HyConvE: A Novel Embedding Model for Knowledge Hypergraph Link Prediction with Convolutional Neural Networks

Chenxu Wang
College of Intelligence and Computing
Tianjin University
Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application
Tianjin, China
cxwang1998@tju.edu.cn

Xin Wang*
College of Intelligence and Computing
Tianjin University
Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application
Tianjin, China
wangx@tju.edu.cn

Zhao Li
College of Intelligence and Computing
Tianjin University
Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application
Tianjin, China
lizh@tju.edu.cn

Zirui Chen
College of Intelligence and Computing
Tianjin University
Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application
Tianjin, China
zrchen@tju.edu.cn

Jianxin Li
School of Information Technology
Deakin University
Geelong, Victoria, Australia
jianxin.li@deakin.edu.cn

ABSTRACT

Knowledge hypergraph embedding, which projects entities and n-ary relations into a low-dimensional continuous vector space to predict missing links, remains a challenging area to be explored despite the ubiquity of n-ary relational facts in the real world. Currently, knowledge hypergraph link prediction methods are essentially simple extensions of those used in knowledge graphs, where n-ary relational facts are decomposed into different subelements. Convolutional neural networks have been shown to have remarkable information extraction capabilities in previous work on knowledge graph link prediction. In this paper, we propose a novel embedding-based knowledge hypergraph link prediction model named HyConvE, which exploits the powerful learning ability of convolutional neural networks for effective link prediction. Specifically, we employ 3D convolution to capture the deep interactions of entities and relations to efficiently extract explicit and implicit knowledge in each n-ary relational fact without compromising its translation property. In addition, appropriate relation and position-aware filters are utilized sequentially to perform two-dimensional convolution operations to capture the intrinsic patterns and position information in each n-ary relation, respectively. Extensive experimental results on real datasets of knowledge hypergraphs and knowledge graphs demonstrate the superior performance of HyConvE compared with state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

Knowledge Hypergraph, Knowledge Graph, Link Prediction

ACM Reference Format:
Chenxu Wang, Xin Wang, Zhao Li, Zirui Chen, and Jianxin Li. 2023. HyConvE: A Novel Embedding Model for Knowledge Hypergraph Link Prediction with Convolutional Neural Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583256

1 INTRODUCTION

The recent development of knowledge graphs has been partly realizing the vision of the Semantic Web [3]. Knowledge graphs store facts of the form $r(h, t)$, where r is the binary relation, and h and t are the head and tail entities, respectively. As the cornerstone of artificial intelligence, recent years have witnessed the rapid adoption of knowledge graphs in fields such as question and answer systems [16], relation extraction [26], and recommender systems [6]. Nevertheless, in addition to binary relational facts, n-ary relational facts involving more than two entities are also prevalent in reality, e.g., in the Freebase[4], more than one-third of entities participate in non-binary relations [38], and around 61% of relations are non-binary [10]. These findings demonstrate that knowledge hypergraphs,

*Xin Wang is the corresponding author.

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583256

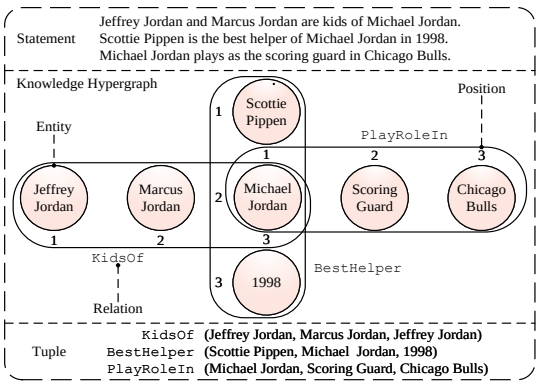


Figure 1: A real-world example of knowledge hypergraph about a set of facts related to *Michael Jordan*, where each tuple is accompanied by different positional information.

which use n-ary relation to describe relationships among several entities, are ubiquitous in the real world. Figure 1 shows a real-world example of a knowledge hypergraph about *Michael Jordan*. The original facts can be represented by n-ary tuples, where each entity in the n-ary relation appears in different positions. For example, *Michael Jordan* is in the 1st, 2nd, and 3rd positions in the ternary relation `PlayRoleIn` (*Michael Jordan*, *Scoring Guard*, *Chicago Bulls*), `BestHelperOf` (*Scottie Pippen*, *Michael Jordan*, *1998*), and `KidsOf` (*Jeffrey Jordan*, *Marcus Jordan*, *Michael Jordan*), respectively.

Like knowledge graphs, due to the exponential growth of multi-source information, it becomes challenging, even impossible, for the large-scale n-ary knowledge base to be updated in an appropriate way, resulting in incomplete and outdated knowledge hypergraphs. To address this issue, several approaches dedicated to knowledge hypergraph link prediction have recently been emerging, out of which the most representative approach is knowledge hypergraph embedding. Early models [38, 41] use a star-to-clique method to convert n-ary relations to several binary ones, which has proven to be less effective due to the information loss [10]. Some models [13, 15, 22] represent n-ary facts in terms of role-value pairs, i.e., $\{r_1 : e_1, r_2 : e_2, ..., r_n : e_n\}$, taking the correlation of role-value pairs as the optimization goal. For example in Figure 1, *Jeffrey Jordan* and *Marcus Jordan* are children of *Michael Jordan*, which are represented in the form of {DAUGHTER: *Jeffrey Jordan*, SON: *Marcus Jordan*, FATHER: *Michael Jordan*}. Following the triple mode in knowledge graphs, there are also some approaches [14, 28] that model the n-ary facts as $\{h, r, t, r_1 : e_1, ..., r_{n-2} : e_{n-2}\}$, where the n-ary facts are decomposed into a major triple and a series of role-entity pairs to account the importance of head and tail entities, while the compatibility between each role–entity pair and the primary triple is independently calculated before a final aggregation. For instance, the ternary relation `BestHelperOf` in Figure 1 can be represented as {*Scottie Pippen*, `BestHelperOf`, *Michael Jordan*, TIME: *1998*}. In other frameworks [8, 10, 21, 41], n-ary facts is modeled as an n-ary relation with entities in different positions: $\{r, e_1, e_2, ..., e_n\}$. In Figure 1, the ternary relation `PlayRoleIn` can be modeled as {`PlayRoleIn`, *Michael Jordan*, *Scoring Guard*, *Chicago Bulls*}. Since there is no decomposition to break the original tuple, the positional information and intrinsic patterns can be better preserved. Similarly, the above methods learn the embeddings of entities and relations in low-dimensional space and choose different scoring functions for fact plausibility measurement. However, the latent and implicit knowledge in the n-ary fact is unavailable for these approaches due to their shallow modeling restrictions [7].

