $\boldsymbol{h} \in \bar{\mathbf{E}}, \boldsymbol{r} \in \bar{\mathbf{R}}$. After a learnable 3D transformation and a linear transformation respectively, we obtain two matrices containing different types of interaction information between entities and relations.

For the first interaction approach, we introduce 3D transformations to allow entities and relations to interact in a high-dimensional space. We simultaneously multiply the head entity and relation vectors by a learnable 3-dimensional matrix, and the results serve as entity and relation interaction features. The computation of the first entity-to-relation interaction approach is as follows:

$$\mathbf{Inter_1} = \boldsymbol{h} \times \mathbf{W}_{\mathbf{Inter_1}} \times \boldsymbol{r} \tag{2}$$

where $\mathbf{W}_{\mathbf{Inter_1}} \in \mathbb{R}^{d \times 2d \times d}$ and $\mathbf{Inter_1} \in \mathbb{R}^{2d}$. We reshape the result after interaction to make the dimension $1 \times H \times U$, where H and U denote the two dimensions of input matrix and $2d = H \times U$.

For the second interaction approach, we use linear transformation to interact between entities and relations. We take advantage of a simple concatenation operation and add one linear layer. Its operation is as follows:

$$\mathbf{Inter_2} = [\boldsymbol{h}; \boldsymbol{r}]\mathbf{W}_{\mathbf{Inter_2}} + \mathbf{b}_{\mathbf{Inter_2}} \tag{3}$$

where $[;]$ is the concatenation operation, $[\boldsymbol{h}; \boldsymbol{r}] \in \mathbb{R}^{2d}$, $\mathbf{W}_{\mathbf{Inter_2}} \in \mathbb{R}^{2d \times 2d}$, and $\mathbf{b}_{\mathbf{Inter_2}} \in \mathbb{R}^{2d}$ is the bias parameter. We keep the dimension of $\mathbf{Inter_2} \in \mathbb{R}^{1 \times H \times U}$ consistent with the first interaction setting. Different interaction results are concatenated along the channel axis and used as convolution inputs, i.e., $\mathbf{I} = [\mathbf{Inter_1}; \mathbf{Inter_2}]$, where $\mathbf{I} \in \mathbb{R}^{2 \times H \times U}$.

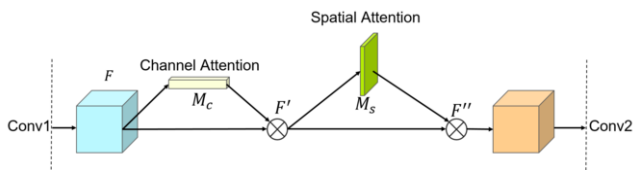

Figure 2. A simplified demonstration of how to introduce CBAM between two convolution layers.

3.4 Interactive Feature Extraction Module

To extract the interactive features between channels or within channels, we use two convolutional layers and for the first time innovatively introduce a convolutional block attention module (CBAM) [19] originally proposed in computer vision, and propose a new interactive feature extraction module as shown in **Figure 1**.

Generally, a pooling operation is performed between two layers of convolution, but the pooling operation will cause some information loss, which is not conducive to our link prediction task. Therefore, CBAM is adopted between the two convolutional layers for further feature extraction, utilizing the *channel attention* mechanism and the *spatial attention* mechanism as shown in **Figure 2**. $\mathbf{M}(\boldsymbol{h}, \boldsymbol{r})$ is the result of extracting features by two layers of convolution as follows:

$$\begin{aligned} \mathbf{F} &= \mathrm{BN}(f_1^{7\times7}(\mathbf{I})) \\ \mathbf{M}(\boldsymbol{h}, \boldsymbol{r}) &= f_2^{7\times7}(\mathbf{CBAM}(\mathbf{F}))) \end{aligned} \tag{4}$$

where $f_1^{7\times7}$ and $f_2^{7\times7}$ denote two 7×7 convolutional layers, $\mathbf{F} \in$

$\mathbb{R}^{C_1 \times H \times U}$, $\mathbf{M}(\boldsymbol{h}, \boldsymbol{r}) \in \mathbb{R}^{C_2 \times H \times U}$, C_1 and C_2 denote the number of $f_1^{7\times7}$ and $f_2^{7\times7}$ convolutional filters respectively. To facilitate data manipulation, we add the *padding* operation to the convolutional layers to maintain the feature map dimension fixed.

