Information Processing and Management 60 (2023) 103350

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Knowledge graph embedding model with attention-based high-low level features interaction convolutional network

Jingxiong Wang [a], Qi Zhang [b], Fobo Shi [c,*], Duantengchuan Li [a,*], Yuefeng Cai [d], Jian Wang [a], Bing Li [a], Xiaoguang Wang [e], Zhen Zhang [e,f], Chao Zheng [a]

[a] School of Computer Science, Wuhan University, Wuhan, China
[b] School of Information Management, Central China Normal University, Wuhan, China
[c] National Engineering Research Center for E-Learning, Central China Normal University, Wuhan, China
[d] ZTE Corporation, Wuhan, China
[e] School of Information Management, Wuhan University, Wuhan, China
[f] School of Sports Engineering and Information Technology, Wuhan Sports University, Wuhan, China

ARTICLE INFO

Keywords:
Link prediction
Knowledge graph embedding
Convolutional neural network
Criss-cross attention mechanism
High-low level features interaction

ABSTRACT

Knowledge graphs are sizeable graph-structured knowledge with both abstract and concrete concepts in the form of entities and relations. Recently, convolutional neural networks have achieved outstanding results for more expressive representations of knowledge graphs. However, existing deep learning-based models exploit semantic information from single-level feature interaction, potentially limiting expressiveness. We propose a knowledge graph embedding model with an attention-based high-low level features interaction convolutional network called ConvHLE to alleviate this issue. This model effectively harvests richer semantic information and generates more expressive representations. Concretely, the multilayer convolutional neural network is utilized to fuse high-low level features. Then, features in fused feature maps interact with other informative neighbors through the criss-cross attention mechanism, which expands the receptive fields and boosts the quality of interactions. Finally, a plausibility score function is proposed for the evaluation of our model. The performance of ConvHLE is experimentally investigated on six benchmark datasets with individual characteristics. Extensive experimental results prove that ConvHLE learns more expressive and discriminative feature representations and has outperformed other state-of-the-art baselines over most metrics when addressing link prediction tasks. Comparing MRR and Hits@1 on FB15K-237, our model outperforms the baseline ConvE by 13.5% and 16.0%, respectively.

1. Introduction

Knowledge graphs (KGs) are proposed to model knowledge in the real world, which are often assembled from numerous sources (Ji, Pan, Cambria, Marttinen, & Yu, 2022). In KGs, copious structural information is stored in the form of (head entity, relation, tail entity), noted as (h, r, t). For instance, Mike Tyson was born on June 30, 1966, which can be represented as (Mike Tyson, Birth_in, June 30, 1966). As a matter of inevitably missing or erroneous relations and entities, imperfections and inconsistencies are the main challenges in KGs. Hence, recent studies have paid significant attention to the link prediction task. That is, knowing the head entity and relation to find corresponding tail entities or knowing the tail entity and relation to infer appropriate head entities.

* Corresponding authors.
E-mail addresses: foboshi99@gmail.com (F. Shi), dtclee1222@whu.edu.cn (D. Li).

https://doi.org/10.1016/j.ipm.2023.103350
Received 26 October 2022; Received in revised form 4 February 2023; Accepted 13 March 2023
Available online 27 March 2023

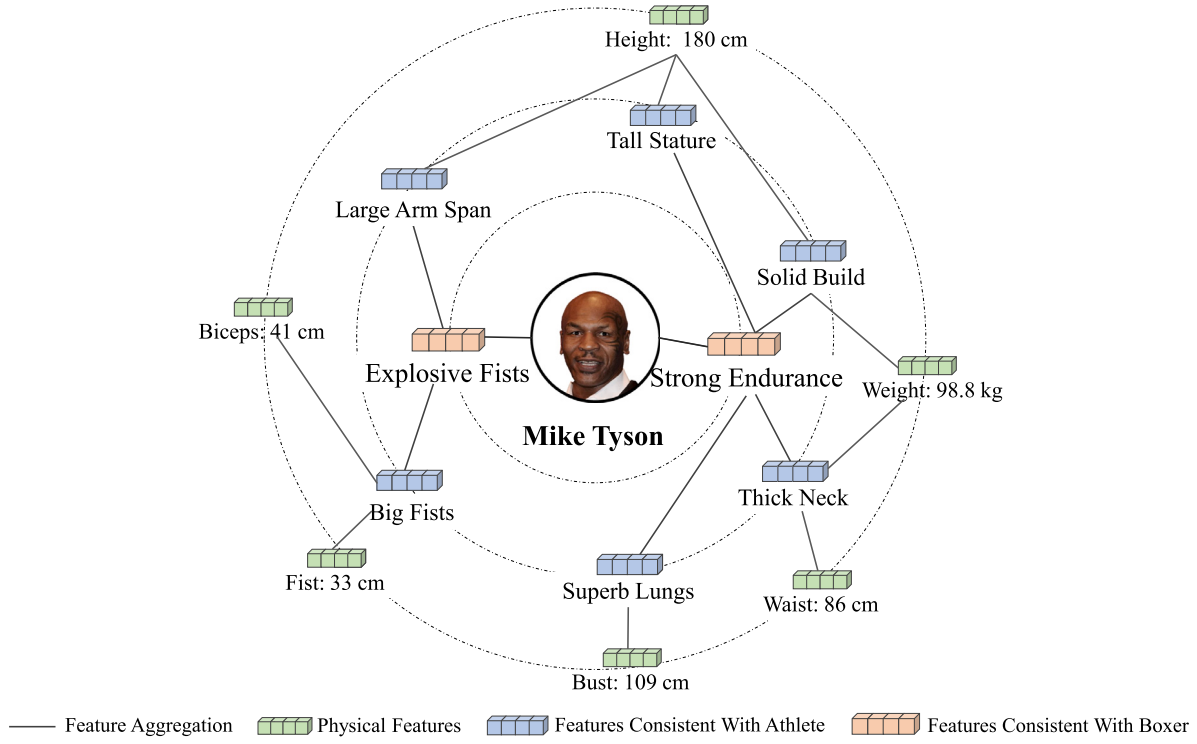

Fig. 1. The illustration of the feature hierarchy in KGs by analyzing Tyson's physical features.

In recent studies, knowledge graph embedding (KGE) is used for link prediction (Xiao, Chen, & Shi, 2021; Xie, Zhu, Liu, Zhou, & Huang, 2022; Zhang, Huang, Gao, Han, & Zhou, 2022), which intends to learn a dense graph representation in a continuous, low dimensional vector. Various KGE models can be roughly classified into the translational distance (Bordes, Usunier, García-Durán, Weston, & Yakhnenko, 2013; Ji, He, Xu, Liu, & Zhao, 2015; Lin, Liu, Sun, Liu, & Zhu, 2015; Wang, Zhang, Feng, & Chen, 2014), semantic matching (Nickel, Tresp, & Kriegel, 2011; Trouillon, Welbl, Riedel, Gaussier, & Bouchard, 2016; Yang, Yih, He, Gao, & Deng, 2015), and convolutional neural network-based models (Dettmers, Minervini, Stenetorp, & Riedel, 2018; Jiang, Wang, & Wang, 2019; Vashishth, Sanyal, Nitin, & Talukdar, 2020). Although translational distance and semantic matching methods have achieved good performance, the model structures primarily employ linear transformations, which limits the learning of the complex relationships in link prediction.

With the remarkable achievements of the convolutional neural network (CNN) in recommendation systems (Isufi, Pocchiari, & Hanjalic, 2021), natural language processing (Arenas-Márquez, del Rocío Martínez Torres, & Toral, 2021), and intelligent education (Li, Wang, Du, Hu, & Yang, 2023), the use of CNNs in KGE has attracted considerable attention and generated highly expressive feature embeddings. Typically, these CNN-based KGE models (Dettmers et al., 2018; Jiang et al., 2019; Vashishth, Sanyal, et al., 2020) perform single convolution operations directly after concatenating entities. Although existing CNN-based models only focus on leveraging single-level heterogeneous features for interactive learning, they ignore the feature hierarchy in the knowledge graph.

There are high and low levels of structural information in KGs (Lin et al., 2017). The knowledge graph centered on Mike Tyson is a vivid illustration. As shown in Fig. 1, we can obtain the following low-level features: Tyson is 180 cm tall, weighs 98.8 kg, has biceps of 41 cm, bust size of 109 cm, waist girth of 86 cm, etc., which are shallow but concrete, represented by d-dimensional vectors. At a higher level, we can sketch the physical qualities of Tyson: He is tall and solid, with a large arm span and superb lungs, which coincide with athletes. If we go to deeper feature extraction, we can learn high-level features that express more semantic information: Tyson has strong endurance, and his fists are explosive and fast. These abstract features are more confirmed with the identity of a heavyweight boxing professional athlete. In this view, existing convolutional models extract only a single level of features, ignoring the natural feature hierarchy in KGs, constraining the expressiveness.

Previous studies (Simonyan & Zisserman, 2015) have demonstrated that deep convolution helps to improve the expressiveness of the model. And though relatively weak in the scale of the receptive field, shallow features generated by shallower convolution are useful for sensing local information concretely. To leverage the full potential of the feature hierarchy in mining richer and more multidimensional semantic information, multilevel features fusion is a new inspiration to boost the quality of the representations and empower KGE models with more robust performance. In addition, considering that the features of different levels contribute differently to the expression of the model, the attention mechanism is an effective solution. To measure the relationships between

each feature, traditional attention mechanisms must generate substantial attention maps, resulting in high temporal and spatial complexity. Previous studies have found that consecutive sparsely-connected structure requires much lower computational resources while increasing additional feature interactions. Therefore, the attention mechanism with this special structure can be used to expand the receptive field of each feature for more efficient interaction performance in KGE.