With the remarkable achievements of convolutional neural networks (CNNs) in computer vision and other fields, previous studies have also introduced CNNs to knowledge graph embedding. ConvE[7] is the first work to use 2D convolution for knowledge graph link prediction. However, due to the simple stacking of entities and relations, solely using external 2D convolution filters cannot capture the interactions between entities and relations well. Subsequently, several representative works such as HypER [1], ConvR [18], AcrE [27], and InteractE [34] emerge to further characterize the interaction between entities and relations and get better results. Nonetheless, these works often use complex convolutional layers, resulting in a dense model structure. These models focus too much on the extraction of implicit information, and their ability to capture surface knowledge is weak due to the loss of translation property.

In this paper, we propose a novel convolution-based knowledge hypergraph link prediction model called HyConvE, in which convolutions with distinct characteristics are jointly applied to the knowledge hypergraph for different feature extraction. On the one hand, to take advantage of the respective advantages of 1D and 2D convolution, we transform the 1D convolution in ConvKB [24] and the 2D convolution in ConvE [7] into 3D convolution for better feature extraction. Due to the preservation of the translation property, surface knowledge will not be compromised when extracting deeper interactions between entities and relations. On the other hand, in the process of 2D convolution, when the entity embedding passes through the relation-specific and position-specific filters, the inherent patterns in the relations and the position (role) information of the entities will be fully captured, respectively. Our model is evaluated on nine standard benchmark datasets of both binary and n-ary. Extensive experimental results show the superior performance compared with a series of state-of-the-art knowledge embedding baseline methods. Our main contributions are summarized as follows:

- **Latent and surface knowledge extraction.** HyConvE can effectively capture feature interactions between relations and related entities using the mechanism of 3D convolution, which learns deeper features in each n-ary relational fact without compromising translation property.
- **Relation and position-aware information capture.** HyConvE takes advantage of different 2D convolutional layers to extract the inherent semantic patterns and position information in each n-ary relation, making the features of each tuple more informative for better performance.
- **Better performance.** Extensive experiments on both knowledge hypergraph and knowledge graph datasets demonstrate that HyConvE outperforms representative baselines over standard benchmarks.

2 RELATED WORK

2.1 Knowledge Graph Embedding

Translational models. Translational methods such as TransE [5], TransH [37], and TransR [20] use distance metrics to measure the

fact plausibility by projecting the entities into low-dimensional latent space. Later works such as TransC [23] and TransRHS [40] are relation hierarchical structure (RHS) based methods aiming at encoding the relation or entity concept as spheres for knowledge graph embedding.

Tensor factorization based models. Tensor factorization based approaches define the scoring function as a bilinear product of entity/relation embeddings. Concretely, the score function in RESCAL [25] is defined as the multiplication of the head-tail entity vector and relation matrix. DistMult [39] and ComplEx [33] use symmetric relation matrices and complex-valued embedding spaces for optimization. RotatE [31] treats the relation as a rotation operation when projecting head and tail entities into complex space, while Tucker [2] and AutoSF [42] introduces tucker decomposition and automated relation matrix into knowledge graph embedding, respectively. These models have shallow and linear structures, resulting in incomplete extraction of implicit knowledge and inefficient performance.

Neural network based models. Another line of knowledge graph link prediction relies closely on the popularity of neural networks, especially convolutional neural networks. ConvE [11], as the first convolution-based model, reshapes the head entity vector and relation vector and concatenates them into a matrix as the input of the convolutional layer, and the triplet score is represented as the inner product of the network output vector and the tail entity vector. ConvKB [24] stacks the head and tail entities and relation vectors as input to the 1D convolutional layer, which retains the translation property in KG. ConvR [18] employs relation-specific filters to deal with the inadequacy of ConvE for interaction capture. HypER [1] uses relation-specific convolution filters generated by a fully connected network to convolve the head entity embedding sufficiently but may come at the expense of extra amount of parameters. InteractE [34] further increases the interaction between relation and entity embeddings by checkered feature reshaping and depthwise circular convolution. AcrE [27] replaces the regular convolution filters with atrous filters to provide better performance and solve the vanishing gradient problem through residual learning. In addition to convolutional neural networks, R-GCN[29], and CompGCN[35] are representative achievements of the combination of knowledge graph and graph neural networks.

Since real-world knowledge often exists in the form of hypergraphs, the above models only designed for knowledge graphs have limitations in modeling, the elaborate binary scoring functions make them tricky to be adapted to n-ary relations. In contrast, HyConvE is not limited by the number of relations, and can perform effective link prediction in mixed-arity knowledge hypergraphs.

2.2 Knowledge Hypergraph Embedding

Translational models. The earliest translation-based knowledge hypergraph embedding methods are extensions of translational approaches from binary relations to n-ary. m-TransH [38], as the earliest model, is the extension of TransH [37] where the plausibility score of each fact is a weighted sum of projected entities. RAE [41] further employs a fully connected network (FCN) to model the relatedness of all involved entities. However, these two models cannot achieve ideal performance due to the weak expressiveness of the translational framework.

Tensor factorization based models. GETD [21] extends Tucker [2] as the tensor-factorization-based approach in knowledge hypergraphs. However, GETD can only be trained and evaluated with fixed arity relations, hindering its real-world application. S2S [8] further extends GETD from fixed to mixed-arity data in response to this problem. HypE [10] is inspired by SimplE [19], which considers position information in each fact with positional filters, while RAM [22] attempts to model the semantics of each role in n-ary relation. However, the above models always have shallow structures, which may lead to insufficient latent and implicit knowledge extraction.

Neural network based models. NaLP [15] models the relatedness of values based on the roles they play in different tuples, while tNalp+[13] further considers type information and optimizes the negative sampling. HINGE [28] and NeuInfer [14] are the following works that decompose the n-ary relational fact into a primary triplet and several role-entity pairs. StarE [12] uses CompGCN for modeling triples of n-ary fact decompositions, focusing only on modeling parts of n-ary facts. These works also use neural networks as a medium for information extraction. Nevertheless, due to the overemphasis on the value of the primary triple, the semantic features and structural integrity of the original n-ary facts are inevitably destroyed through decomposition.

To the best of our knowledge, our proposed HyConvE is the first model that treats the n-ary fact as a tuple and makes full use of different convolutional neural network properties to perform effective link prediction for knowledge hypergraphs.

3 PROBLEM FORMULATION

3.1 Knowledge Hypergraph.

Given a finite set of entities $\mathcal{E}$, relations $\mathcal{R}$, and observed tuples $\mathcal{T}_O$, a *knowledge hypergraph* can be represented as $\mathcal{H} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_O)$. Each observed fact in $\mathcal{T}_O$ is in the form of a tuple $t = r(e_1, e_2, ..., e_k)$, where k is the non-negative arity of relation r representing the number of entities involved within each relation. A knowledge graph is a special case of a knowledge hypergraph where the arity of all relations is two.