CBAM sequentially infers a 1D channel attention map $\mathbf{M_c} \in \mathbb{R}^{C_1 \times 1 \times 1}$ and a 2D spatial attention map $\mathbf{M_s} \in \mathbb{R}^{1 \times H \times U}$ over feature map $\mathbf{F}$. The overall attention process can be summarized as:

$$\begin{aligned} \mathbf{F}' &= \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F} \\ \mathbf{F}'' &= \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}' \end{aligned} \tag{5}$$

where $\otimes$ denotes element-wise multiplication.

For *channel attention* module, we produce a channel attention map by exploiting the inter-channel relationship of features. To compute the channel attention, we apply average-pooling and max-pooling operations on feature map. $\mathbf{F_c^{avg}}$ and $\mathbf{F_c^{max}}$ denote average-pooling features and max-pooling features, respectively. The channel attention module utilizes the multi-layer perceptron (MLP) to extract channel features. The number of channels is compressed to $1/r$ times the original number, and then is extended to the original number of channels, where r is the reduction ratio. In short, the channel attention is computed as:

$$\begin{aligned} \mathbf{M_c}(\mathbf{F}) &= \sigma(\mathrm{MLP}(\mathrm{AvgPool}(\mathbf{F})) + \mathrm{MLP}(\mathrm{Max\,Pool}(\mathbf{F}))) \\ &= \sigma\left(\mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F_c^{avg}}\right)\right) + \mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F_c^{max}}\right)\right)\right) \end{aligned} \tag{6}$$

where σ denotes the *tanh* function, $\mathbf{W_0} \in \mathbb{R}^{C_1/r \times C_1}$, and $\mathbf{W_1} \in \mathbb{R}^{C_1 \times C_1/r}$. Note that the MLP weights, $\mathbf{W}_0$ and $\mathbf{W}_1$, are shared for both inputs.

For *spatial attention* module, we generate a spatial attention map by utilizing the inter-spatial relationship of features. To compute the spatial attention, we apply average-pooling and max-pooling operations along the channel axis and concatenate them to generate an efficient feature map. Through the use of two pooling procedures, we aggregate channel information of a feature map to generate two 2D maps: $\mathbf{F_s^{avg}} \in \mathbb{R}^{1 \times H \times U}$ and $\mathbf{F_s^{max}} \in \mathbb{R}^{1 \times H \times U}$. They are then concatenated and convolved by a standard convolution layer, producing a 2D spatial attention map. The spatial attention is computed as:

$$\begin{aligned} \mathbf{M_s}(\mathbf{F}) &= \sigma\left(f^{7\times7}([\mathrm{AvgPool}(\mathbf{F}); \mathrm{MaxPool}(\mathbf{F})])\right) \\ &= \sigma\left(f^{7\times7}\left([\mathbf{F_s^{avg}}; \mathbf{F_s^{max}}]\right)\right) \end{aligned} \tag{7}$$

where $f^{7\times7}$ denotes the convolution operation.

3.5 Prediction Module

The prediction module is used to predict missing entities. We perform semantic similarity calculations with candidate entities by extracting head entity and relation interaction features. If the triple is correct, the semantic similarity is higher.

We perform a linear transformation on the result $\mathbf{M}(\boldsymbol{h}, \boldsymbol{r})$ of the interactive feature extraction module so that its dimension is consistent with that of the candidate entity $\boldsymbol{t}$. The tensor $\mathbf{M}(\boldsymbol{h}, \boldsymbol{r})$ is then reshaped into a vector $vec(\mathbf{M}(\boldsymbol{h}, \boldsymbol{r})) \in \mathbb{R}^{C_2 HU}$ which is then projected into a d-dimensional space using a linear transformation parameterized by the matrix $\mathbf{W} \in \mathbb{R}^{C_2 HU \times d}$.

Finally, the scoring function $\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$ for the ConvEICF method after the nonlinear convolution is defined as below:

$$\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t}) = \mathrm{ReLU}(vec(\mathbf{M}(\boldsymbol{h}, \boldsymbol{r}))\mathbf{W})\boldsymbol{t} \tag{8}$$

where $\boldsymbol{t} \in \bar{\mathbf{E}}$ is used to evaluate all candidate entities.

We regularise our model by *using dropout in several stages* as shown in Figure 1. In particular, we perform dropout on the input of the convolutional layer (i.e., the result of entity and relation interaction step), the feature maps after the convolution operation (i.e., the feature map after the last convolutional layer), and the fully connected hidden unit.