Hence, we propose a CNN-based model (ConvHLE) which is capable of fusing multilevel features. We use a multi-layer convolutional neural network to fuse high and low-level features, and then use the criss-cross attention mechanism to redistribute the weights of different levels of features, which greatly reduces the model's computational complexity and memory overhead. Finally, the feature maps are projected into the corresponding embedding dimension for the link prediction task. Comprehensive experiments demonstrate that ConvHLE can effectively fuse multilevel features, benefit the efficiency of interaction, and improve the expressiveness of embeddings. Moreover, ConvHLE presents a novel insight into the feature hierarchy in KGs, with both theoretical and practical significance.

1.1. Research objectives and contributions

To break the conventional mode of single-level feature interaction in existing CNN-based KGE models, we aim to introduce the high-low level features interaction to improve the expressiveness of the model. Furthermore, in addition to learning high-low level features interactions, we also enable each feature to perceive neighbor information through a new attention mechanism. Based on the above insights, we build a knowledge graph embedding model with attention-based high-low level features interaction convolutional network, which incorporates multilevel features for richer semantic information. We introduce the criss-cross attention mechanism based on consecutive sparsely-connected graphs to expand the receptive field and reduce model parameters. Ultimately, our work aims to achieve two goals: (1) feature fusion and interaction, in which features from multilevel can interact synergistically to represent more abundant semantic information, and (2) attention mechanism, in which the receptive field is expanded while utilizing lightweight computing and memory.

Our primary contributions might be summarized as follows:

- A knowledge graph embedding model with attention-based high-low level features interaction convolutional network called ConvHLE is proposed, which incorporates multilevel features for richer semantic information and more accurate embedding representations.
- A novel criss-cross attention mechanism based on consecutive sparsely-connected graphs is proposed to expand the receptive field, increase additional interactions, boost the quality of feature interaction, and reduce model parameters.
- A set of experiments are conducted on six public datasets to prove the capabilities of our proposed model. Experimental results demonstrate that ConvHLE shows state-of-the-art performance on link prediction tasks.

The remainder of this essay is structured as follows. We first give a summary and analysis of the relevant KGE works in Section 2. The architecture of the ConvHLE is introduced in Section 3 after defining research problems. In Section 4, comprehensive experiments on six benchmark datasets are given and analyzed in detail. Section 5 provides further discussions and implications. The conclusion and the assumption for future work are represented in Section 6.

2. Related work

The KGE models usually employ a scoring function to calculate triplets for link prediction tasks, which can assess the model's performance. KGE models can be roughly classified into the translational distance, semantic matching, and convolutional network models.

2.1. Translational distance models

TransE (Bordes et al., 2013) pioneers translational distance models, which considers the relation as a translation operation from the head entity to the tail entity in a low-dimensional vector space. For all positive samples, TransE learns vectors $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ to minimize the vector distance between $(\mathbf{h}+\mathbf{r})$ and $\mathbf{t}$. For negative samples, TransE tries to learn representations that keep $\mathbf{h}+\mathbf{r}$ far away from $\mathbf{t}$. The loss function is as below:

$$\mathcal{L} = \sum_{(h,r,t)\cup(h',r,t')} max(0, \|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_p - \|\mathbf{h}'+\mathbf{r}-\mathbf{t}'\|_p + \gamma), \tag{1}$$

where p means the L_1 or L_2 norm. $max(0,x)$ returns the higher value between 0 and x. γ represents the hyperparameter margin. (h',r,t') represents the negative triplet sample. TransE is a simple and swift method of knowledge graph embedding. Nevertheless, TransE has restricted capacity to handle complex relations. To improve it, a host of variants have been proposed. TransH (Wang et al., 2014) projects head and tail entities into different hyperplanes for each relation. TransR (Lin et al., 2015) extends this approach by representing entities and relations in different spaces. For each relation r, it sets a matrix to project entities from entity space to relation space. In order to simplify TransR, TransD (Ji et al., 2015) uses a projection vector to associate entities and relations. Other similar works, including TorusE (Ebisu & Ichise, 2018) and RotatE (Sun, Deng, Nie, & Tang, 2019), have improved translational distance models in different aspects. Compared with deep, multi-layer models, translational distance models may not capture rich semantic information and complex relations.

2.2. Semantic matching models

Semantic matching models calculate the triplets (h, r, t) by mapping potential semantics in vector spaces. As a emblematic model, RESCAL (Nickel et al., 2011) represents the knowledge graph as a factorization of three-way tensor. DistMult (Yang et al., 2015) reduces the model parameters by restricting the relation matrix of the bilinear model to a diagonal matrix. Note that DistMult loses the ability to make asymmetric relations. As an extension of DistMult, ComplEx (Trouillon et al., 2016) copes with asymmetric relations by projecting the knowledge graph embedding to the complex space. SimplE (Kazemi & Poole, 2018) uses the inverse of the relation to handle the independence of the head entity vector and tail entity vector in canonical polyadic decomposition. Noteworthy, semantic matching models significantly increase redundancy over translational distance models. These models mainly deal with entities and relations through linear transformations, which may lead to overfitting when dealing with large-scale knowledge graphs.

2.3. Convolutional network models

The applications of the convolutional network are more recent approaches for KGE. ConvE (Dettmers et al., 2018) firstly applied the convolutional neural network to link prediction tasks. It uses convolutional kernels to extract nonlinear relationships in the knowledge graph but the interaction between entities and relations is not sufficient. As a similar model approach based on the convolutional neural network, HypER (Balazevic, Allen, & Hospedales, 2019a) generates a relation-specific convolution kernel matrix by a fully connected layer (hypernetwork), eliminating the need for vector-to-matrix transformation and outperforming ConvE. ConvR (Jiang et al., 2019) generates the semantic feature vectors by relations directly and utilizes them as the convolution kernels for entity convolution. Considering that the number of feature interactions that can be captured by ConvE is finite, InteractE (Vashishth et al., 2020) is proposed based on feature permutation, feature reshaping, and circular convolution. With the involvement of ResNet and Atrous Convolution (Chen, Papandreou, Schroff, & Adam, 2017), AcrE (Ren et al., 2020) intends to improve the model's effectiveness by adding more convolutional layers. Besides the aforementioned representative models, many excellent CNN-based models have also achieved satisfactory experimental results.

Convolutional neural network-based models are more expressive than the other two models mentioned previously. Moreover, the parameters of these models do not expand with the expansion of datasets, so it is more capable of modeling large-scale KGs. However, existing convolutional models also have limitations. According to the previous analysis, they only extract features at a single level, ignoring the feature hierarchy that naturally exists in KGs, which inevitably limits the expressiveness.

3. Methodology

To facilitate understanding, we explain the implementation of ConvHLE in detail, including problem formulation in Section 3.1, the architecture of ConvHLE in Section 3.2, and training and optimization in Section 3.3.

3.1. Problem formulation

In this paper, a knowledge graph is denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E} = \{e_1, e_2 \dots e_{|\mathcal{E}|}\}$ denotes the collection of entities. $\mathcal{R} = \{r_1, r_2, \dots, r_{|\mathcal{R}|}\}$ represents the set of all relations. $|\mathcal{E}|$ *and* $|\mathcal{R}|$ denotes the number of entities and relations. $\mathcal{T} \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ represents the set of triplets and one single triplet can be expressed as (h, r, t). The letters h, r, and t denote the head entity, relation, and tail entity respectively, and $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$ represent the k-dimension embeddings of head entity vector, relation vector, and tail entity vector respectively.

Link prediction was conducted during our experiments, which means that given any two parts of (h, r, t), predict the missing one. For example, given the head entity and relation, forecast the tail entity, denoted as $(h, r, ?)$. A pre-designed scoring function $\psi(h, r, t)$ was utilized to score the candidate triplets.

3.2. The architecture of ConvHLE

The architecture of ConvHLE is presented in this section. As shown in Fig. 2, ConvHLE involves high-low level features interaction, criss-cross attention mechanism, and link prediction. A multi-layer convolutional neural network is utilized to extract features at different levels. For the shallow feature map, the input features are first convolved, which is then employed as the input to perform more convolutions until three layers of feature maps are obtained. The three-layer feature map is concatenated for the feature fusion, which is achieved by a fully connected layer. The criss-cross attention mechanism is applied to reassign weights for each feature after interactions with informative neighbors. Finally, the processed feature map is flattened and projected to the same dimension of tail entities to predict the potential association between (h, r) and candidate t. The score of the triplet is calculated using the inner product of the processed feature map and the tail entity vector.

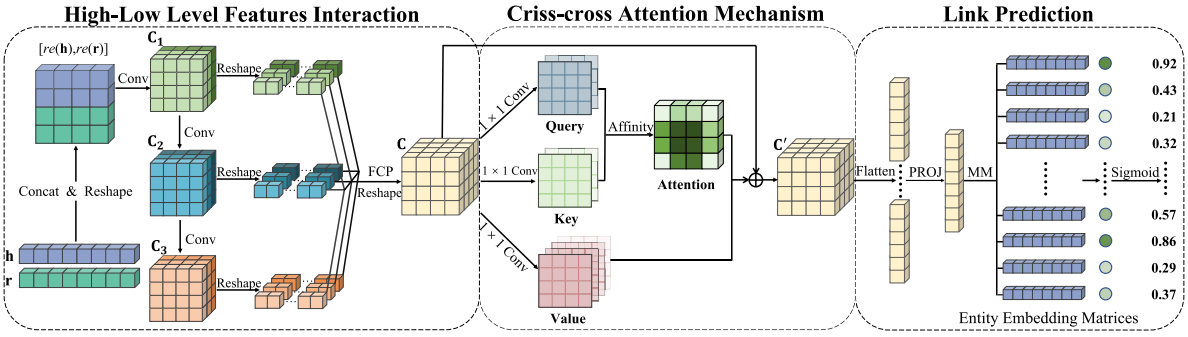

Fig. 2. The architecture of ConvHLE. **h** represents the head entity embeddings and **r** is the relation embeddings. $re(\cdot)$ represents the reshape operation. Concat is the abbreviation of the concatenating operation, the result of which is denoted as $[re(\mathbf{h}), re(\mathbf{r})]$. Conv means the convolution operation. FCP represents the fully connected projection. PROJ is the abbreviation of projection. MM is the matrix multiplication operation with all candidates entities embedding matrices. Finally, the sigmoid function is utilized to calculate the triplets.