3.2 Knowledge Hypergraph Link Prediction.

Let $\mathcal{T}_T$ denote the set of ground truth tuples, where $\mathcal{T}_O \subseteq \mathcal{T}_T$. The knowledge hypergraph link prediction task aims at predicting missing component in n-ary facts, where the missing component can be either an entity in the tuple $r(e_1, e_2, ..., ?, ..., e_k)$ or an n-ary relation $?(e_1, e_2, ..., e_k)$. Knowledge hypergraph embedding is one of the most effective methods for knowledge hypergraph link prediction where an n-ary tuple $r(e_1, e_2, ..., e_k)$ is projected into a low-dimensional latent space. The scoring function is optimized iteratively to achieve higher probability scores for those real facts.

In subsequent sections, we use lowercase letters for scalars, bold lowercase letters for vectors, and bold uppercase letters for matrices. For the index, we denote a[i] as the i-th element of vector $\mathbf{a}$, $\mathbf{A}[i, j]$ as the $[i, j]$-th element of matrix $\mathbf{A}$.

4 METHODOLOGY

The overall architecture of HyConvE is shown in Figure 2. The final score of each n-ary fact consists of two feed-forward paths. On the one hand, the 1D convolution in ConvKB and the 2D convolution

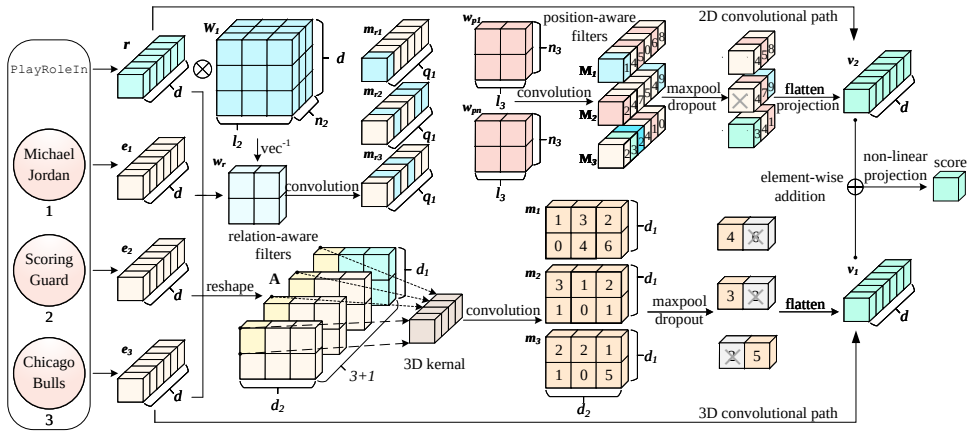


Figure 2: The framework of the HyConvE model.

in ConvE are extended to 3D convolution to capture the deeper interactions of each relational fact without compromising the translation property, in which way the explicit and implicit knowledge can be learned jointly. On the other hand, the 2D relation-aware and position-aware convolution are employed successively to capture the position information and inherent pattern of entities in each relation. Finally, we sum up the output features of two paths through the element-wise addition and then convert the vector as a scalar using a fully connected projection layer (linear operation) as the final score of the input tuple. Specifically, given an k-ary fact $r(e_1, e_2, ..., e_k)$, we first embed it as a d-dimension row vector $\boldsymbol{r} \in \mathbb{R}^d$ and $\boldsymbol{e}_i \in \mathbb{R}^d, i = 1, 2, ..., k$.

4.1 Latent and Surface Knowledge Extraction

Considering that ConvE only stacks entries of the same dimension of the reshaped entity and relation matrices, entities and relations interact only at the connections and lose their translation property. We reshape the d-dimension vectors $\boldsymbol{r}$ and $\boldsymbol{e_i}$ into matrices (*images*) $\overline{\boldsymbol{r}} \in \mathbb{R}^{d_1 \times d_2}$ and $\overline{\boldsymbol{e_i}} \in \mathbb{R}^{d_1 \times d_2}, i = 1, 2, ..., k$, where $d_1 \times d_2 = d$. Then the k images is concatenated into a cube (*video*) $A = [\overline{\boldsymbol{r}} \,||\, \overline{\boldsymbol{e_1}}, ..., ||\, \overline{\boldsymbol{e_k}}] \in \mathbb{R}^{(k+1) \times d_1 \times d_2}$, where $\overline{\boldsymbol{r}}$ and $\overline{\boldsymbol{e_i}}, i = 1, 2, ..., k$ denotes the reshaped vector. Note that in ConvKB, the embedding vectors are simply stacked into a matrix of 3 rows and d columns for 1D convolution, while we perform a reshape operation to make the embedding representation more stereoscopic. Thanks to the extension from image to *video*, the cube can be equated to a *video* with $k + 1$ frames of *images*, and 3D convolution can be naturally applied for feature extraction. In the mixed-arity knowledge hypergraph, to ensure that the dimension of the feature maps output by different n-ary facts through the convolution layer is consistent, for a k-ary facts input, we employ n_1 filters $\boldsymbol{w} \in \mathbb{R}^{n_1 \times (k+1) \times i \times j}$ that are repeatedly operated over every row of the cube A through 3D convolution layer and finally generate a series of feature maps $m_1, m_2, ..., m_k \in \mathbb{R}^{d_1 \times d_2}$ which characterize the feature within the neural space. Each feature map can be formulated as follows:

$$m_i = \boldsymbol{w_i} * \mathbf{A} + b = \boldsymbol{w} * [\overline{\boldsymbol{r}} \,||\, \overline{\boldsymbol{e_1}}, ..., ||\, \overline{\boldsymbol{e_k}}] \tag{1}$$

where $b \in \mathbb{R}$ is a bias term and m_l is the l-th feature map generated by the l-th filter, and $*$ denotes the convolutional operation.

After passing the convolutional layer, we compress the feature and simplify the network complexity through max-pooling followed by the application of a dropout layer to flatten these feature maps into a vector:

$$\boldsymbol{v_1} = \mathbf{maxpool}(\boldsymbol{w} * [\overline{\boldsymbol{r}} \,||\, \overline{\boldsymbol{e_1}}, ..., ||\, \overline{\boldsymbol{e_k}}]) \tag{2}$$

where $\boldsymbol{v_1}$ is the output features of 3D convolutional path and **maxpool** denotes the max-pooling operation. It is worth noting that after reshaping and stacking operations before 3D convolution, both explicit and implicit features will be effectively captured and the translational property will be maintained simultaneously.

4.2 Intrinsic Semantic Information Capture

As aforementioned, according to our modeling approach for the n-ary facts, the intrinsic semantic information of an entity in each n-ary relational fact is also closely related to the relation in the n-ary tuple and its position (role) in the n-ary relation. Therefore, we set up relation-aware and position-aware filters for feature extraction, respectively. Specifically, for a k-ary tuple, we first generate the matrix of relation-specific convolutional filters by passing the relation embedding vector through a linear transformation matrix $\mathbf{W_1}$, then result is reshaped to generate a matrix of convolutional filters, the above process can be formulated as follows:

$$\boldsymbol{w_r} = \mathbf{vec}^{-1}(\boldsymbol{r} \cdot \mathbf{W_1}) \tag{3}$$

where $\mathbf{W_1} \in \mathbb{R}^{d \times l_2 n_2}$ denotes the linear transformation matrix, l_2 the filter length, n_2 the number of filters per relation. **vec** is a vectorization of a matrix and $\mathbf{vec}^{-1}$ its inverse.

