Heterogeneous Graph Neural Network with Hypernetworks for Knowledge Graph Embedding

Xiyang Liu[1], Tong Zhu[1], Huobin Tan[1](✉), and Richong Zhang[2]

[1] School of Software, Beihang University, Beijing, China
{liuxiyang,zhutong_software,thbin}@buaa.edu.cn

[2] SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
zhangrc@act.buaa.edu.cn

Abstract. Heterogeneous graph neural network (HGNN) has drawn considerable research attention in recent years. Knowledge graphs contain hundreds of distinct relations, showing the intrinsic property of strong heterogeneity. However, the majority of HGNNs characterize the heterogeneities by learning separate parameters for different types of nodes and edges in latent space. The number of type-related parameters will be explosively increased when HGNNs attempt to process knowledge graphs, making HGNNs only applicable for graphs with fewer edge types. In this work, to overcome such limitation, we propose a novel heterogeneous graph neural network incorporated with hypernetworks that generate the required parameters by modeling the general semantics among relations. Specifically, we exploit hypernetworks to generate relation-specific parameters of a convolution-based message function to improve the model's performance while maintaining parameter efficiency. The empirical study on the most commonly-used knowledge base embedding datasets confirms the effectiveness and efficiency of the proposed model. Furthermore, the model parameters have been shown to be significantly reduced (from 415M to 3M on FB15k-237 and from 13M to 4M on WN18RR).

Keywords: Knowledge graph embedding · Link prediction · Heterogeneous graph neural network

1 Introduction

Knowledge graphs (KGs), storing quantities of structured human knowledge in the form of triples (*subject entity*, *relation*, *object entity*), have been widely applied to many domains, such as question answering [4], recommendation systems [56], and dialog systems [29]. However, practical KGs, such as Freebase [3] and DBpedia [26], often suffer from incompleteness. As discussed in [10], 71% of people in Freebase have no known place of birth.

U. Sattler et al. (Eds.): ISWC 2022, LNCS 13489, pp. 284–302, 2022.
https://doi.org/10.1007/978-3-031-19433-7_17

To infer missing links in KGs, numerous knowledge graph embedding (KGE) methods are proposed. Embedding models that represent entities and relations in low-dimensional vector spaces, can preserve the semantics and inherent structures of KGs. The early works of this line tend to employ simple, shallow models to learn the information contained in KGs, e.g., the translational distance models [5,51,58] and the tensor factorization-based models [35,53]. Methods such as ConvE [8] and RSN [14] apply deep neural networks to capture more expressive features. These approaches focus on capturing the semantics preserved in a single triple without explicitly encoding the graph structures.

To better model the structural information, there has been an increasing interest in leveraging graph neural networks (GNN) to incorporate the connectivity structures of KGs into the embedding space [40]. Most recent GNN-based models such as VR-GCN [54], CompGCN [46], and HRAN [27] first represent entities and relations as discretized embeddings with the same dimension size. Then, the embeddings of the connected entities and relations are combined into mixed embeddings which are subsequently processed by a graph convolution operation. However, relations play a critical role in knowledge graphs. For example, the average occurrence of entities in FB15k-237 [44], a subset of Freebase, is 37.4, while the average occurrence of relations reaches 1148. We argue that such a mechanism learns relation embeddings in the lower-level layers of models, which is sub-optimal for GNNs to extract the rich semantics of relations. In this paper, we consider building a HGNN that learns relational representations in higher-level layers of the architecture.

Although a bench of heterogeneous graph neural networks is proposed to achieve better performance than homogeneous GNNs in many tasks [18–20], most HGNNs are incapable of KGs. HGNNs typically define independent network parameters for different types of nodes and edges, where the number of parameters will be explosively increased with the kinds of types. Such fact limits HGNNs to handle graphs with fewer edge types and is not applicable for KGs with hundreds and even thousands of relations. For example, MAGNN [12] is a recent HGNN model of popular metapath-based methods. Assuming we leverage MAGNN to process a KG with 100 relations, the number of 2-order metapaths will be $10,000$. Each metapath will be assigned separate parameters, which results in an immense number of trainable weights. Hence, MAGNN is suitable for datasets with serval kinds of edges, but it is not reasonable to apply the model to KGs without alteration. Similar issues broadly exist in other HGNN methods [18–20,50,55].