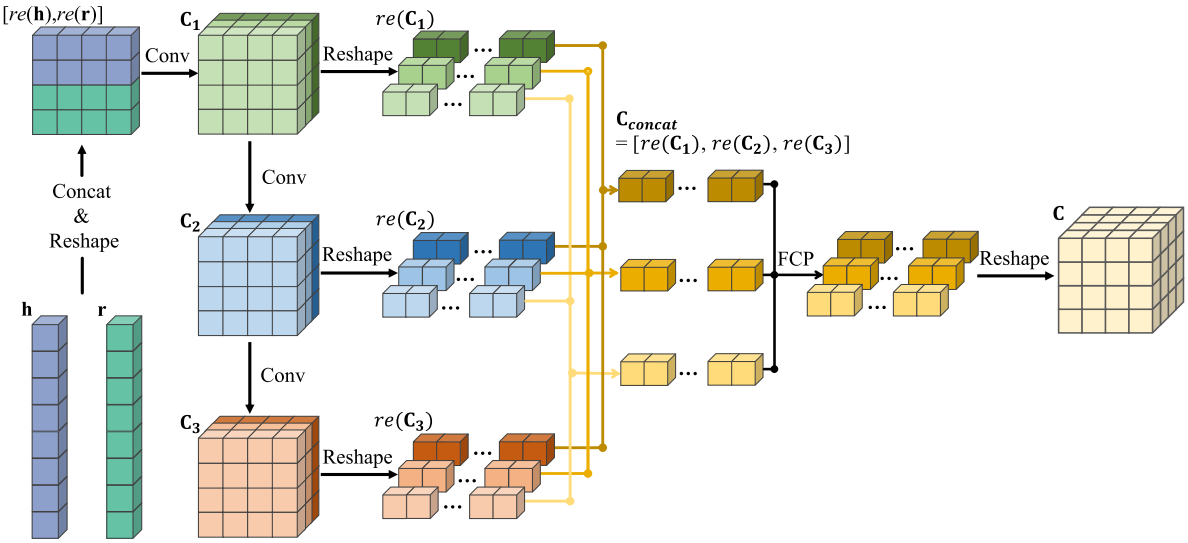

Fig. 3. The detail of High-low Level Features Interaction. **h** denotes the head entity embeddings and **r** is the relation embeddings. $re(\cdot)$ means reshape operation. $[\cdot]$ means concatenating operation, the result of which is denoted as $[re(h), re(r)]$. Conv means convolution. FCP represents the fully connected projection. $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ represents the result of the first, second and third convolution respectively.

3.2.1. *High-low level features interaction*

As previously mentioned, deeper convolutions extract in-depth features, which have less detail but more comprehensive semantic information. In contrast, shallow features can express local information concretely but may lack coverage of the receptive field. To acquire a broader range of semantic information at multilevel and boost the quality of the interaction, we introduce High-low Level Features Interaction. The detailed process is shown in Fig. 3,

Our overall goal is to obtain entities embedding matrix and relations embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d}, \mathbf{R} \in \mathbb{R}^{|\mathcal{R}| \times d}$ for other downstream tasks. To achieve this goal, we first initialize the head entity vector **h** and the relation vector **r**. They are reshaped to 2D and concatenated to get the input of convolution, which is denoted as $[re(\mathbf{h}), re(\mathbf{r})]$. The specific process is as follows:

$$\begin{aligned} &\mathbf{h} \in \mathbb{R}^d \rightarrow re(\mathbf{h}) \in \mathbb{R}^{d_w \times d_h}, \mathbf{r} \in \mathbb{R} \rightarrow re(\mathbf{r}) \in \mathbb{R}^{d_w \times d_h} \\ &[re(\mathbf{h}), re(\mathbf{r})] = concat\big(re(\mathbf{h}), re(\mathbf{r})\big) \end{aligned}, \tag{2}$$

where **h** refers to the head entity vector, and $re(\mathbf{h})$ denotes the head entity vector after 2D transformation, **r** refers to the relation vectors, and $re(\mathbf{r})$ denotes relation vector after 2D transformation. The *concat* is a function to concatenate two vectors into a single one with **h** and **r** as the input.

Secondly, we convolve the concatenated result $[re(\mathbf{h}), re(\mathbf{r})] \in \mathbb{R}^{(2d_w) \times d_h}$ three times, and the results of each convolution correspond to the low, medium, and high-level features, respectively, which is denoted as follows:

$$\begin{aligned} \mathbf{C_1} &= [re(\mathbf{h}), re(\mathbf{r})] * \mathbf{K_1} \\ \mathbf{C_2} &= \mathbf{C_1} * \mathbf{K_2} \\ \mathbf{C_3} &= \mathbf{C_2} * \mathbf{K_3} \end{aligned}, \tag{3}$$

where $*$ refers to the convolution. $\mathbf{K}_1 \in \mathbb{R}^{c_1 \times k_{w_1} \times k_{h_1}}$ is the convolutional kernel of the first convolution, the size of which is $k_{w_1} \times k_{h_1}$. $\mathbf{K}_2 \in \mathbb{R}^{c_1 \times c_2 \times k_{w_2} \times k_{h_2}}$ is the convolution kernel of the second convolution, the size of which is $k_{w_2} \times k_{h_2}$. $\mathbf{K}_3 \in \mathbb{R}^{c_3 \times c_2 \times k_{w_3} \times k_{h_3}}$ is the convolution kernel of the third convolution, the size of which is $k_{w_3} \times k_{h_3}$. We denote the result of each convolution as $\mathbf{C_1} \in \mathbb{R}^{c_1 \times m_{w_1} \times m_{h_1}}, \mathbf{C_2} \in \mathbb{R}^{c_2 \times m_{w_2} \times m_{h_2}}, \mathbf{C_3} \in \mathbb{R}^{c_3 \times m_{w_3} \times m_{h_3}}$, where $m_{w_1} \times m_{h_1}, m_{w_2} \times m_{h_2}, m_{w_3} \times m_{h_3}$ is the size of each output feature map. The number of convolution kernels for all convolution operations is set to c: $c_1 = c_2 = c_3 = c$, and the output size of each feature map is set to be fixed: $m_{w_1} = m_{w_2} = m_{w_3} = m_w$, $m_{h_1} = m_{h_2} = m_{h_3} = m_h$.

Thirdly, after the reshape process of three convolutions results, the feature maps of each channel are concatenated together and then fused using a fully connected mapping layer, which is denoted as:

$$\mathbf{C}_{concat} = \left[re(\mathbf{C_1}), re(\mathbf{C_2}), re(\mathbf{C_3})\right] = concat\left(re(\mathbf{C_1}), re(\mathbf{C_2}), re(\mathbf{C_3})\right), \tag{4}$$

where $re\left(\mathbf{C}_i\right) \in \mathbb{R}^{C \times (m_{w_i} \times m_{h_i})}$ represents the reshape operation. $concat(\cdot)$ represents the concatenating process.

Finally, the fully connected layer is utilized for fusion operation. The feature map is adjusted to the appropriate size as input to the next module. The process is defined as follows:

$$\mathbf{C} = \mathbf{C}_{concat}\mathbf{W} + \mathbf{b}, \tag{5}$$

where $\mathbf{C}_{concat} \in \mathbb{R}^{c_i \times (3 \times m_w \times m_h)}$ is the concatenated result. $\mathbf{W} \in \mathbb{R}^{(3 \times m_\omega \times m_h) \times (m_\omega \times m_h)}, \mathbf{b} \in \mathbb{R}^{c \times (m_\omega \times m_h)}$ are parameters in model. The feature map $\mathbf{C} \in \mathbb{R}^{c \times (m_w \times m_h)}$ is generated by reshape process, which transfers it from 1D to 2D.

3.2.2. Criss-cross attention mechanism

To equip each feature with the information of other neighbors in the fused feature map, we introduce Criss-cross Attention, which can expand the receptive field of each feature for more efficient interaction performance. The specific process is as follows:

We firstly input the feature map to perform a 1×1 convolution to generate feature matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{c' \times m_w \times m_k}$ and $\mathbf{V} \in \mathbb{R}^{c \times m_w \times m_k}$. Note that c' represents the account of channels, which is less than c to reduce dimension.

Next, affinity operation is utilized to create an attention map $\mathbf{A} \in \mathbb{R}^{(m_w+m_h-1) \times (m_w \times m_h)}$. A vector $\mathbf{Q}_u \in \mathbb{R}^{c'}$ can be obtained for every position u in the feature of $\mathbf{Q}$. Furthermore, by extracting feature vectors from $\mathbf{K}$ that are in the same row or column as position u, the set $\mathbf{\Omega}_u \in \mathbb{R}^{(m_w+m_h-1) \times c'}$ is acquired. $\mathbf{\Omega}_{i,u} \in \mathbb{R}^{c'}$ represents the ith element of $\mathbf{\Omega}_u$. Then the degree of correlation between features $\mathbf{Q}_u$ and $\mathbf{\Omega}_{i,u}, i = \left[1, \ldots, m_w + m_h - 1\right]$ is calculated as follows:

$$d_{i,u} = \mathbf{Q}_u \mathbf{\Omega}_{i,u}^T, \tag{6}$$

where $d_{i,u} \in \mathbf{D} \in \mathbb{R}^{(m_w+m_h-1) \times (m_w \times m_h)}$. Then, to acquire the attention map $\mathbf{A}$, a softmax layer is applied on $\mathbf{D}$ in the dimension of the channel.