Given s_2 the stride of relation-specific convolution. Then, the entity embeddings involved in the n-ary relation will be convolved with that relation-specific filter to obtain a series of relation-aware entity feature maps $m_{ri} \in \mathbb{R}^{q_2 \times n_2}$, each of which can be formulated as follows:

$$m_{ri} = \boldsymbol{e_i} * \boldsymbol{w_r} = \boldsymbol{e_i} * \mathbf{vec}^{-1}(\boldsymbol{r} \cdot \mathbf{W_1}) \tag{4}$$

where $q_2 = (d - l_2)/s + 1$ is the feature map size.

Let l_3 denote the length of position-aware convolutional filters, s_3 the stride of position-aware convolution, and $w_{pi} \in \mathbb{R}^{n_3 \times l_3}$ the convolutional filters associated with each position in a tuple, where

n_3 is the number of filters per position. For a given fact, all n_3 feature maps corresponding to an entity are concatenated into a vector $\boldsymbol{v_p}$ of size $n_3 \times q_3$:

$$\mathbf{M_i} = \left[m_{ri} * \boldsymbol{w_{pi1}} \,||\, m_{ri} * \boldsymbol{w_{pi2}} \,||\, ... \,||\, m_{ri} * \boldsymbol{w_{pin_3}}\right] \tag{5}$$

where $q_3 = (q_2 - l_3)/s_3 + 1$ is the size of position-aware feature map. Thus, each entity embedding e_i appearing at position i in a given tuple is sequentially convolved with a relation-specific filter and a position-specific filter to obtain a feature map of size q_2. Finally, the hidden vector is obtained after a max pooling process followed by concatenating and flattening and projecting it through the transformation matrix $\mathbf{W_2}$ to obtain the output of the 2D convolutional path:

$$\boldsymbol{v_2} = \mathbf{maxpool}(\mathbf{vec}([\mathbf{M_i} \,||\, \mathbf{M_2} \,||\, ... \,||\, \mathbf{M_k}]))\mathbf{W_2} \tag{6}$$

Eventually, we obtain the output characteristics of the tuple by element-wise addition: $\boldsymbol{v} = \boldsymbol{v1} + \boldsymbol{v2}$. After introducing the activation function and randomly discarding some neurons to prevent overfitting, a fully connected projection layer $\mathbf{W_3}$ is added to confirm the vector into a scalar as the final score of the input n-ary tuple. The final tuple score given by the model can be formalized as:

$$\boldsymbol{score} = \mathrm{g}(\boldsymbol{v_1} + \boldsymbol{v_2})\mathbf{W_3} \tag{7}$$

We use rectified linear units, e.g., ReLU, as the nonlinear activation function g and apply batch normalization after each layer to accelerate training and stabilize convergence. Some operations, such as dropout [30] and batch normalization [17], are not described in detail.

4.3 Joint Training

Using the scoring function obtained above, we designed the training loss as well as the learning objective. In each learning iteration, A batch of positive tuples is first selected from the knowledge hypergraph. Since we only have positive instances available, we also need to train our model on the negative instance. Thus, we develop a negative sampling strategy for knowledge hypergraph link prediction. following the contrastive approach used in [5], for each positive tuple, we produce a set of negative samples of size $N|r|$ by replacing each of the entities with N random entities in the tuple:

$$\bigcup_{i=1}^{k} \mathcal{N}_x^{(i)} \equiv \bigcup_{i=1}^{k} \{e_1, \cdots, \bar{e}_i, \cdots, e_k \notin \mathcal{F} | \bar{e}_i \in \mathcal{E}, \bar{e}_i \neq e_i\} \tag{8}$$

where $\mathcal{N}_x^{(i)}$ replaces the entity in the i-th position. Our model was trained using Stochastic Gradient Descent with mini-batches and AdaGrad [9] for tuning the learning rate, by minimizing the negative log-likelihood of the logistic model with $\mathcal{L}_2$ regularization:

$$\mathcal{L} = \sum_{r(e_1,...,e_k) \in \{\mathcal{H} \cup \mathcal{H}'\}} \log(1 + \exp(l_{r(e_1,...,e_k)} \cdot f(r(e_1, ..., e_k)))) + \phi \tag{9}$$

where $\mathcal{H}$ and $\mathcal{H}'$ represent the positive and negative sampling tuple sets, respectively, and $l_{r(e_1,...,e_k)}$ represents the label of the tuple.

$$l_{(r,e_1,...,e_k)} = \begin{cases} 1 & \text{for } r(e_1, ..., e_k) \in \mathcal{H} \\ -1 & \text{for } r(e_1, ..., e_k) \in \mathcal{H}' \end{cases} \tag{10}$$

The regularization term ϕ in Equation 9 consists of two parts: squared norms of parameters of the several convolution layers, max-pool layers, and fully connected layers in the two paths, and squared norms of the entity and relation embeddings:

$$\phi = \lambda(\|\boldsymbol{c}\|_2^2 + \|\boldsymbol{w}\|_2^2 + \|\boldsymbol{p}\|_2^2 + \sum_{i=1}^{k} \|\boldsymbol{e_i}\|_2^2 + \|\boldsymbol{r}\|_2^2 \tag{11}$$

Algorithm 1 summarizes the training procedure for HyConvE. For each sampled fact, we first obtain the negative samples. Then, we compute the confidence scores for the sample tuples. Finally, HyConvE is trained in a mini-batch fashion iteratively.

Due to space limitation, it can be observed from the convergence curves of the model scalability experiments in Appendix A that HyConvE can achieve optimal performance in fewer epochs. Therefore, HyConvE can achieve better performance than other baseline models when applied to large-scale knowledge hypergraphs.