3.6 Training

During training, we exploit the InfoNCE loss to train model parameters for better results:

$$\mathcal{L} = -\log \frac{e^{\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})/\boldsymbol{\tau}}}{\sum_{i=1}^{|\mathcal{E}|} e^{\phi\left(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t_i'}\right)/\boldsymbol{\tau}}} \tag{9}$$

where $\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})$ is the semantic similarity score for the positive triple, $\phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t_i'})$ is the semantic similarity score for a candidate triple, $\boldsymbol{\tau}$ denotes a temperature parameter, which can adjust the relative importance of negatives, and smaller $\boldsymbol{\tau}$ makes the loss put more emphasis on hard negatives.

4 Experiments

In this section, we conduct extensive experiments to evaluate our proposed ConvEICF model.

4.1 Datasets and Experiment Settings

Two widely used benchmark datasets are used in our link prediction experiments: FB15k-237 [14] and WN18RR [5]. Link prediction is to predict the missing h or t for a correct triple (h, r, t), i.e., predict t given h, r or predict h given t, r^{-1}. We report several standard evaluation metrics: the Mean of those predicted Reciprocal Ranks (MRR), and the Hits@N (i.e., the proportion of correct entities ranked in the top N, where N = 1, 3, 10). Higher MRR and Hits@N mean better performance. We use the filtered setting, while evaluating on test triples. We filter out all the valid triples from the candidate set, which is generated by either corrupting the head or tail entity of a triple. We use the standard evaluation metrics to evaluate the effect of predicting the head entity and tail entity, and average them as the final results.

In our experiments, we fix mini-batch as 256, initial learning rate to 0.0005, and label smoothing coefficient to 0.1. We set the dropout threshold in LTE to 0.3. On each dataset, we choose the optimal configuration with the highest MRR on the validation set within 500 epochs and report its performance on the test set. In the experiment of this paper, the highest MRR scores are obtained when using $d = 200$, $\tau = 0.007$, $C_1 = 64$, $C_2 = 200$, input drop $= 0.2$, feature drop $= 0.2$, hidden drop $= 0.2$ on FB15k-237, and $d = 200$, $\tau = 0.001$, $C_1 = 64$, $C_2 = 250$, input drop $= 0.2$, feature drop $= 0.1$, hidden drop $= 0.4$ on WN18RR. LTE performs only on entities on the FB15k-237 and on entities and relations on the WN18RR.

4.2 Baselines

We compare ConvEICF against a variety of competitive baselines that can be divided into three groups:

- The Translation Methods: TransE [4], RotatE [13], where RotatE is a quite powerful KGE model.

Table 1. Link prediction results on WN18RR and FB15k-237 test set. † denotes that the experimental data derived from our reproduction experiments, which have better results than them in the original papers. Other data derived from their original papers or related papers. The best scores highlighted in bold, and the second ones in underline.

Models	FB15k-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE [4]	0.332	0.240	0.368	0.516	0.205	0.022	0.347	0.519
RotatE [13]	0.338	0.241	0.375	0.533	<u>0.476</u>	0.428	<u>0.492</u>	**0.571**
DistMult [20]	0.279	0.202	0.306	0.433	0.410	0.389	0.420	0.450
TuckER† [1]	<u>0.358</u>	<u>0.267</u>	0.391	<u>0.541</u>	0.470	0.433	0.482	0.526
ConvE† [5]	0.321	0.233	0.351	0.498	0.462	0.431	0.473	0.526
AcrE(Parallel)† [10]	0.356	0.264	0.391	<u>0.541</u>	0.459	0.422	0.473	0.532
CTKGC† [6]	0.330	0.240	0.362	0.512	0.459	0.426	0.472	0.521
InteractE† [15]	0.355	0.263	<u>0.392</u>	0.539	0.463	0.430	-	0.528
HypER† [2]	0.341	0.252	0.376	0.520	0.465	<u>0.436</u>	0.477	0.522
CompGCN† [16]	0.354	0.262	0.388	0.534	0.467	0.429	0.480	0.542
SACN [11]	0.350	0.260	0.390	0.540	0.470	0.430	0.480	0.540
LTE-ConvE† [22]	0.351	0.260	0.382	0.532	0.471	0.434	0.484	0.542
ConvEICF(Ours)	**0.365**	**0.271**	**0.401**	**0.554**	**0.486**	**0.446**	**0.501**	<u>0.560</u>

- The Bilinear Models: DistMult [20], TuckER [1], where TuckER improves DistMult and attains competitive results for link prediction.
- The Neural Network Methods including competitive CNN-based models: ConvE [5], SACN [11], CompGCN [16], AcrE [10], InteractE [15], LTE-ConvE [22], CTKGC [6].