To reassign the weights of features, $\mathbf{V} \in \mathbb{R}^{C \times W \times H}$ is generated. For each position u in the feature of $\mathbf{V}$, we may construct a vector $\mathbf{V}u \in \mathbb{R}^c$. $\mathbf{\Phi}_u \in \mathbb{R}^{(m_w+m_h-1) \times c}$ is a group of feature vectors in the same row or column as position u in $\mathbf{V}$. Finally, an aggregation operation is conducted to collect contextual information, which is defined as follows:

$$\mathbf{C}'_u = \sum_{i=1}^{m_w+m_h-1} \mathbf{A}_{i,u} \mathbf{\Phi}_{i,u} + \mathbf{C}_u, \tag{7}$$

where $\mathbf{C}' \in \mathbb{R}^{c \times m_w \times m_h}$. $\mathbf{C}'_u \in \mathbb{R}^c$ is the vector of position u in $\mathbf{C}'$. $\mathbf{C}_u \in \mathbb{R}^c$ is the vector of the input $\mathbf{C}$ at position u.

3.3. Optimization and inference

We introduce the following scoring function to assess the triplet (h, r, t). Concisely, the feature map $\mathbf{C}'$ is matched with the vectorized tail entity after dimensionality reduction. The dimensionality reduction operation means that the feature map is fully concatenated and projected to the same dimension as the tail vector. All of the above are defined as the scoring function below:

$$\psi(h, r, t) = \sigma\left(f\left(vec\left(\mathbf{C}'\right)\mathbf{W} + \mathbf{b}\right)\mathbf{t}\right), \tag{8}$$

where $\psi(h, r, t)$ is the score function of a given triplet (h, r, t). $\sigma(\cdot)$ is the sigmoid activation function. $f(\cdot)$ represents ReLU activation function. $vec(\cdot)$ represents the vectorization operation, which flattens the vector. $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the fully connected layer. For a correct triplet, $\psi(h, r, t)$ is anticipated to be close to 1, otherwise 0. After processing the head entities and relations, the scores with N tail entities are calculated. All triplets of this process are denoted as $(h, r, *)$, where $*$ represents the tail entities. The predicted value is the $(h, r, *)$ score by scoring function (8).

Algorithm 1: Learning ConvHLE

Data: knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$;
Input: embedding dimension d;
batch size b;
learning rate a;
early stop parameter es;
negative sample n.
Output: embedding matrix $\mathbf{E}, \mathbf{R}$

data reading and initialization;
random batch processing of data with a batch size of b;
initialize the embedding matrix of $\mathcal{E}$ and $\mathcal{R}$ as $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times d}$ and $\mathbf{R} \in \mathbb{R}^{|\mathcal{R}|\times d}$;
loop
  $\mathcal{T}_{batch} \leftarrow \text{sample}(\mathcal{T}, b)$; // extracting a set of randomly sampled triplets $\mathcal{T}_{batch}$
  for $(h,r,t) \in \mathcal{T}_{batch}$ **do**
    $(h,r,t) \rightarrow (h,r,t')$; // Generate the incorrect triplet.
    $\mathcal{T}'_{batch} \leftarrow (h,r,t') \wedge (h,r,t') \notin \mathcal{G}$; // Generate the incorrect triplets set
    $\psi(h,r,t) = \sigma\left(f\left(vec\left(\mathbf{C}'\right)\mathbf{W} + \mathbf{b}\right)\mathbf{t}\right)$; // Return the correct triplet score
    $\psi(h,r,t') = \sigma\left(f\left(vec\left(\mathbf{C}'\right)\mathbf{W} + \mathbf{b}\right)\mathbf{t}'\right)$; // Return the incorrect triplet score.
  update embedding matrix $\mathbf{E}, \mathbf{R}$ by Equation (10).
end loop;

In the experiment, the value of N can be the total number of all entities in the knowledge map, namely, $N = |\varepsilon|$, which is called the 1-N scoring mode. The value of N can also be $X+1$; that is, during model training, X wrong entities are selected for each (h,r) through negative sampling operation, which is called 1-X mode. Specifically, the negative sampling operation randomly selects X entities in the entity set as the tail entities and generates triplets that do not exist in the knowledge maps. The process is as follows:

$$(h,r,t) \rightarrow (h,r,t') \ \ or \ \ (h,r,t) \rightarrow (h',r,t)\,. \tag{9}$$

In this paper, Cross Entropy is used to train parameters, and the label smoothing approach is utilized for optimization. We define the loss function as follows:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y log\left(\psi(h,r,t)\right) + (1-y)\, log\left(1-\psi(h,r,t)\right)\right], \tag{10}$$

in which

$$y = \begin{cases} 1 & , \ for(h,r,t) \in \mathcal{T} \\ 0 & , \ for(h,r,t) \in \mathcal{T}'. \end{cases} \tag{11}$$

The training process of ConvHLE is given in Algorithm 1. Our source data is the given knowledge graph $\mathcal{G}$. The input includes embedding dimension, batch size, learning rate, early stop parameter, and negative sample. The output consists of the entity embedding matrix $\mathbf{E}$ and the relation embedding matrix $\mathbf{R}$. The source data is first read and batched. Considering that ConvHLE utilizes three-layer convolution and uses Sigmoid and ReLU as activation functions, we adopt Xavier initialization (Glorot & Bengio, 2010) instead of the more common Gaussian distribution initialization to avoid gradient disappearance or explosion. Then, we extract a set of randomly sampled triplets, which is denoted as $\mathcal{T}_{batch}$ and an incorrect triplet is generated to form the incorrect triplet set. Then, the scoring function (8) is used to score the correct and incorrect triplet. The entity matrix $\mathbf{E}$ and the relation embedding matrix $\mathbf{R}$ are updated according to the loss function (10). This process repeats itself until all data in the knowledge graph has been processed. The final embeddings of entity matrix $\mathbf{E}$ and relation matrix $\mathbf{R}$ are the model's output, which can be applied to other downstream tasks.

Note that to mitigate overfitting, random dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) is inserted into necessary processes. After the convolution operation and the projection to the tail entity space, the batch normalization process (Ioffe & Szegedy, 2015) is conducted to improve the convergence speed. Label smoothing (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) is adopted to enhance the generalization capabilities of the model while alleviating overfitting.

4. Experiments and analysis

4.1. Experimental settings

4.1.1. Datasets

To comprehensively and objectively evaluate the performance of ConvHLE, we selected six publicly available datasets and divided them following (Bordes et al., 2013; Dettmers et al., 2018; Lin, Han, Xie, Liu, & Sun, 2018; Trouillon et al., 2016), which can ensure

Table 1
Statistical details of the datasets for knowledge graph.

Dataset	#Entities	#Relations	#Triplets		
			Train	Valid	Test
FB15K	14,951	1345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5000
FB15K-237	14,541	237	272,115	17,535	20,446
WN18RR	40,943	11	86,835	3,034	3134
Kinship	104	25	8,544	1068	1074
DB100K	99,604	470	595,572	50000	50,000

that the follow-up researchers can conduct objective experimental comparisons. The datasets used are: FB15K,[1] WN18,[2] FB15K-237,[3] WN18RR,[4] Kinship,[5] and DB100K.[6] Detailed statistics are shown in Table 1.

- FB15K is extracted from Freebase,[7] which is an extensive knowledge base that includes information about films, celebrities, sports teams, prizes, etc.
- Extracted from WordNet8,[8] WN18's entities mirror the word senses, while its relations represent relationships between words, such as the upper and lower words.
- FB15K-237 eliminated reversible relations of FB15K.
- WN18RR removed reversible relations of WN18.
- Kinship is a small dataset describing kinship relationships.
- DB100K is simplified from the numerous data in the DBpedia[9] by eliminating relations not included in the DBpedia ontology and discarding entities that appear less than 20 times.

4.1.2. Evaluation metrics

For a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, $\mathcal{T}_{test} = \{x_1, x_2, \ldots, x_{|\mathcal{T}_{test}|}\}$ denotes the test set. Given the head entity and relation, we utilize the learned embedding to calculate the score of each tail entity (candidate entities) in the range of 0 to 1. Moreover, for 1-N training, the candidate entities are all in the test set. If it is 1-X training, the candidate entities are the total set of negative sampled entities and the correct tail entities. We utilize MRR (Mean Reciprocal Rank) and Hits@k (Hit Ratio values when $k = 1$, 3, and 10) as metrics, which are defined as follows:

$$MRR: \quad \frac{1}{|\mathcal{T}_{test}|} \sum_{i=1}^{|\mathcal{T}_{test}|} \left(\frac{1}{rank_i}\right), \tag{12}$$

$$Hits@k: \frac{1}{|\mathcal{T}_{test}|} \sum_{i=1}^{|\mathcal{T}_{test}|} f\left(rank_i\right), \; f(x) = \begin{cases} 1, & x \le k \\ 0, & x > k. \end{cases} \tag{13}$$

Note that in our experiments, both the test set and its inverse will be merged, denoted as $(h, r, t) \to (t, r', h)$. Negative sampling is performed by swapping the head and tail entities, which is equivalent to generating incorrect triplets. $rank_i$ is the rank of the correct tail entities among the candidate tail entities. Additionally, if there is the same expression in the following formula, it means the same and will not be repeated.

MRR means the average reverse rank for all test triplets, while Hits@k means the probability of ranking less than or equal to k. Higher MRR or Hits@k indicates a more satisfying model performance. When the latter value is the same, the lower the value of k, the more accurate the model is.