Algorithm 1: Training procedure for HyConvE

Input: N-ary KHG $\mathcal{H} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_\mathcal{O})$, the negative sampling rate N, n_{iter}=1000
Output: The score of each tuple
Init: $\boldsymbol{E}$ for $e \in \mathcal{E}$, $\boldsymbol{R}$ for $r \in \mathcal{R}$

```
for t = 1, ··· , n_iter do
    Sample a mini-batch F_batch ∈ F of size m_b;
    for each x := {r, e_1, e_2, ..., e_k} ∈ F_batch do
        Construct negative samples for fact x;
        v_1 ← compute 3D convolutional vector using (2);
        v_2 ← compute 2D convolutional vector using (6);
        score ← get the final score of the tuple (7);
    end
    Update learnable parameters w.r.t. gradients based on
    the whole objective in (9);
end
```

5 EXPERIMENTS

5.1 Experimental setup

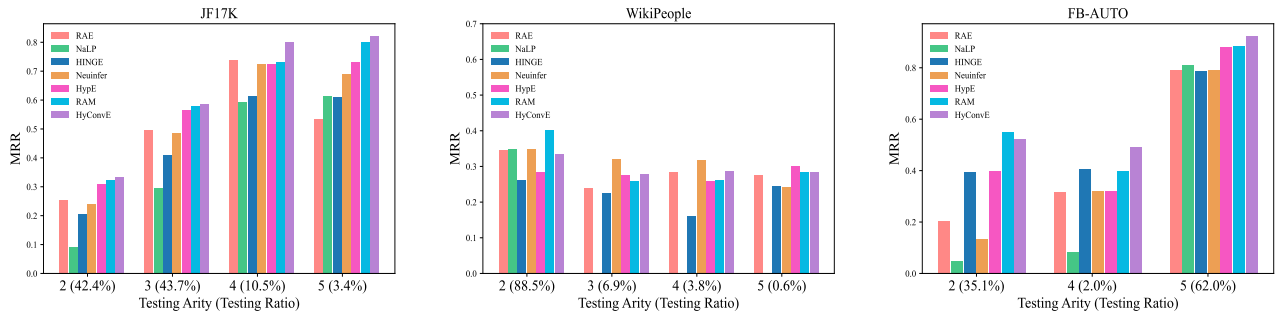
Dataset. To demonstrate the enhanced robustness and better generalization capabilities of HyConvE, we intensively delve into standard datasets in the relevant literature and select a representative of several widely used datasets. The experiments of knowledge hypergraph link prediction were conducted on three common benchmarks, i.e., JF17K [38], FB-AUTO [10], and WikiPeople [15]. WikiPeople is an n-ary knowledge hypergraph extracted from Wikidata [36] where all facts are related to people. The data in JF17K and FB-AUTO are all from Freebase [4], among which the multivariate data in JF17K accounts for a larger proportion, while facts in FB-AUTO are related to automotive. Since JF17K lacks a validation set, we randomly select 20% of the training set as validation set. Following the settings in GETD [21], experiments were also implemented on 4 subsets extracted from WikiPeople and JF17K with fixed arity, i.e., WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4. There are also four widely used benchmarks for knowledge graph link prediction: FB15k-237 [32], WN18RR [7], FB15k [5], and WN18

Table 1: Dataset Statistics. The size of the train, valid, and test columns represent the number of triples or tuples, respectively. "Arity" denotes the involved arities of relations.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	Arity	# train	# valid	# test	# arity=2	# arity=3	# arity=4	# arity $\geq$ 5
FB15k-237	14,541	237	2	272,115	17,535	20,466	310,116	–	–	–
WN18RR	40,943	11	2	86,835	3,034	3,134	93,003	–	–	–
JF17K	29,177	327	2-6	61,104	15,275	24,568	56,332	34,550	9,509	2,267
WikiPeople	47,765	707	2-9	305,725	38,223	38,281	337,914	25,820	15,188	3,307
FB-AUTO	3,388	8	2, 4, 5	6,778	2,255	2,180	3,786	–	125	7,212
JF17K-3	11,541	104	3	27,645	3,454	3,455	–	34,544	–	–
JF17K-4	6,536	23	4	7,607	951	951	–	–	9509	–
WikiPeople-3	12,270	66	3	20,656	2,582	2,582	–	25,820	–	–
WikiPeople-4	9,528	50	4	12,150	1,519	1,519	–	–	15188	–

Table 2: Results of Link Prediction on Knowledge Hypergraph Datasets. The best results are in boldface and the second best are underlined. Experimental results with "-" are those results that were not presented in the original paper. All experimental results are obtained locally.

Model	JF17K				WikiPeople				FB-AUTO			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
RAE [41]	0.392	0.312	0.433	0.561	0.253	0.118	0.343	0.463	0.703	0.614	0.764	0.854
NaLP [15]	0.310	0.239	0.334	0.450	0.338	0.272	0.362	0.466	0.672	0.611	0.712	0.774
HINGE[28]	0.473	0.397	0.490	0.618	0.333	0.259	0.361	0.477	0.678	0.630	0.706	0.765
NeuInfer [14]	0.451	0.373	0.484	0.604	0.351	0.274	0.381	0.467	0.737	0.700	0.755	0.805
HypE [10]	0.494	0.399	0.532	0.650	0.263	0.127	0.355	0.486	0.804	0.774	0.824	0.856
tNaLP+ [13]	0.449	0.370	0.484	0.598	0.339	0.269	0.369	0.473	0.729	0.645	0.748	0.826
S2S [8]	0.528	0.457	0.570	0.690	**0.364**	0.273	0.402	**0.503**	-	-	-	-
RAM [22]	0.539	0.463	0.572	0.690	0.363	0.271	**0.405**	0.500	0.830	0.803	0.851	0.876
HyConvE (ours)	**0.590**	**0.478**	**0.610**	**0.729**	0.362	**0.275**	0.388	0.501	**0.847**	**0.820**	**0.872**	**0.901**



(a) JF17K (b) WikiPeople (c) FB-AUTO

Figure 3: Breakdown performance across relations with different arities. x-axis identifies the relation arity and the ratio of testing samples. 6-ary relational facts and beyond are few and unreliable, thus not reported.

[5]. Despite their wide application to previous knowledge graph link prediction tasks, FB15k and WN18 suffer from serious data leakage problems, a logic rule-based link prediction model can easily achieve the best results on these two datasets according to [7], so we drop them and choose the other two datasets as the main validation benchmarks. The detailed statistics of the datasets are summarized in Table 1.

Baselines. For the task of knowledge hypergraph link prediction, we choose several state-of-the-art baselines, including translational model RAE [41], tensor decomposition model GETD [21], n-CP [21], n-Tucker [21], and S2S [8], and neural network based model HINGE [28], NeuInfer [14], and HypE [10]. For the task of knowledge graph link prediction, experiments were conducted with representative models such as translational model TransE [5], tensor decomposition model DistMult [39], ComplEx [33], in addition, state-of-the-art models HypE [10], RAM [22], and S2S [8] are also compared.

Evaluation Metrics. Similar to previous works, two metrics are utilized for model evaluation, mean reciprocal rank (MRR) and Hit@k ($k = 1, 3, 10$), respectively. Each entity is first replaced by all entities in the entity set to form a collection of candidate facts, among which those facts that exist in the training/validation/test set are filtered. Concretely, for each tuple $r(e_1, e_2, ..., e_k)$ in τ_{test} and each position

within the tuple, $|\mathcal{E}| - 1$ corrupted tuples are generated by replacing e_i with each of the entities in $\mathcal{E} \setminus \{e_i\}$. Then we score the given tuple combined with the candidate set and sort their scores in descending order. Let $rank_i(r(e_i, ..., e_k))$ be the ranking of $r(e_i, ..., e_k)$, denote MRR as $\frac{1}{K}\sum_{r(e_i,...,e_k)\in\tau_{test}}\sum_{i=1}^{k}\frac{1}{rank_i(r(e_i,...,e_k))}$, where $K = \sum_{r(e_i,...,e_k)\in\tau_{test}}$, α is the number of prediction tasks. We count the number of tuples in the test set that score within the top k and calculate the value of the Hit@k metric, which is a ratio determined by the top k counts and the number of test set tuples.