To solve the problems mentioned above, we believe that a key challenge in designing dedicated HGNNs for KGs is *how to effectively capture the high heterogeneity of relations while controlling the number of type-related parameters*?

In this paper, we propose a novel **H**eterogeneous **K**nowledge **G**raph neural **N**etwork (HKGN) incorporated with external networks called hypernetworks for parameter generation to resolve the aforementioned challenge. In our method, hypernetworks are effective at controlling the parameter counts through explicitly encoding the common information among relations. The rationale is that relations are not completely irrelevant, and there exist correlations among relations. Zhu et al. [58] found that there exists a low-rank structure over different relation embeddings. Specifically, we first introduce a primary HGNN utilized to process multiple single-relational subgraphs split from original KGs. We then incorporate external hypernetworks [15] into the primary HGNN to obtain essential parameters for graph convolution. To encode the rich information of relations, HKGN learns multiple independent representations of relations in different model layers. For each relation, we extend the linear transformation adopted in the message function to a convolution neural network for capturing the expressive feature combinations. To control the number of relation-specific parameters, hypernetworks are defined as mapping functions that take relational weight vectors with appropriate dimensions as input and output the corresponding parameters. HKGN eventually learns the abundant semantics of relations and reduces plentiful meaningless trainable parameters.

In summary, our main contributions are as follows:

– We present HKGN - a dedicated heterogeneous graph neural network for KGs which decouples relation embeddings as input of hypernetworks from entity embeddings as input of primary multi-relational graph convolution network. We leverage a relation-aware 2D convolution to promote the model's capability in learning representative information from neighbors, while external hypernetworks are introduced to keep parameters efficient.
– We conduct experiments on two benchmarks to evaluate the link prediction capability of HKGN and show that HKGN obtains better performance than state-of-the-art (SOTA) embedding models.

2 Related Work

2.1 Heterogeneous Graph Neural Network

Heterogeneous Graph Neural Networks are brought forward to handle ubiquitous heterogeneous graphs like academic network [18], product review network [19], and educational data [20]. HAN [50] applies node-level and semantic-level attention on metapath based graphs. HetGNN [55] aggregates node features transformed based upon node types sampled by random walks. HGT [18] designs a heterogeneous mutual transformer-like attention mechanism to propagate messages. These HGNNs outperform previous homogeneous GNNs like GCN [25] and GAT [47] on graph data with substantial heterogeneity. However, all these HGNNs are expected to handle heterogeneous graphs with few edge types and are not scalable to KGs in model complexity and computational demand.

2.2 Hypernetwork

Hypernetwork is a neural network trained to generate the weights for another network (called the primary network). Hypernetworks are first introduced for visual tasks like DFN [21] and SRCNN [38], in which the convolutional weights are dynamically generated dependent on input images. HyperRNN [15] uses a recurrent network to generate the parameters of another recurrent network.

Recently, hypernetworks have also been applied in graph neural networks. Nachmani and Wolf [30] utilize an MLP to get the GNN's weights. They find that the performance of the primary GNN can be improved when the first message and current message are combined as the input of the hypernetwork [31]. LGNN [28] employs dense layers to obtain node-level and edge-level localized weights. In this study, we incorporate hypernetworks into HGNNs to alleviate the problem of the count of heterogeneous parameters increasing explosively.

2.3 Knowledge Graph Embedding

Knowledge graph embedding has received considerable attention in recent years. Translation-based models treat relations as translations from subject entities to object entities, such as TransE [5] and TransH [51]. Tensor factorization-based methods regard a KG as a high-dimensional sparse tensor that can be factorized into smaller tensors, such as RESCAL [35] and DistMult [53]. HypER [1] proposes a hypernetwork to generate 1D relation-specific filters. CoPER [42] is a recent approach using relation embeddings to generate parameters of two basic models: ConvE [8] and MINERVA [7].

Graph neural network has been utilized in learning KGE, which has yielded promising performance [40]. However, many recent GNN-based works, such as VR-GCN [54], CompGCN [46], and KBGAT [32], learn relation embeddings in lower-level layers of the models. We argue that such a mechanism can lead to the limited representational power of GNNs.