4.3 Experimental Results

Table 1 shows the experimental results of our model ConvEICF and baselines. From the results, we can see that:

1. Experiments show that our proposed model ConvEICF is signingicantly effective for link prediction and *consistently outperforms* all the state-of-the-art methods on FB15k-237. On Hits@10, ConvEICF is 5.6% higher than ConvE, 2.2% higher than LTE-ConvE, 1.3% higher than TuckER and 2.1% higher than RotatE.
2. On WN18RR, our model also *achieves better performance* than other methods and obtains state-of-the-art results compared to CNN-based models. On Hits@10, ConvEICF is 3.4% higher than ConvE, 1.8% higher than LTE-ConvE, 3.4% higher than TuckER and 3.2% higher than InteractE. Compared with the RotatE model, except Hits@10, other evaluation metrics are significantly improved, with an average improvement of about 1.2%.
3. It can be seen from **Figure 3** that our proposed model ConvEICF has *faster training speed* and better model performance compared with GCN-based models (e.g., SACN and CompGCN). And our model has an average improvement of 1.5% in all metrics without using GCNs to aggregate entity adjacent node information.

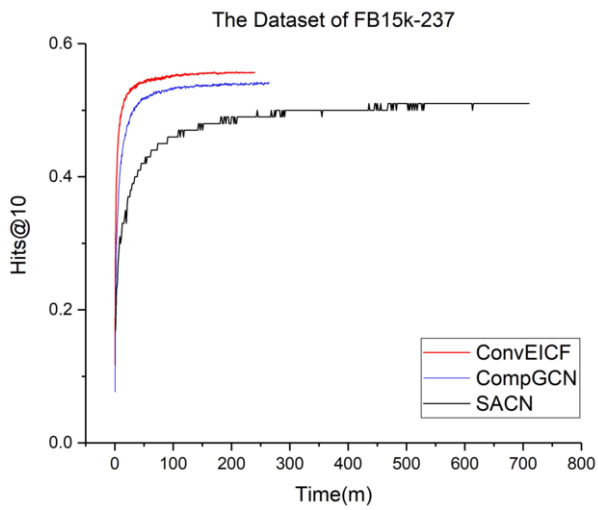

Figure 3. The Hits@10 and the running time with ConvEICF, CompGCN, and SACN for the link prediction tasks on FB15k-237.

5 Anaylses

5.1 Ablation Study

We design a set of ablation experiments to investigate the influence of main modules in ConvEICF, and the experimental results are shown in **Table 2**. Our model is mainly optimized in four aspects: entity and relation embeddings, entity-to-relation interaction, CNN structure optimization, and loss function. Therefore, we will explore the impact of the above *four aspects* on ConvEICF.

For entity and relation embeddings, we would like eliminate LTE, abbreviated as **Without LTE**. There are two kinds of entity-to-relation interaction approaches as convolution inputs, and we ablate each interaction in turn, which is referred to as **Without $\mathbf{Inter_1}$** and **Without $\mathbf{Inter_2}$** for short. In terms of CNN structure, the attention mechanism CBAM is removed, referred to as **Without CBAM**. To study the impact of single-layer convolution, it is necessary to remove the first convolutional layer and CBAM, abbreviated as **Without CBAM and Conv**. We replace the loss function with the conventional method BCE loss function, referred to as **Without InfoNCE (BCE)**.

The results of our ablation experiments are shown in Table 2. We find that: (*i*) LTE has a significant impact on ConvEICF. For exam-

Table 2. Ablation experiment results.

Model Name	FB15k-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Without LTE	0.335	0.242	0.368	0.524	0.470	0.439	0.484	0.529
Without $\mathbf{Inter_1}$	0.355	0.262	0.393	0.544	0.482	0.443	0.499	0.556
Without $\mathbf{Inter_2}$	0.359	0.266	0.393	0.548	0.468	0.430	0.485	0.539
Without CBAM	0.364	0.270	0.400	0.553	0.480	0.442	0.495	0.558
Without CBAM and Conv	0.363	0.269	0.399	0.548	0.482	0.445	0.495	0.554
Without InfoNCE (BCE)	0.353	0.262	0.386	0.535	0.472	0.429	0.491	0.550
ConvEICF	**0.365**	**0.271**	**0.401**	**0.554**	**0.486**	**0.446**	**0.501**	**0.560**

Table 3. Link prediction results by relation category on FB15k-237 under MRR.