Hyper-parameters. In our experiments, we set the learning rate to 0.01, the batch size to 128, and the epochs to 500 for each dataset. We fix the entity and relation embedding size to 400 dimensions. In the 3D convolution path, we fix the output channel of the convolution filter to 6, the step size of the maximum pooling layer to $(2, 2, 1)$; in the 2D convolution, the size of the relation-specific filter and the position-specific filter are 1×3 and 1×9, the step size of the maximum pooling layer is $(1, 2)$, the learning rate is selected from 0.01, 0.005, 0.003, 0.001, 0.0005, and 0.0001, respectively. Dropout is used for regularization, chosen from 0.0, 0.2, 0.3, and 0.4, respectively. The implementation of HyConvE is available at at this GitHub link[1].

5.2 Results on Knowledge Hypergraphs

5.2.1 Results on mixed arity fact. In the experiment of mix-arity link prediction, facts with different arities are trained simultaneously. Results in the Table 2 shows the overall result of knowledge hypergraph link prediction. Figure 3 reports the breakdown performance on single-arity data.

According to Table 2, it can be seen that our model achieves a significant performance improvement on the JF17K and FB-AUTO datasets in comparison with all representative baselines. Specifically, early methods NaLP and RAE are tricky with complex network design and overfitting. The worse performance shown by NeuInfer and HINGE demonstrates the inevitable loss of structural and semantic information due to the introduction of decomposition, the inherent information in the original n-ary fact was broken, instead, we treat the n-ary fact as a tuple and the information is preserved more completely. Although works such as HypE and RAM also treat n-ary facts as a whole, they only extract position (role) information between entities within n-ary relations and hence perform poorly. Translation-based models and tensor-factorization-based models are able to extract surface semantic knowledge because they have translation operations, such as addition and multiplication. However, they are incapable to capture the latent and implicit knowledge due to their shallow structures. The overall better performance demonstrates the capability and effectiveness of our proposed HyConvE for capturing and extracting various types of information and knowledge. The breakdown performance of single-arity relation link prediction also shows the superiority of our proposed model. HyConvE performs consistently well over different arities on JF17K and FB-AUTO, while the relatively weak performance on the higher-arity data of the WikiPeople dataset is due to the unbalanced data distribution in the dataset with the dramatically large amount of sparsity in higher-arity relations shown in Appendix C. Notably, the performance of HyConvE is significantly improved over baselines for higher-arity relation ($n > 2$) per dataset, indicating that our model is able to adequately model the interaction of entities and relations within n-ary facts, including the extraction of surface and latent knowledge as well as the capture of position information and inherent patterns.

5.2.2 Results on fixed arity fact. Under the fixed-arity design, the facts of different arities are separated and trained independently, and the robustness of the models reflects in the superior performance of link prediction in both mixed-arity and fixed-arity settings. Besides, we also want to investigate whether the training of mixed-arity relational data has a promising effect on the learning of higher-arity relational data compared with single-arity relational data. Relations over 5-ary have been filtered due to their sparsity. Experiments were performed on 4 widely used subsets, i.e., WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4. The experimental results are presented in Table 3.

In particular, under the setting of fixed arity, the ternary and quaternary relations in JF17K and WikiPeople are modeled by different convolution processes, respectively. The experimental results demonstrate the modeling ability for HyConvE in fixed arity relations. Unlike NeuInfer, HINGE, and other methods that introduce decomposition to break the n-ary facts, we treat the n-ary facts as a tuple and model the compatibility of entities and relations by scoring each tuple after two convolutional paths so that the original information can be preserved more completely. The slightly worse results for ternary relational data in WikiPeople-3 are mainly affected by noise introduced by other relational data. Overall, HyConvE achieves better results in single-arity link prediction, demonstrating the superiority of our method and the way n-ary facts are handled. At the same time, it can be seen clearly from the results that during the mixed-arity training process in WikiPeople, there may be uneven distribution or noise among different arities of facts, and thus the link prediction of each arity is not as good as those under the fixed-arity setting.

5.3 Results on Knowledge Graphs

Since knowledge graph can be seen as a special case of knowledge hypergraphs, to demonstrate the compatibility and robustness of HyConvE in binary relational link prediction, we conducted experiments on representative knowledge graph datasets FB15k-237, WN18RR as well as the binary data in three knowledge hypergraph datasets. We take the representative translation-based model TransE, tensor-factorization-based model DistMult, and ComplEx as well as four knowledge hypergraph embedding model HypE, GETD, S2S, and RAM. The experimental results are presented in Table 4. Overall, HyConvE achieves fairly competitive performance on almost all metrics in almost all datasets. More importantly, HyConvE outperforms all other baseline models on MRR metrics which is of virtual importance in knowledge graph link prediction. Our proposed HyConvE can effectively model binary relational facts and can be well generalized to binary knowledge graphs, but due to the lack of positional and semantic information contained in the binary relations, compared with the excellent performance of our model on the knowledge hypergraph datasets, the performance of HyConvE is not significantly improved compared with the baselines.

[1]https://github.com/CarllllWang/HyConvE/tree/master

Table 3: Results on fixed arity datasets. The best results are in boldface and the second best are underlined.

Model	JF17K-3			JF17K-4			WikiPeople-3			WikiPeople-4		
	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
RAE [41]	0.505	0.430	0.644	0.707	0.636	0.835	0.239	0.168	0.379	0.150	0.080	0.273
NaLP [15]	0.515	0.431	0.679	0.719	0.673	0.805	0.301	0.226	0.445	0.342	0.237	0.540
n-CP [21]	0.669	0.613	0.801	0.754	0.701	0.855	0.313	0.237	0.476	0.253	0.163	0.432
n-tucker [21]	0.727	0.664	0.852	0.786	0.723	0.851	0.315	0.236	0.478	0.335	0.225	0.536
GETD [21]	0.725	0.660	0.858	0.822	0.761	0.924	**0.363**	**0.272**	**0.545**	0.346	0.229	0.542
RAM [22]	0.578	0.505	0.722	0.743	0.701	0.845	0.254	0.190	0.383	0.226	0.161	0.367
HyConvE (ours)	**0.729**	**0.670**	**0.861**	**0.827**	**0.770**	**0.931**	0.318	0.240	0.482	**0.386**	**0.271**	**0.607**

Table 4: Results of Link Prediction on Knowledge Graph Datasets. The best results are in boldface and the second best are underlined. Experimental results with "-" are those results that were not presented in the original paper. All experimental results are obtained locally.