3 Methodology

In this section, we elaborate on the details of the proposed HKGN. The knowledge graph is a collection of triples (*subject*, *relation*, *object*) denoted as (s, r, o). All triples are connected to form a heterogeneous graph denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ with $\mathcal{E}$ as the entity set, $\mathcal{R}$ as the relation set, and $\mathcal{T}$ as the graph edge set. Following [40], a corresponding inverse triple (o, r^{-1}, s) is created with inverse relation r^{-1}, for each (s, r, o). To ensure that entities can receive information from themselves, we add self-connection $\top$ to each entity. In that case, the relation set and the graph edge set are extended as

$$\mathcal{R}' = \mathcal{R} \cup \left\{r^{-1} | r \in \mathcal{R}\right\} \cup \{\top\} \tag{1}$$

$$\mathcal{T}' = \mathcal{T} \cup \mathcal{T}^{-1} \cup \{s, \top, s \mid s \in \mathcal{E}\} \tag{2}$$

$$\mathcal{T}^{-1} = \left\{(o, r^{-1}, s) \mid (s, r, o) \in \mathcal{T}\right\} \tag{3}$$

Figure 1 illustrates an overview of HKGN. The architecture of HKGN follows the encoder-decoder scheme widely adopted by GNN-based KGE models. In the encoding phase, HKGN can be seen as an instantiation of the Message Passing Neural Network (MPNN) [13] framework. A message function is performed to encode the semantics of a single edge in latent space. Then, a multi-relational message propagation function is designed to contextualize entity embeddings with localized neighborhood structures by aggregating messages from neighbor edges. The relational parameters (also called heterogeneous parameters in this paper) utilized in this process are all generated by external hypernetworks. HKGN takes ConvE, one of the most generally used scoring functions, as the decoder to infer missing triples.

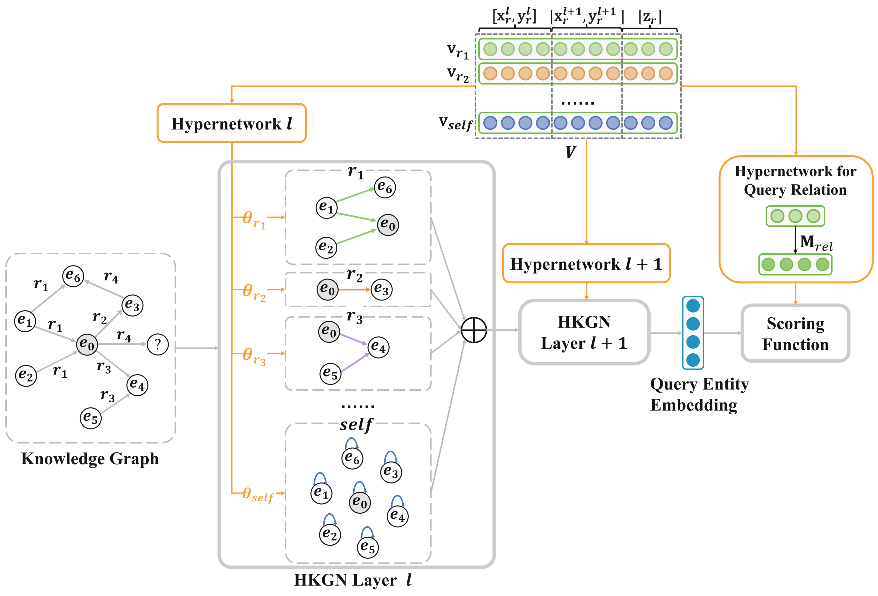

Fig. 1. An overview of HKGN. HKGN consists of two modules: the hypernetworks module used to obtain weights and the primary HGNN module utilized to process the multi-relational knowledge graph.

3.1 Hypernetworks for HGNNs

To flexibly control the number of trainable heterogeneous parameters in HGNNs, we evolve hypernetworks for parameter generation. A hypernetwork derived for HGNNs is a mapping function that takes the independent weights $\mathbf{w}_r^l$ of relation r as input and outputs the needed parameters $\boldsymbol{\theta}_r^l$:

$$\boldsymbol{\theta}_r^l = g^l(\mathbf{w}_r^l, \boldsymbol{\theta}_g^l) \tag{4}$$

– $\mathbf{w}_r^l$ describes the unique information about the structure of the weights specific to relation r. In this study, $\mathbf{w}_r^l$ is just considered as vectorized representations $\mathbf{v}_r^l \in \mathbb{R}^{d_r^l}$. d_r^l is the dimension of relation vector $\mathbf{v}_r^l$.