4.1.3. Comparison baselines

To fully illustrate the performance of our proposed ConvHLE in link prediction tasks, we contrast it with 16 state-of-the-art baselines from various research areas: (1) Translational distance models: TransE (Bordes et al., 2013), TorusE (Ebisu & Ichise, 2018), RotatE (Sun et al., 2019), (2) Semantic matching models: DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), TuckER (Balazevic, Allen, & Hospedales, 2019b), (3) Graph neural network models: R-GCN (Schlichtkrull et al., 2018), SACN (Shang, Tang, Huang, Bi, He, & Zhou, 2019), CompGCN (Vashishth, Sanyal, et al., 2020), (4) Convolutional neural network models:

[1] https://everest.hds.utc.fr/lib/exe/fetch.php?media=en:fb15k.tgz
[2] https://everest.hds.utc.fr/lib/exe/fetch.php?media=en:wordnet-mlj12.tar.gz
[3] https://www.microsoft.com/en-us/download/details.aspx?id=52312
[4] https://paperswithcode.com/dataset/wn18rr
[5] https://archive.ics.uci.edu/ml/datasets/kinship
[6] https://github.com/neukg/AcrE/tree/master/data/DB100K
[7] http://www.freebase.be/
[8] https://wordnet.princeton.edu/
[9] https://wiki.dbpedia.org/

Table 2
Hyperparameters settings on different datasets.

Dataset	Learning rate	Batch size	Dropout			Kernel size	Embedding size	Kernel number	Label smoothing
			Input	Feature map	Hidden layer				
FB15K	0.001	256	0.2	0.1	0.2	3×3	225	96	0.1
WN18	0.001	256	0.2	0.2	0.3	3×3			
FB15K-237	0.001	128	0.5	0.2	0.5	3×3			
Kinship	0.001	128	0.5	0.4	0.5	3×3			
DB100K	0.001	128	0.2	0.2	0.2	3×3			
WN18RR	0.0005	128	0.5	0.1	0.3	7×7			

ConvE (Dettmers et al., 2018), ConvKB (Nguyen, Nguyen, Nguyen, & Phung, 2018), HypER (Balazevic et al., 2019a), ConvR (Jiang et al., 2019), InteractE (Vashishth et al., 2020), ArcE (Ren et al., 2020) (Serial). In addition to the baselines mentioned above, on some specific datasets, we utilize relevant models with publicly available results as baselines to demonstrate the efficiency of ConvHLE more comprehensively: (1) Link prediction on Kinship: HAKE (Zhang, Cai, Zhang, & Wang, 2020), (2) Link prediction on DB100K: RUGE (Guo, Wang, Wang, Wang, & Guo, 2018), SEEK (Xu et al., 2020), MQuadE (Yu, Cai, Sun, & Li, 2021), (3) Complex relations modeling: Jointly (Xu, Qiu, Chen, & Huang, 2017).

Considering that some methods are optimized for particular datasets, we need to evaluate the generalization ability of our model on multiple datasets. Furthermore, some baseline models are only experimented on a few datasets, resulting in incomplete experimental results of the baseline models.

4.1.4. Experimental settings

Inspired by previous research (Balazevic et al., 2019a; Dettmers et al., 2018; Ren et al., 2020; Smith, 2015; Vashishth et al., 2020; Yin & Shen, 2018) and considering our experimental data, we choose the following hyperparameters to evaluate our model fairly and effectively: Entity and relation embedding dimensions are chosen from set [100, 150, 200, 250, 300]. The dropout of input, hidden layer, and feature map are selected from [0.1, 0.2, 0.3, 0.4, 0.5]. Other hyperparameters' ranges are as follows: batch size[128, 256], learning rate [0.005, 0.001, 0.0005, 0.0001], kernel size [3×3, 5×5, 7×7, 9×9, 11×11], kernel number [32, 64, 96].

The following combination of parameter settings has outstanding performance on all data sets: entity and relation embedding size 225, label smoothing 0.1, kernel number 96. For FB15K-237, input dropout is set to 0.5, feature map dropout to 0.2, and hidden dropout to 0.5, respectively. As to WN18RR, we set the learning rate to 0.0005, input dropout, hidden dropout, and feature map dropout are set to 0.5, 0.3, and 0.1. The kernel size is set to 7×7. As to FB15K, the batch size is set to 256, and the input dropout, hidden dropout, and feature map dropout are set to 0.2, 0.2, and 0.1. We set the batch size for WN18 at 256, input dropout, hidden dropout, and feature map dropout set to 0.2, 0.3, and 0.2. For Kinship, input dropout, hidden dropout, and feature map dropout are set to 0.5, 0.5, and 0.4. For DB100K, input dropout, hidden dropout, and feature map dropout are all set to 0.2. The above settings are enumerated in Table 2.

4.2. Results and discussions

4.2.1. Main results of link prediction

We use state-of-the-art baselines for comparison, including translational distance, semantic matching, and neural network-based models, to assess ConvHLE's performance in link prediction tasks. We evaluate our model on the FB15K-2373 and WN18RR datasets, which are commonly used in link prediction tasks. We can conclude from the results in Table 3 that our proposed model can achieve competitive effectiveness on both datasets compared with baseline models. ConvHLE exhibits a 13.2%, 5.6%, and 2.3% relative improvements in MRR and a 16.0%, 6.3%, and 3.1% relative improvements in Hits@1 when compared to the convolutional neural network-based models ConvE, HypER, and AcrE(serial). Compared with RotatE, ConvHLE has improved the performance on both datasets: MRR increased by 6.5%, @1, @3, @10 improved by 11.2%, 5.3%, and 2.1%, respectively.

The results of the experiments demonstrate ConvHLE's advantage in link prediction tasks. Incidentally, it is commonly acknowledged that models based on convolutional neural networks are generally considered to have difficulty in fitting WN18RR. Experimental results demonstrate that our proposed approach outperforms ConvE, basically the same as HypER and ArcE (serial), with little difference between the optimal results.

FB15K and WN18 were used to further verify the validity of our model. Experiments were conducted on these two datasets, shown in Table 4. Compared with ConvE, Hyper, and AcrE (Serial), which are based on convolutional neural networks, MRR improved by 23.3%, 2.5%, and 2.4%, @1 increased by 35.7%, 3.1%, and 4.1%, @3 increased by 17.4%, 2.4% and 1.3%, and @10 increased by 8.1%, 1.5% and 0.2%. Compared with other baselines, MRR improved by 3.5%, @1 increased by 3.0%, @3 increased by 3.2%, and @10 increased by 1.8%. According to the experimental results on the WN18, ConvHLE outperforms ConvE, which is the same as other state-of-the-art baselines.

Several representative state-of-the-art models are used as baselines to further assess the efficiency of ConvHLE in inferring the missing head entities and tail entities on FB15K-237. The experimental results are shown in Table 5. It proves that ConvHLE outperforms the baselines across all evaluation metrics.

Kinship is a small dataset describing various family relationships, which can test whether the model can learn quickly and fit with limited information. The experimental results are shown in Table 6. The differences between the optimal models are not very

Table 3
Results of MRR and Hits@k on FB15K-237 and WN18RR in link prediction task.

Models	FB15K-237				WN18RR			
	MRR	Hits			MRR	Hits		
		@1	@3	@10		@1	@3	@10
TransE (Bordes et al., 2013)	0.288	0.192	0.325	0.478	0.192	0.021	0.367	0.435
TorusE (Ebisu & Ichise, 2018)	0.316	0.217	0.335	0.484	0.452	0.422	0.464	0.512
RotatE (Sun et al., 2019)	0.318	0.216	0.358	0.525	0.474	0.425	0.490	**0.571**
DistMult (Yang et al., 2015)	0.180	0.100	0.191	0.347	0.322	0.243	0.371	0.463
ComplEx (Trouillon et al., 2016)	0.233	0.145	0.266	0.408	0.380	0.337	0.423	0.472
TuckER (Balazevic et al., 2019b)	0.354	0.262	0.388	0.535	0.463	0.435	0.478	0.516
R-GCN (Schlichtkrull et al., 2018)	0.249	0.151	0.264	0.417	0.226	0.157	0.269	0.376
SACN (Shang et al., 2019)	0.350	0.260	0.390	0.540	0.470	0.430	0.480	0.540
CompGCN (Vashishth, Sanyal, et al., 2020)	0.355	0.264	0.390	0.535	**0.479**	**0.443**	**0.494**	0.546
ConvE (Dettmers et al., 2018)	0.318	0.231	0.351	0.493	0.430	0.400	0.440	0.520
ConvKB (Nguyen et al., 2018)	0.243	0.155	0.371	0.421	0.249	0.057	0.417	0.524
HypER (Balazevic et al., 2019a)	0.341	0.252	0.376	0.520	0.465	0.436	0.477	0.522
ConvR (Jiang et al., 2019)	0.350	0.261	0.385	0.528	0.475	**0.443**	0.489	0.537
InteractE (Vashishth et al., 2020)	0.355	0.263	0.390	0.539	0.465	0.433	0.481	0.526
ArcE (Ren et al., 2020) (Serial)	0.352	0.260	0.388	0.537	0.463	0.429	0.478	0.534
ConvHLE	**0.360**	**0.268**	**0.395**	**0.544**	0.469	0.437	0.482	0.532

Table 4
Results of MRR and Hits@k on FB15K and WN18 in link prediction task.