Model	FB15k-237			WN18RR			JF17K			WikiPeople			FB-AUTO		
	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
TransE [5]	0.294	-	0.561	0.226	-	0.501	0.276	0.167	0.495	0.312	0.146	0.574	0.313	0.132	0.562
DistMult[28]	0.241	0.155	0.419	0.431	0.390	0.490	0.228	0.144	0.411	0.275	0.193	0.388	0.494	0.444	0.566
ComplEx [14]	0.253	0.158	0.428	0.440	0.411	0.512	0.308	0.219	0.498	0.326	0.232	0.461	0.487	0.442	0.568
HypE [10]	0.240	0.160	0.400	0.363	0.332	0.473	-	-	-	-	-	-	-	-	-
S2S [8]	**0.348**	**0.256**	**0.540**	**0.498**	**0.455**	**0.577**	-	-	-	-	-	-	-	-	-
RAM [22]	-	-	-	-	-	-	0.324	0.232	0.515	**0.408**	**0.313**	**0.568**	0.489	0.444	**0.576**
HyConvE (ours)	0.339	0.212	0.458	0.461	0.432	0.534	**0.338**	**0.246**	**0.525**	0.388	0.281	0.556	**0.493**	**0.445**	0.572

Table 5: Results of ablation study. The best results are in boldface. HyConvE-path1-only means to use only the 3D path of HyConvE when conducting experiments and HyConvE-path2-only means the other.

Model	JF17K				WikiPeople				FB-AUTO			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
HyConvE-path1-only	0.528	0.457	0.570	0.690	0.323	0.227	0.344	0.478	0.831	0.796	0.851	0.899
HyConvE-path2-only	0.102	0.054	0.094	0.168	0.072	0.048	0.094	0.172	0.145	0.082	0.164	0.212
HyConvE (ours)	**0.590**	**0.478**	**0.610**	**0.729**	**0.352**	**0.275**	**0.388**	**0.501**	**0.847**	**0.820**	**0.872**	**0.901**

5.4 Ablation study

Ablation experiments are essential to confirm the necessity of two convolutional paths. We chose three knowledge hypergraph datasets JF17K, FB-AUTO, and WikiPeople, to perform ablation experiments with the same hyper-parameters. Experimental results are presented in Table 5. When only using path 2 of HyConvE for link prediction, we observe a decline in model performance on all metrics. Specifically, on the JF17K dataset, using 3D convolutional path merely reduces the MRR by 6.2%, Hit@1 by 2.1%, Hit@3 by 4.0%, and Hit@10 by 3.9%. While on the FB-AUTO dataset, the reduction of the evaluation metrics is relatively stable. This gap highlights the effectiveness of 2D convolution, in which 2D relation-aware and position-aware filters are utilized successively to capture the intrinsic patterns and positional information in n-ary relations. As a comparison, we also design experiments to demonstrate the effectiveness of 3D convolutional path. Obviously, solely using the 2D convolutional path presents a significant gap in a series of evaluation metrics compared with the complete HyConvE, where MRR is reduced from 0.847 to 0.145 in FB-AUTO, from 0.352 to 0.072 in JF17K, and from 0.590 to 0.102 in WikiPeople, respectively. Overall, the experimental results of the ablation study show that a slightly competitive performance can be obtained using only the 3D convolutional path. But its final link prediction results are still worse than the complete HyConvE model due to the lack of relation and position information, while only using the 2D convolutional path leads to a dramatic drop in all metrics. Therefore, it is obvious that only the combination of 2D and 3D convolution of HyConvE can achieve best performance.

6 CONCLUSION

In this paper, we propose a novel convolutional-based embedding model for knowledge hypergraph link prediction called HyConvE. Considering the drawbacks of existing approaches, we fully exploits the characteristics of convolutional neural networks in knowledge hypergraph embedding. We use 3D convolution to extract the deeper interaction while preserving the translational property of entities and relations. While in the 2D convolutional process, we employee relation-specific and position-specific filters to capture the corresponding features. For future work, we will consider to leverage adjacent tuples for more structural information.

ACKNOWLEDGMENTS

This work was supported in part by The National Key R&D Program of China (2020AAA0108504), the National Natural Science Foundation of China (61972275), CAAI-Huawei MindSpore Open Fund (2022037A), and the Australian Research Council Linkage Project (LP180100750).

REFERENCES

[1] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*. Springer, 553–565.

[2] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590* (2019).

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific american* 284, 5 (2001), 34–43.

[4] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[6] Robin Burke. 2000. Knowledge-based recommender systems. *Encyclopedia of library and information systems* 69, Supplement 32 (2000), 175–186.

[7] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[8] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to sparsify tensor decomposition for n-ary relational data. In *Proceedings of the Web Conference 2021*. 4043–4054.

[9] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12, 7 (2011).

[10] Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2019. Knowledge hypergraphs: Prediction beyond binary relations. *arXiv preprint arXiv:1906.00137* (2019).

[11] Wenying Feng, Daren Zha, Lei Wang, and Xiaobo Guo. 2022. Convolutional 3D Embedding for Knowledge Graph Completion. In *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 1197–1202.

[12] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. *arXiv preprint arXiv:2009.10847* (2020).

[13] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo none Wang, and Xueqi Cheng. 2021. Link prediction on n-ary relational data based on relatedness evaluation. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[14] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. Neuinfer: Knowledge inference on n-ary facts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6141–6151.

[15] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In *The World Wide Web Conference*. 583–593.

[16] Jiale Han, Bo Cheng, and Xu Wang. 2021. Two-phase hypergraph based reasoning with dynamic relations for multi-hop KBQA. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 3615–3621.

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[18] Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 978–987.

[19] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems* 31 (2018).

[20] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[21] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing tensor decomposition for n-ary relational knowledge bases. In *Proceedings of The Web Conference 2020*. 1104–1114.

[22] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-aware modeling for n-ary relational knowledge bases. In *Proceedings of the Web Conference 2021*. 2660–2671.

[23] Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. *arXiv preprint arXiv:1811.04588* (2018).

[24] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2017. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121* (2017).

[25] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Icml*.

[26] Meng Qu, Xiang Ren, Yu Zhang, and Jiawei Han. 2018. Weakly-supervised relation extraction by pattern-enhanced embedding learning. In *Proceedings of the 2018 World Wide Web Conference*. 1257–1266.

[27] Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, and Yujia Bai. 2020. Knowledge graph embedding with atrous convolution and residual learning. *arXiv preprint arXiv:2010.12121* (2020).

[28] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond triplets: hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*. 1885–1896.

[29] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. Springer, 593–607.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[31] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[32] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 1499–1509.

[33] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR, 2071–2080.

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

[35] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082* (2019).

[36] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[38] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. *arXiv preprint arXiv:1604.08642* (2016).