	RotatE	ConvE	InteractE	ConvEICF
1-1	**0.491**	0.370	0.377	0.415
1-N	0.421	0.427	**0.442**	0.294
N-1	0.273	0.257	0.270	**0.459**
N-N	0.313	0.318	0.336	**0.343**

Table 4. The link prediction results of the ConvE and ConvEICF for LTE exploratory experiment. In the table, ℓ, $\mathbf{b}$, and $\mathbf{d}$ represent the linear layer, BatchNorm, and dropout operations, respectively. $\mathbf{Null}$ means that the above operations are not taken.

Operation	ConvE		ConvEICF	
	MRR	Hits@10	MRR	Hits@10
Null	0.321	0.498	0.335	0.524
ℓ	0.319	0.494	0.337	0.523
$\mathbf{b}$	0.315	0.491	0.337	0.524
$\mathbf{d}$	**0.355**	**0.536**	**0.360**	**0.548**

ple, ConvEICF performs better than "**Without LTE**" with 3% and 1.6% improvements on FB15k-237 and WN18RR under MRR, respectively. (*ii*) We discover that different datasets and interaction approaches have different effects on ConvEICF. For instance, $\mathbf{Inter_1}$ has a significat effect in FB15k-237, whereas $\mathbf{Inter_2}$ has a significat effect in WN18RR. FB15K-237 has more complex relations and fewer entities than WN18RR. We reckon $\mathbf{Inter_1}$ is more suitable for more complex datasets such as FB15k-237, while $\mathbf{Inter_2}$ is the opposite. (*iii*) Comparing "**Without CBAM**" and "**Without CBAM and Conv**", we find that the CBAM attention mechanism and two-layer convolution can effectively improve ConvEICF for link prediction. (*iv*) Comparing different loss functions for ConvEICF, the InfoNCE loss is significantly more effective than the BCE loss function with 1.2% and 1.4% improvements under MRR on FB15k-237 and WN18RR, respectively. **Figure 4** shows the Hits@10 value of ConvEICF with different loss functions on two datasets during training. Compared with the BCE loss function, the InfoNCE loss function has a faster training speed and can achieve better results. In the initial stage of the InfoNCE loss, the Hits@10 metrics grow fast and the training speed is faster.

Moreover, we further analyze the performance of ConvEICF on various relation categories on FB15k-237. We choose FB15k-237 for analysis due to its diverse and extensive set of relations. Following [18], we divide the relation categories into four groups based on the average number of tails per head and heads per tail: one-to-one (1-1), one-to-many (1-N), many-to-one (N-1), and many-to-many (N-N). For comparison, we select several representative models for different types of relations and evaluate them using the MRR metric. The comparison results are presented in **Table 3**. We observe that ConvEICF outperforms other models for N-1 and N-N types of relations, demonstrating its ability to handle complex relation categories.

5.2 The effects of Linear Layer, BatchNorm and Dropout

As shown in Equation (1), our embedding enhancement module based on LTE consists of *three operations*: linear layer (ℓ), BatchNorm ($\mathbf{b}$), and dropout ($\mathbf{d}$). We go deeper to investigate the effects of the above three operations on the CNN-based models.

Firstly, using ConvE and ConvEICF as basic models, we carry out extensive experiments on FB15k-237 to explore which of the above operations plays a decisive role in achieving good performance. On FB15k-237, we perform LTE on the entities embedding matrix to achieve better performance. We perform the above three operations in turn, and the experimental results are shown in **Table 4**.

From Table 4, we find that among the three operations involved in LTE, dropout plays a decisive role. "**d**" performs better than "**Null**" in ConvE and ConvEICF with 3.8% and 2.4% improvements under Hits@10, respectively. The performance of ConvE and ConvEICF decline if only linear layers or BatchNorm operations are used. The results show that *the performance of the model can be improved by simply performing a dropout operation on the entity embedding matrix*.