– $\boldsymbol{\theta}_g^l$ represents the global semantics shared among relations. Let n_g denote the number of parameters defined in $\boldsymbol{\theta}_g^l$.
– $g(\cdot)^l$ can be an arbitrary reasonable mapping function from relational input vector $\mathbf{v}_r^l$ to parameters required by HGNNs. A special case is adopting the lookup operation, which means there will be no $\boldsymbol{\theta}_g^l$ and the model is acting the same as normal HGNNs.
– l denotes the generated $\boldsymbol{\theta}_r^l$ utilized in layer l of HGNNs.

The above formulation compresses the heterogeneous parameter $\boldsymbol{\theta}_r^l$ into the vector $\mathbf{v}_r^l$. n_p^l denotes the parameter counts of $\boldsymbol{\theta}_r^l$. Assuming a knowledge graph with N_r relations, we can calculate a ratio q to assess the hypernetwork's ability in parameter reduction, omitting l for simplicity:

$$q = \frac{n_p \times N_r}{d_r \times N_r + n_g} = \frac{n_p}{d_r + \frac{n_g}{N_r}} \tag{5}$$

The effect of hypernetworks in controlling parameter counts is more significant with a bigger q value. Equation (5) shows that the impact of hypernetworks with the same architecture differs for KGs with various relation amounts. Knowledge graphs with more relations will be more likely affected by hypernetworks.

3.2 Message Construction with Hypernetworks

Keeping a separate weight matrix for each node or edge type is the most commonly applied mechanism for HGNNs to model the heterogeneity. For example, R-GCN [40] uses a relation-specific linear transformation to model the relational patterns. Given an edge (s, r, o), the representation of incoming message is learned as:

$$\mathbf{m}_{(r,o)} = \mathbf{W}_r \mathbf{e}_o \tag{6}$$

where $\mathbf{e}_o \in \mathbb{R}^d$ denotes the embedding of object entity o and $\mathbf{W}_r \in \mathbb{R}^{d' \times d}$ is the weight matrix assigned to r. Every element $\mathbf{m}^{[i]}$ in message embedding $\mathbf{m}$ is the weighted sum of all features from entity embedding $\mathbf{e}_o$:

$$\mathbf{m}_{(r,o)}^{[i]} = \sum_{j=0}^{d} \mathbf{W}_r^{[i,j]} \mathbf{e}_o^{[j]} \tag{7}$$

This type of transformation can implicitly lead to the restricted expressive capability of models because it has an intrinsic flaw in recognizing patterns that original features can be combined as a whole to contribute to messages. The importance of feature $\mathbf{e}_o^{[j]}$ is measured by coefficient $\mathbf{W}_r^{[i,j]}$ independently of other features to derive $\mathbf{m}_{(r,o)}^{[i]}$, as in Eq. (7). Nevertheless, some feature combinations may be prominent for relation r, and some other feature combinations play a pivotal role in another relation $\hat{r}$.

In this paper, we introduce a relational convolution layer to explore rich feature combinations:

$$\mathbf{c}_{(r,o)} = \sigma(Re2D(\mathbf{e}_o) * \boldsymbol{\omega}_r) \tag{8}$$

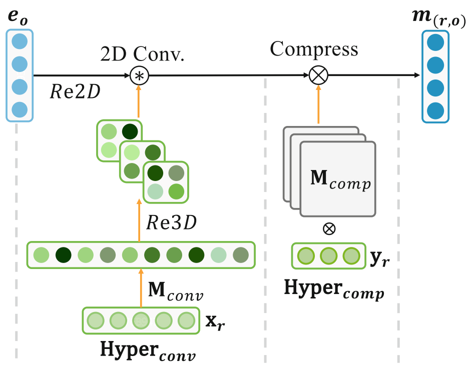

Fig. 2. An illustration of constructing messages incorporated with hypernetworks.