[39] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

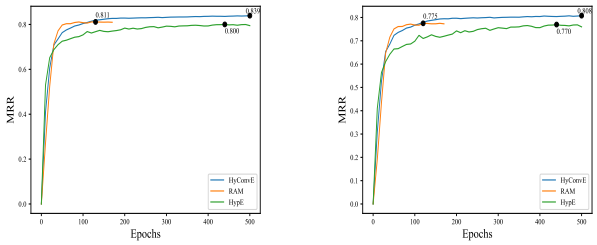
[40] Fuxiang Zhang, Xin Wang, Zhao Li, and Jianxin Li. 2020. TransRHS: A Representation Learning Method for Knowledge Graphs with Relation Hierarchical Structure.. In *IJCAI*. 2987–2993.

[41] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*. 1185–1194.

[42] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. 2020. AutoSF: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 433–444.

A CONVERGENCE CURVE

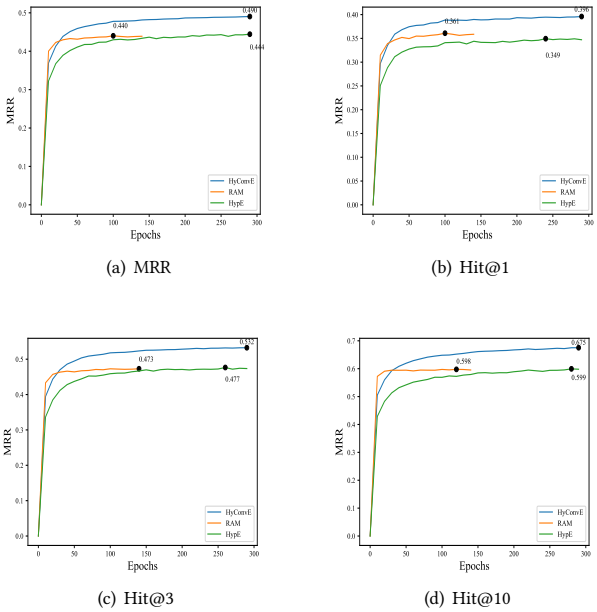
To make the comparison more vivid, we visualize the performance of HyConvE, RAM, and HypE models on the validation set on two datasets, JF17K and FB-AUTO.



(a) MRR (b) Hit@1

(c) Hit@3 (d) Hit@10

Figure 4: Comparison on the validation performance vs. training epoches of HyConvE, HypE, RAM in FB-AUTO.



(a) MRR (b) Hit@1

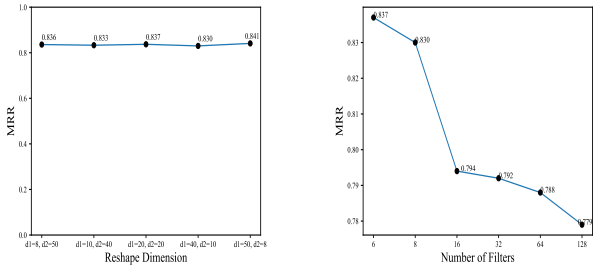
(c) Hit@3 (d) Hit@10

Figure 5: Comparison on the validation performance vs. training epoches of HyConvE, HypE, RAM in JF17K.

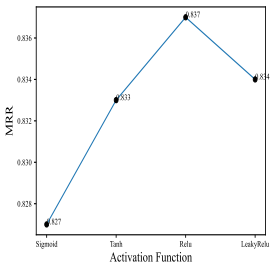
The results are shown in Figure 4 and 5, respectively. We can clearly observe that HyConvE outperforms the other models on all metrics throughout the validation process of both FB-AUTO and JF17K datasets. From Figure 4, it can be seen that RAM reaches convergence in the fastest number of iterations (about 100), and the model performance gradually decreases after 100 iterations, with its best MRR, Hit@1, Hit@3, and Hit@10 being 0.811, 0.775, 0.837, 0.880, respectively. In contrast, the performance of HyConvE and HypE continues to grow steadily with the number of iterations, even after 100 iterations.

B CONVOLUTION QUANTITATIVE ANALYSIS

In addition to the regular link prediction task, we also conducted quantitative analysis experiments on the FB-AUTO dataset. The variables to be explored are: the reshaping dimension, the number of 3D filters, and the different activation functions, respectively. These experiments are conducted on the FB-AUTO dataset for 500 epochs. (1) Figure 6(a) shows that model performance does not change significantly with different reshape dimensions. Note that in HyConvE the 3D convolution form is essential rather than the specific image shape (reshape dimension). (2) Figure 6(b) shows that simply increasing the number of filters does not result in better performance. When the number of filters is set to 6, the MRR reaches a maximum value of 0.837. (3) Figure 6(c) shows the most competitive activation function in HyConvE is Relu. This indicates that it is vital to select the proper activation function for training a good convolution model.

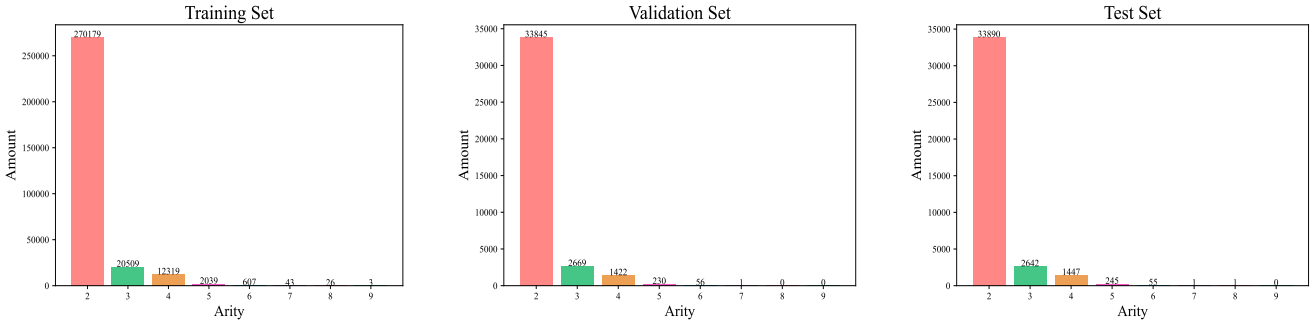


(a) Reshape Dimension (b) Number of Filters



(c) Activation Function

Figure 6: The effects of 3 convolution parameters.



(a) Training Set (b) Validation Set (c) Test Set

Figure 7: Distribution of relational data of different arities in the WikiPeople dataset.

C DATASET ANALYSIS

The distribution of relational data of different arities in the WikiPeople dataset is given in Figure 7. It can be seen that the distribution of data in the WikiPeople dataset is uneven, with binary relational data dominating (88.5%). Since the binary relational data contains less semantic and position information, our model is not as effective in the WikiPeople dataset compared with that in the JF17K and FB-AUTO datasets.