Further, in order to fully explore the above observation, considering that ConvE and ConvEICF adopt a 1-N scoring method, we explore whether the additional dropout affects other models based on the 1-N scoring method (e.g., AcrE, CTKGC, HypER, TuckER, and InteractE) on FB15k-237. Dropout operation is added directly to the entity embedding matrix of other models as ConvE does, but several models do not perform better. We discover that the additional dropout placement position and the existing dropout in these models have a large impact on the performance. We consider the dropout threshold more than 0.3 to be a large weight, and vice versa.

From **Table 5**, we discover that adding dropout operations to these 1-N scoring models significantly improves the performance of these models. For example, CTKGC is improved by at most 2.2%, while TuckER is improved at least by 0.5% under Hits@10. The evaluation metrics that are not improved in these models are kept consistent with the original model, such as InteractE on Hits@3. From Table 5, we find some interesting phenomena:

1. For HypER and TuckER, input dropout is performed on the

Table 5. The experiments further explore whether the dropout affects other models based on the 1-N scoring method. **Original** means to reproduce the experimental results of the original paper. **BDrop** is a dropout operation on the entity matrix. **ADrop** is to perform dropout operation on the candidate entities of the 1-N scoring method. **Reset** modifies the dropout threshold existing in the model. **Dropout Rate** is used to represent input dropout, feature dropout, hidden dropout in order.

Model	Category	Dropout Rate	Addition Dropout	MRR	Hit@1	Hit@3	Hit@10
AcrE(Parallel)	Original	0.3-0.2-0.5	-	0.356	0.264	0.391	0.541
	BDrop	0.3-0.2-0.5	0.1	0.363	0.270	0.398	0.546
	Reset&BDrop	0.3-0.2-0.3	0.3	**0.365**	**0.271**	**0.400**	**0.550**
CTKGC	Original	0.2-0.2-0.5	-	0.330	0.240	0.362	0.512
	BDrop	0.2-0.2-0.5	0.1	0.342	0.252	0.377	0.524
	Reset&BDrop	0.2-0.2-0.2	0.3	**0.352**	**0.261**	**0.389**	**0.534**
HypER	Original	0.3-0.2-0.3	-	0.335	0.245	0.367	0.516
	ADrop	0.3-0.2-0.3	0.3	**0.355**	**0.264**	**0.390**	**0.537**
TuckER	Original	0.3-0.4-0.5	-	0.358	**0.267**	0.391	0.541
	Reset&ADrop	0.3-0.3-0.3	0.3	**0.359**	0.266	**0.393**	**0.546**
InteractE	Original	0.2-0.5-0.5	-	0.355	0.263	**0.392**	0.539
	Reset&BDrop	0.2-0.3-0.3	0.3	**0.357**	**0.265**	**0.392**	**0.546**

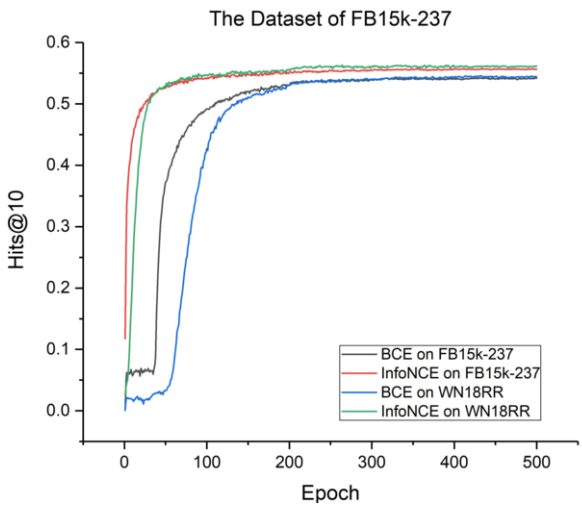

Figure 4. In ConvEICF with different loss functions, the Hits@10 evaluation metrics change with epoch during training.

head entity vector and we utilize **ADrop** rather than **BDrop**. **BDrop** will cause the input dropout threshold of these two models to be too large, which affects the generalization ability.

2. By looking at the **Dropout Rate**, TuckER and InteractE use two dropouts with large weights to make the model have sufficient generalization ability. Adding dropout again will make the model learn too few parameters, making it difficult to learn important features. Therefore, we need to perform the **Reset** operation and then add the dropout operation for TuckER and InteractE.
3. CTKGC and AcrE both have one large weight dropout, indicating that their generalization abilities are slightly residual. The model evaluation metrics can be improved by adding dropout in two different ways: **BDrop** and **Reset&BDrop**. However, **BDrop** can only increase the dropout by 0.1 weight, thus the model can only be slightly improved. **Reset&BDrop** method modifies the dropout existing in these models to a small weight, and adding one dropout significantly improves the performance of the two models.