Models	FB15K				WN18			
	MRR	Hits			MRR	Hits		
		@1	@3	@10		@1	@3	@10
TransE (Bordes et al., 2013)	0.380	0.231	0.472	0.641	0.454	0.089	0.823	0.934
DistMult (Yang et al., 2015)	0.654	0.546	0.733	0.824	0.822	0.728	0.914	0.936
ComplEx (Trouillon et al., 2016)	0.692	0.599	0.759	0.840	0.941	0.936	0.936	0.947
R-GCN (Schlichtkrull et al., 2018)	0.696	0.601	0.760	0.842	0.814	0.686	0.928	0.955
ConvE (Dettmers et al., 2018)	0.657	0.558	0.723	0.831	0.943	0.935	0.946	0.956
TorusE (Ebisu & Ichise, 2018)	0.733	0.674	0.771	0.832	0.947	0.943	0.950	0.954
RotatE (Sun et al., 2019)	0.782	0.735	0.823	0.882	0.948	0.943	0.952	0.958
ArcE (Ren et al., 2020)(Serial)	0.791	0.727	0.838	0.896	**0.950**	**0.946**	**0.953**	**0.959**
ConvHLE	**0.810**	**0.757**	**0.849**	**0.898**	**0.950**	**0.946**	0.952	0.957

significant. All the indicators of ConvHLE are the most optimal, which proves that ConvHLE can also model small datasets. Namely, it can effectively distinguish entities and relations when the information is limited.

DB100K is a public dataset widely used recently, with many entities and rich triplets. It is more challenging to conduct link prediction tasks on DB100K. Whether the large-scale KGs can be modeled is an important indicator to measure the performance of the future research direction of KGE. Experimental results are displayed in Table 7. By comparison with other state-of-the-art baselines, the previous three metrics of ConvHLE are optimal, and Hits@10 is similar to the optimal result. Specifically, compared with the convolutional neural network-based AcrE (Serial), MRR, Hits@1, and Hits@3 improved by 7.0%, 14.6%, and 2.4%, respectively. When compared with the best results of other baselines, the MRR increased by 6.3%, Hits@1 increased by 10.7%, Hits@3 increased by 2.8%, and Hits@10 increased by 3.7%. This demonstrates that ConvHLE can model larger and more complex knowledge graphs.

Table 5
Results of predicting head and tail entities on FB15K-237.

Models	Predicting head				Predicting tail			
	MRR	Hits			MRR	Hits		
		@1	@3	@10		@1	@3	@10
ConvE (Dettmers et al., 2018)	0.211	0.132	0.231	0.368	0.416	0.323	0.457	0.601
SACN (Shang et al., 2019)	0.241	0.158	0.260	0.409	0.446	0.352	0.490	0.631
ArcE (Ren et al., 2020) (Serial)	0.254	0.166	0.279	0.434	0.451	0.353	0.497	0.642
RotatE (Sun et al., 2019)	0.239	0.149	0.265	0.424	0.432	0.329	0.477	0.639
InteractE (Vashishth et al., 2020)	0.258	0.170	0.283	0.437	0.454	0.358	0.498	0.644
ConvHLE	**0.261**	**0.171**	**0.289**	**0.442**	**0.460**	**0.366**	**0.501**	**0.647**

Table 6
Results of MRR and Hits@k on Kinship in link prediction task.

Model	Kinship			
	MRR	Hits@1	Hits@3	Hits@10
ComplEx (Trouillon et al., 2016)	0.823	0.733	0.899	0.971
ConvE (Dettmers et al., 2018)	0.833	0.738	0.917	0.981
ConvKB (Nguyen et al., 2018)	0.614	0.436	0.755	0.953
RotatE (Sun et al., 2019)	0.843	0.760	0.919	0.978
HAKE (Zhang et al., 2020)	0.852	0.769	0.928	0.980
InteractE (Vashishth et al., 2020)	0.777	0.664	0.870	0.959
CompGCN (Vashishth, Sanyal, et al., 2020)	0.778	0.667	0.868	0.967
ArcE (Ren et al., 2020)(Serial)	0.864	0.787	0.931	**0.987**
ConvHLE	**0.870**	**0.797**	**0.941**	**0.987**

Table 7
Results of MRR and Hits@k on DB100K in link prediction task.

Model	DB100K			
	MRR	Hits@1	Hits@3	Hits@10
DistMult (Sun et al., 2019)	0.233	0.115	0.301	0.448
ComplEx (Trouillon et al., 2016)	0.242	0.126	0.312	0.440
RUGE (Guo et al., 2018)	0.246	0.129	0.325	0.433
ComplEx-NEE+AER (Ding, Wang, Wang, & Guo, 2018)	0.306	0.244	0.334	0.418
SEEK (Xu et al., 2020)	0.338	0.280	0.370	0.467
MQuadE (Yu et al., 2021)	0.402	0.318	0.451	0.546
AcrE(Serial) (Ren et al., 2020)	0.399	0.304	0.453	**0.570**
ConvHLE	**0.429**	**0.356**	**0.464**	0.567

4.2.2. Experiments on complex relations modeling

With various relations contained, FB15K is particularly useful in assessing the performance of complex relations modeling. In KGs, relations can be defined as 1-to −1, 1-to-N, N-to −1, and N-to-N (Bordes et al., 2013). According to the magnitude of the average number of tail entities per head entity (tph) and the average number of head entities per tail entity (hpt), the following divisions are made:

$$\begin{cases} hpt < \eta \ and \ tph < \eta \Rightarrow 1 - to - 1 \\ hpt < \eta \ and \ tph \geq \eta \Rightarrow 1 - to - N \\ hpt \geq \eta \ and \ tph < \eta \Rightarrow N - to - 1 \\ hpt \geq \eta \ and \ tph \geq \eta \Rightarrow N - to - N \end{cases}, \tag{14}$$

Table 8
Comparison results of Hits@10 on FB15K in different relations.

Models	Predicting Head(Hits@10)				Predicting Tail(Hits@10)			
	1-to-1	1-toN	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
TransE (Bordes et al., 2013)	0.437	0.657	0.182	0.472	0.437	0.197	0.667	0.500
DistMult (Sun et al., 2019)	0.889	0.954	0.539	0.795	0.884	0.624	0.949	0.879
ComplEx (Trouillon et al., 2016)	0.885	0.947	0.404	0.683	0.888	0.511	0.939	0.750
Jointly (Xu et al., 2017)	0.838	0.951	0.211	0.479	0.830	0.308	0.947	0.531
ConvE (Dettmers et al., 2018)	0.866	0.959	0.598	0.786	0.852	0.704	0.957	0.858
TorusE (Ebisu & Ichise, 2018)	0.775	0.906	0.479	0.859	0.773	0.592	0.893	0.890
RotatE (Sun et al., 2019)	**0.929**	0.967	0.602	0.893	0.923	0.713	0.961	0.922
HypER (Balazevic et al., 2019a)	0.924	0.968	0.634	0.891	**0.924**	0.731	0.962	0.920
ConvHLE	0.921	**0.973**	**0.698**	**0.894**	0.921	**0.844**	**0.963**	**0.923**

Table 9
Ablation experiments on link prediction task evaluation.

Model	FB15K-237				FB15K			
	MRR	Hits			MRR	Hits		
		@1	@3	@10		@1	@3	@10
ConvHLE (w/o CA)	0.357	0.265	0.392	0.541	**0.810**	0.756	**0.849**	**0.898**
ConvHLE (CA2)	0.356	0.263	0.392	0.542	0.807	0.753	0.846	0.895
ConvHLE (Add)	0.328	0.239	0.360	0.508	0.774	0.703	0.826	0.889
ConvHLE (Conv3)	0.331	0.242	0.363	0.501	0.778	0.708	0.830	0.892
ConvHLE	**0.360**	**0.268**	**0.395**	**0.544**	**0.810**	**0.757**	**0.849**	**0.898**

note that for the reasonableness of the experiment, η is generally taken as 1.5.

Experimental Results are presented in Table 8, proving that ConvHLE has achieved satisfactory results for most metrics. What stands out in the table is that compared with convolutional neural network-based ConvE and HypER, the head entities prediction results of ConvHLE for N-to −1 relations have been improved by 14.3% and 5.1%, respectively. As for other state-of-the-art models, the head entities prediction results of ConvHLE have been improved by 6.2%. The head entities prediction results of the 1-to-N relations have been improved by 16.6% and 13.3%, respectively. Compared with other paradigms, the result has improved by 15.5%. The experiment indicates that ConvHLE is capable of handling complex relations modeling, demonstrating the effectiveness of our proposed approach.

4.2.3. Ablation experiments

To examine the effectiveness of each separate module of the ConvHLE, ablation experiments were conducted on FB15K and FB15K-237. Experimental results are presented in Table 9. w/o CA indicates that the criss-cross attention mechanism will not be performed; CA2 means that after the convolution output, the criss-cross attention mechanism is conducted twice to enable each feature to obtain more information; Add represents different levels of features fused by an addition operation without the fully connected layers. Conv3 means computing by a three-layer convolutional network and output directly, without any fusion or attention mechanism.

By comparing the results, we conclude that each metric of the ConvHLE model is optimal on both datasets. ConvHLE outperforms ConvHLE (w/o CA), which proves the effectiveness of the criss-cross attention mechanism. The comparison between ConvHLE and ConvHLE (CA2) shows that it can easily cause insufficient fitting by fusing too much information at each output from the feature map. The comparison between ConvHLE and ConvHLE (Add) proves the effectiveness of fully connected projection. By comparing with ConvHLE (Conv3), we conclude that if the information at all levels can be effectively fused, it can equip the model with more expressiveness.

4.2.4. Experiments on hyperparameters optimization

Batch size and learning rate settings: In the experiments of hyperparameter search, we only modify the batch size and learning rate parameters. The results are given in Table 10. The bold parts are the optimal results. From the experimental results, we can learn that when the learning rate is set to 0.0005, and the batch size is set to 128, ConvHLE can achieve optimal results.

Table 10
Results of batch size and learning rate settings on WN18RR dataset.

LR	Batch size	WN18RR			
		MRR	Hits@1	Hits@3	Hits@10
0.001	256	0.463	0.430	0.476	0.526
0.0001	256	0.446	0.419	0.456	0.498
0.0005	256	0.460	0.427	0.472	0.527
0.005	256	0.451	0.417	0.463	0.520
0.001	128	0.462	0.431	0.472	0.522
0.0001	128	0.447	0.421	0.456	0.501
0.0005	**128**	**0.469**	**0.437**	**0.482**	**0.532**
0.005	128	0.447	0.414	0.461	0.511

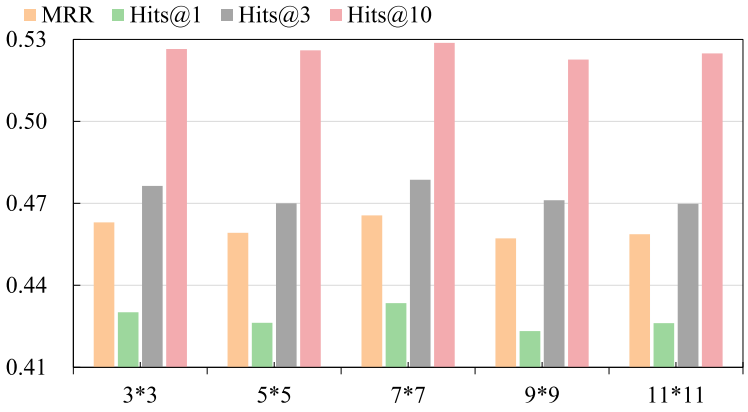

Fig. 4. Effect of kernel size on experimental results on WN18RR dataset.

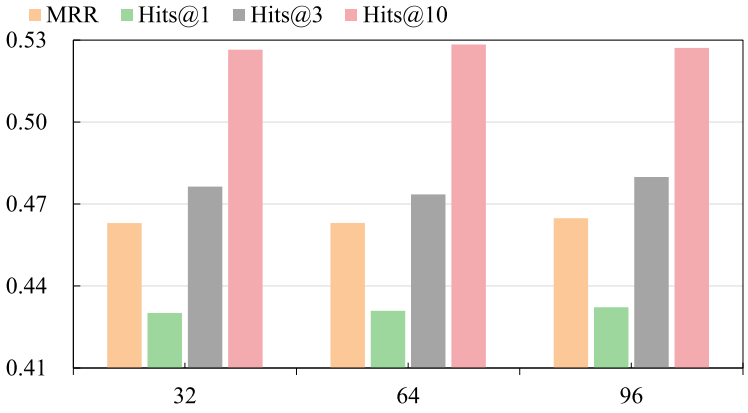

Fig. 5. Effect of the number of convolution kernel on experimental results on WN18RR dataset.

Kernel size settings: The hyperparameter of convolution kernel size also needs to be adjusted to fit with different models and data sets. Different convolution kernel sizes indicate the difference in the number of head entities and relations interactions in one convolution operation. For this purpose, five commonly used convolution kernel sizes were used for experiments, namely $3 \times 3, 5 \times 5, 7 \times 7, 9 \times 9$ and 11×11. The results obtained from the experiment are shown in Fig. 4. Through comparison, it can be found that all indicators are optimal using 7×7 convolution kernels. While using 3×3 and 11×11 convolution kernels, the performances are basically in the second ladder. 5×5 and 9×9 convolution kernels have the least satisfactory results. Smaller convolution kernel sizes limit the number of interactions, while larger ones are prone to overfitting.

Number of convolution kernels settings: The number of convolution kernels affects the performance of the model ability. With more convolution kernels, more feature combinations can be extracted. However, too many convolution kernels may also lead to overfitting. The commonly used number of convolution kernels is 32, 64, and 96. The results obtained from the experiment are shown in Fig. 5. Other indicators are at their best when the convolution kernel number is set to 96, with the exception of Hits@10. So the best setting of the convolution kernel number is 96.

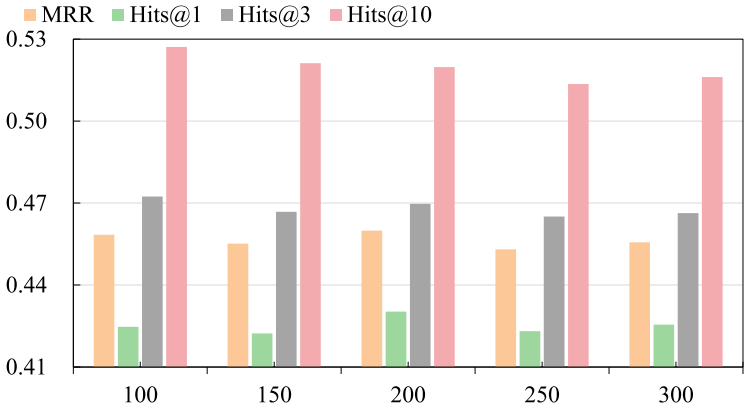

Fig. 6. Effect of different embedding dimensions on WN18RR dataset.

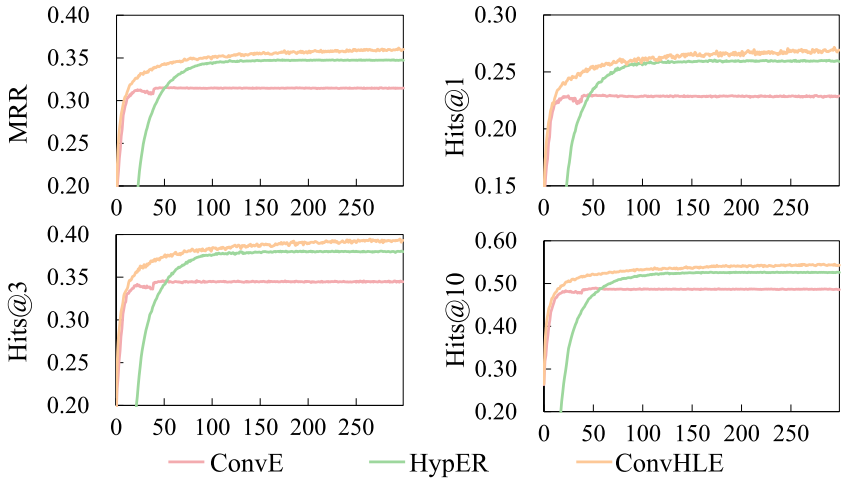

Fig. 7. Convergence results of ConvE, HypER, and ConvHLE on FB15K-237 dataset.

Table 11
Experiments on ConvHLE efficiency evaluation. Space complexity and training time in seconds on three deep models.

Model	Space complexity	Dataset				
		FB15K	WN18	FB15K-237	WN18RR	DB100K
ConvE (Dettmers et al., 2018)	$(\lvert\mathcal{E}\rvert + \lvert\mathcal{R}\rvert)d$	33.10	34.89	17.68	22.86	240.43
HypER (Balazevic et al., 2019a)	$(\lvert\mathcal{E}\rvert + \lvert\mathcal{R}\rvert)d$	35.89	33.87	18.71	21.86	232.00
ConvHLE	$(\lvert\mathcal{E}\rvert + \lvert\mathcal{R}\rvert)d$	38.68	34.57	20.83	21.89	241.63

Embedding size settings: Embedding size usually affects the model's performance because too small embedding will lead to poor model fitting, and too large embedding size will lead to overfitting of the model. Embedding size are chosen from 100, 150, 200, 250, and 300. The results are shown in Fig. 6. In conclusion, the embedding size of 200 obtains optimal MRR and Hits@1. Generally speaking, we pay more attention to MRR and Hits@1, so we conclude that the best embedding size is 200.

4.2.5. Model efficiency evaluation

We have analyzed the training process of CNN-based ConvE, HypER, and ConvHLE, shown in Fig. 7. It can be found that ConvHLE has achieved better results than others. It is still slowly fitting after 100 epochs. In comparison, ConvE and HypER have finished fitting after 100 epochs, which confirms that ConvHLE achieves optimal results and fits better than other CNN-based models.

The experiments of inference speed are conducted on FB15K, WN18, FB15K-237, WN18RR, and DB100K, on a PC server equipped with Tesla V100. The comparison models are convolutional neural network-based baselines ConvE and HypER. Table 11 provides

the experimental results. The space complexities are almost $(|\mathcal{E}| + |\mathcal{R}|)d$, and the inference time spent on each dataset is very close, which proves that ConvHLE has significant performance improvement under acceptable time complexity.

5. Further discussions and implications

5.1. Results analysis

Compared to state-of-the-art baselines, ConvHLE achieves more satisfactory performance across most metrics. The comparison with other CNN-based models indicates that ConvHLE possesses more accurate results and a rapid convergence speed. Experiments on complex relations modeling highlight its viability and robustness. The ablation study shows that the advances in expressiveness are mostly obtained from the high-low level features interaction and the criss-cross attention mechanism. In conclusion, ConvHLE can significantly and efficiently improve knowledge graph embedding results, maintaining satisfying time and space consumption with an effective combination of hyperparameters. The primary advantages of our work are brought to light, which can be mostly manifested in the following areas: (1) With the high-low level features fusion and interaction, ConvHLE can effectively handle feature hierarchy existing in KGs and show more gratifying performance. (2) The criss-cross attention mechanism can effectively expand the receptive field for more efficient interaction performance. (3) Each separate module in ConvHLE cooperates with high efficiency, which intuitively explains why our proposed model obtains state-of-the-art performance and outperforms other baselines.

5.2. Theoretical and practical implications

The significance of our work is theoretical and practical. From the theoretical perspective, ConvHLE not only develops new thinking for handling feature hierarchy in KGs but also provides a methodology for incorporating attention mechanisms into the KGE methods. Compared with translational distance and semantic matching models, it naturally extracts more abundant semantic information by convolutional-based feature extraction. Compared with convolutional neural network-based models, its rational consideration of feature hierarchy significantly improves the effectiveness of knowledge graph embeddings. Furthermore, with the cross-cross attention mechanism, it maximizes the receptive field of each feature and acquires more efficient interaction performance, providing a new idea for applying the attention mechanism in the KGE models. We present novel insight into feature fusion and interaction in the KGE, with state-of-the-art performance on most metrics.