To sum up, the experiments show that the very common 1-N scoring technique in KGE can be considerably improved by just adding a dropout operation. We argue that this may be an important finding, which will be instructive for subsequent models to further improve link prediction performance.

6 Conclusion

In this paper, we investigate and discover four key aspects that have a strong influence on link prediction. Therefore, we propose a novel KGE model for link prediction, namely ConvEICF. Our proposed ConvEICF is based on ConvE and optimizes from four aspects: entity and relation embeddings, entity-to-relation interaction, CNN structure optimization, and loss function. We provide extensive experiments and exhaustive empirical analyses to verify the effectiveness and efficiency for the ConvEICF model. The experiment results show that our ConvEICF significantly improves in all evaluation metrics compared to previous CNN-based models. Further exploration experiments show that the common 1-N scoring technique in KGE can be considerably improved by just adding a dropout operation, which may be instructive for subsequent models to further improve link prediction performance. In the future, we plan to explore the issue of parameters of the model and the other aspects related to the KGE for link prediction.

Acknowledgements

We would like to thank the referees for their comments, which helped improve this paper considerably. The work is supported by the National Natural Science Foundation of China (62276057), and the Fundamental Research Funds for the Central Universities (No. N2216008).

References

[1] Ivana Balazevic, Carl Allen, and Timothy Hospedales, 'TuckER: Tensor factorization for knowledge graph completion', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5185–5194, (2019).

[2] Ivana Balaž ević, Carl Allen, and Timothy M. Hospedales, 'Hypernetwork knowledge graph embeddings', in *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, pp. 553–565, (2019).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, 'Freebase: A collaboratively created graph database for structuring human knowledge', in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pp. 1247–1250, (2008).

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings for modeling multi-relational data', in *Advances in Neural Information Processing Systems (NeurIPS)*, volume 26, (2013).

[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel, 'Convolutional 2d knowledge graph embeddings', in *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 32, pp. 1811–1818, (2018).

[6] Jianzhou Feng, Qikai Wei, Jinman Cui, and Jing Chen, 'Novel translation knowledge graph completion model based on 2d convolution', *Applied Intelligence*, **52**(3), 3266–3275, (2022).

[7] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao, 'An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge', in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 221–231, (2017).

[8] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu, 'Squeeze-and-excitation networks', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, (2018).

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S. Auer, and Christian Bizer, 'Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia', *Semantic Web*, **6**, 167–195, (2015).

[10] Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, and Yujia Bai, 'Knowledge graph embedding with atrous convolution and residual learning', in *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pp. 1532–1543, (2020).

[11] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou, 'End-to-end structure-aware convolutional networks for knowledge base completion', in *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 33, pp. 3060–3067, (2019).

[12] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum, 'Yago: A core of semantic knowledge', in *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pp. 697–706, (2007).

[13] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang, 'Rotate: Knowledge graph embedding by relational rotation in complex space', in *International Conference on Learning Representations (ICLR)*, (2019).

[14] Kristina Toutanova and Danqi Chen, 'Observed versus latent features for knowledge base and text inference', in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pp. 57–66, (2015).

[15] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar, 'Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions', in *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 34, pp. 3009–3016, (2020).

[16] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar, 'Composition-based multi-relational graph convolutional networks', in *International Conference on Learning Representations (ICLR)*, (2019).

[17] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu, 'SimKGC: Simple contrastive knowledge graph completion with pre-trained language models', in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4281–4294, (2022).

[18] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen, 'Knowledge graph embedding by translating on hyperplanes', in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1112–1119, (2014).

[19] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, 'Cbam: Convolutional block attention module', in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, (2018).

[20] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng, 'Embedding entities and relations for learning and inference in knowledge bases', in *Proceedings of the International Conference on Learning Representations (ICLR)*, (2015).

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

[22] Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu, 'Rethinking graph convolutional networks in knowledge graph completion', in *Proceedings of the ACM Web Conference (WWW)*, pp. 798–807, (2022).

[23] Yunxiang Zhao, Jianzhong Qi, Qingwei Liu, and Rui Zhang, 'WGCN: Graph convolutional networks with weighted structural features', in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 624–633, (2021).