Practically, ConvHLE can be employed to model large-scale KGs. By incorporating multilevel features with the criss-cross attention mechanism, it can successfully capture richer semantic information and improve the expressiveness of embeddings. The knowledge representations that our proposed model has learned can also be applied to various downstream tasks, such as recommendation systems, question and answering, and dialogue systems, etc. We can use the entity and relation vector learned by ConvHLE to the recommendation system (Shimizu, Matsutani, & Goto, 2022) through one-by-one learning, joint learning, alternate learning, and other methods. Based on the stronger modeling ability of ConvHLE, we can conduct more complex reasoning, which is beneficial to the development of question-answering (Bakhshi, Nematbakhsh, Mohsenzadeh, & Rahmani, 2022). Language and knowledge are closely related, and the knowledge graph modeled by ConvHLE can also have critical applications in dialogue systems (Tuan et al., 2022).

6. Conclusion and future work

The constraint of single-level feature interactions affects the majority of existing CNN-based KGE models. Therefore, we propose ConvHLE to incorporate multilevel feature interaction and harvest semantic information for richer expressiveness. Significantly, the criss-cross attention mechanism contributes to boosting the additional interactions. We conduct comprehensive experiments to evaluate the performance of ConvHLE on six benchmark datasets. Results illustrate our model's advantages compared to other baselines. Ablation experiments show that each component of our model is necessary. Moreover, the effects of different parameters on the effectiveness of the model are investigated comprehensively.

Certainly, we recognize that ConvHLE has limitations. The fundamental one is that although utilizing the criss-cross attention mechanism to reduce the computation, the time and space consumption is still unavoidably high due to the three-layer convolution design. Thus, ConvHLE is not significantly remarkable in training time. Besides, our focus is more on structure-based knowledge graph embedding. The semantic information that the names of entities and relations carry with them is also worth investigating. For future work, we intend to enhance the robustness of ConvHLE through multiple corpus fusion. Efforts will be made to make breakthroughs in the space–time consumption of ConvHLE through various approaches. Introducing the idea of text-based knowledge graph embedding into our proposed model is also a major direction of our future work.

CRediT authorship contribution statement

Jingxiong Wang: Resources, Data curation, Writing – review & editing. **Fobo Shi:** Conceptualization, Methodology, Writing, Supervision. **Duantengchuan Li:** Resources, Data curation, Writing – review & editing, Conceptualization, Project administration, Supervision. **Yuefeng Cai:** Supervision, Writing – review & editing, Data curation. **Bing Li:** Supervision, Writing – review & editing. **Xiaoguang Wang:** Visualization, Validation. **Zhen Zhang:** Supervision, Writing – review & editing. **Chao Zheng:** Supervision, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by Major Project of the National Social Science Fund (No. 21&ZD334), the National Natural Science Foundation of China (Nos. 62032016), and the Key Research and Development Program of Hubei Province (No. 2021BAA031), and the Scientific Research Fund for Young Teachers of Wuhan Sports University (No. 2022S10).

References

Arenas-Márquez, F. J., del Rocío Martínez Torres, M., & Toral, S. (2021). Convolutional neural encoding of online reviews for the identification of travel group type topics on TripAdvisor. *Information Processing and Management*, *58*(5), Article 102645.
Bakhshi, M., Nematbakhsh, M., Mohsenzadeh, M., & Rahmani, A. M. (2022). SParseQA: Sequential word reordering and parsing for answering complex natural language questions over knowledge graphs. *Knowledge-Based Systems*, *235*, Article 107626.
Balazevic, I., Allen, C., & Hospedales, T. M. (2019a). Hypernetwork knowledge graph embeddings. In *Lecture notes in computer science*: *vol. 11731*, *ICANN (workshop)* (pp. 553–565).
Balazevic, I., Allen, C., & Hospedales, T. M. (2019b). TuckER: Tensor factorization for knowledge graph completion. In *EMNLP/IJCNLP (1)* (pp. 5184–5193).
Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *NIPS* (pp. 2787–2795).
Chen, L., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. CoRR, abs/1706.05587.
Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D knowledge graph embeddings. In *AAAI* (pp. 1811–1818).
Ding, B., Wang, Q., Wang, B., & Guo, L. (2018). Improving knowledge graph embedding using simple constraints. In *ACL (1)* (pp. 110–121).
Ebisu, T., & Ichise, R. (2018). TorusE: Knowledge graph embedding on a lie group. In *AAAI* (pp. 1819–1826).
Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *JMLR proceedings*: *vol. 9*, *AISTATS* (pp. 249–256).
Guo, S., Wang, Q., Wang, L., Wang, B., & Guo, L. (2018). Knowledge graph embedding with iterative guidance from soft rules. In *AAAI* (pp. 4816–4823).
Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *JMLR workshop and conference proceedings*: *vol. 37*, *ICML* (pp. 448–456).
Isufi, E., Pocchiari, M., & Hanjalic, A. (2021). Accuracy-diversity trade-off in recommender systems via graph convolutions. *Information Processing and Management*, *58*(2), Article 102459.
Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)* (pp. 687–696).
Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(2), 494–514.
Jiang, X., Wang, Q., & Wang, B. (2019). Adaptive convolution for multi-relational learning. In *NAACL-HLT (1)* (pp. 978–987).
Kazemi, S. M., & Poole, D. (2018). SimplE embedding for link prediction in knowledge graphs. In *NeurIPS* (pp. 4289–4300).
Li, H., Wang, J., Du, X., Hu, Z., & Yang, S. (2023). KBHN: A knowledge-aware bi-hypergraph network based on visual-knowledge features fusion for teaching image annotation. *Information Processing and Management*, *60*(1), Article 103106.
Lin, T., Dollár, P., Girshick, R. B., He, K., Hariharan, B., & Belongie, S. J. (2017). Feature pyramid networks for object detection. In *CVPR* (pp. 936–944).
Lin, Y., Han, X., Xie, R., Liu, Z., & Sun, M. (2018). Knowledge representation learning: A quantitative review. CoRR, abs/1812.10901.
Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *AAAI* (pp. 2181–2187).
Nguyen, D. Q., Nguyen, T. D., Nguyen, D. Q., & Phung, D. Q. (2018). A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL-HLT (2)* (pp. 327–333).
Nickel, M., Tresp, V., & Kriegel, H. (2011). A three-way model for collective learning on multi-relational data. In *ICML* (pp. 809–816).
Ren, F., Li, J., Zhang, H., Liu, S., Li, B., Ming, R., et al. (2020). Knowledge graph embedding with atrous convolution and residual learning. In *COLING* (pp. 1532–1543).
Schlichtkrull, M. S., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *Lecture notes in computer science*: *vol. 10843*, *ESWC* (pp. 593–607).
Shang, C., Tang, Y., Huang, J., Bi, J., He, X., & Zhou, B. (2019). End-to-end structure-aware convolutional networks for knowledge base completion. In *AAAI* (pp. 3060–3067).
Shimizu, R., Matsutani, M., & Goto, M. (2022). An explainable recommendation framework based on an improved knowledge graph attention network with massive volumes of side information. *Knowledge-Based Systems*, *239*, Article 107970.
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.
Smith, L. N. (2015). No more pesky learning rate guessing games. CoRR, abs/1506.01186.
Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.
Sun, Z., Deng, Z., Nie, J., & Tang, J. (2019). RotatE: Knowledge graph embedding by relational rotation in complex space. In *ICLR (poster)*.
Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *CVPR* (pp. 2818–2826).
Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. In *JMLR workshop and conference proceedings*: *vol. 48*, *ICML* (pp. 2071–2080).
Tuan, Y., Beygi, S., Fazel-Zarandi, M., Gao, Q., Cervone, A., & Wang, W. Y. (2022). Towards large-scale interpretable knowledge graph reasoning for dialogue systems. In *ACL (findings)* (pp. 383–395).
Vashishth, S., Sanyal, S., Nitin, V., & Talukdar, P. P. (2020). Composition-based multi-relational graph convolutional networks. In *ICLR*.
Vashishth, S., et al. (2020). InteractE: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *AAAI* (pp. 3009–3016).
Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *AAAI* (pp. 1112–1119).
Xiao, H., Chen, Y., & Shi, X. (2021). Knowledge graph embedding based on multi-view clustering framework. *IEEE Transactions on Knowledge and Data Engineering*, *33*(2), 585–596.
Xie, Z., Zhu, R., Liu, J., Zhou, G., & Huang, J. X. (2022). An efficiency relation-specific graph transformation network for knowledge graph representation learning. *Information Processing and Management*, *59*(6), Article 103076.
Xu, J., Qiu, X., Chen, K., & Huang, X. (2017). Knowledge graph representation with jointly structural and textual encoding. In *IJCAI* (pp. 1318–1324).
Xu, W., Zheng, S., He, L., Shao, B., Yin, J., & Liu, T. (2020). SEEK: Segmented embedding of knowledge graphs. In *ACL* (pp. 3888–3897).
Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (poster)*.
Yin, Z., & Shen, Y. (2018). On the dimensionality of word embedding. CoRR, abs/1812.04224.
Yu, J., Cai, Y., Sun, M., & Li, P. (2021). MQuadE: A unified model for knowledge fact embedding. In *WWW* (pp. 3442–3452).
Zhang, Z., Cai, J., Zhang, Y., & Wang, J. (2020). Learning hierarchy-aware knowledge graph embeddings for link prediction. In *AAAI* (pp. 3065–3072).
Zhang, J., Huang, J., Gao, J., Han, R., & Zhou, C. (2022). Knowledge graph embedding by logical-default attention graph convolution neural network for link prediction. *Information Sciences*, *593*, 201–215.