where $\mathbf{e}_o$ is reshaped into a 2D feature map and fed to a relation-aware 2D convolution layer with filters $\boldsymbol{\omega}_r \in \mathbb{R}^{f \times k_{size} \times k_{size}}$ to extract more interactions. As suggested in [8], compared with 1D convolution, one element in entity vector $\mathbf{e}_o$ can interact with distant elements rather than immediate elements by applying 2D convolution. The activation function $\sigma(\cdot)$ is chosen to be $tanh(\cdot)$. $\boldsymbol{\omega}_r$ interacts with diverse regions in $Re2D(\mathbf{e}_o)$ to generate different entries of $\mathbf{c}_{(r,o)}$. We notice that a range of convolution-based embedding methods like circular convolution [45], inception network [52], etc., can be adopted for further performance improvement.

To obtain embeddings with suitable dimensions for subsequent layers, a linear transformation operation is performed on $\mathbf{c}$ as:

$$\mathbf{m}_{(r,o)} = \mathbf{W}_r vec\left(\mathbf{c}_{(r,o)}\right) \tag{9}$$

where the resultant map is flattened into the 1D vector by vector concatenation operator $vec(\cdot)$ and $\mathbf{W}_r \in \mathbb{R}^{d' \times d^c}$ is the transformation matrix.

We have now defined convolution filters and weight matrix for each relation, which may incur a severe over-parameterization problem. To alleviate this issue, we exploit hypernetworks to improve the efficiency of heterogeneous parameters. The convolution filters are generated by the hypernetwork Hyper$_{conv}$:

$$\boldsymbol{\omega}_r = Re3D(\mathbf{M}_{conv}\mathbf{x}_r) \tag{10}$$

where $\mathbf{M}_{conv} \in \mathbb{R}^{f \cdot k_{size} \cdot k_{size} \times d_x}$ denotes the weight matrix and $\mathbf{x}_r \in \mathbb{R}^{d_x}$ is the relation vector. The transformed vector $\mathbf{M}_{conv}\mathbf{x}_r$ is reshaped into the 3D tensor $\boldsymbol{\omega}_r$ with the shape $f \times k_{size} \times k_{size}$. In practice, we have also experimented with employing a multi-layer perceptron $MLP(\cdot)$ to get $\boldsymbol{\omega}_r$. However, we found that the simple linear projection achieves the best results, which may be contributed to the better generalization of linear projection than $MLP(\cdot)$.

To limit the number of learnable parameters in $\mathbf{W}_r$, we introduce another hypernetwork Hyper$_{comp}$:

$$\mathbf{W}_r = <\mathbf{M}_{comp}, \mathbf{y}_r> \tag{11}$$

where $<,>$ denotes the tensor dot product between global tensor $\mathbf{M}_{comp} \in \mathbb{R}^{d' \times d^c \times d_y}$ and relation weight vector $\mathbf{y}_r \in \mathbb{R}^{d_y}$. The message construction approach applied in HKGN is depicted in Fig. 2.

In this paper, we let hypernetworks process all relation vectors in the same procedure. Hypernetworks can be arbitrary network architectures such as convolution neural networks or graph neural networks [31]. More carefully designed hypernetworks may let information flow among relations more reasonably. We defer this for future work.

3.3 Multi-relational Message Propagation

The central entity s can be surrounded by different neighboring triples, where the entities under the same relation are located in a single-relational graph. To aggregate information from the neighborhood, we first split neighboring triples into diverse single-relational subgraphs according to their relations. The same neighbor entity o located in different single-relational subgraphs will be processed by distinct message functions. Then we compute the new representation for each entity in the $(l+1)$-th layer by accumulating message embeddings learned in the l-th layer from single-relational graphs:

$$\mathbf{e}_s^{l+1} = \sigma\left(\sum_{r \in \mathcal{R}'} \sum_{o \in \mathcal{N}_s^r} \mathbf{m}_{(r,o)}^l\right) \tag{12}$$

where $\mathcal{N}_s^r$ is the set of neighboring entities under relation r.

To avoid mutual interference among network parameters of different GNN layers, we employ independent hypernetworks for each HKGN layer. So far, the parameter associated with relation r is a segmented vector:

$$\mathbf{v}_r = [\ldots, \mathbf{x}_r^l, \mathbf{y}_r^l, \mathbf{x}_r^{l+1}, \mathbf{y}_r^{l+1} \ldots, \mathbf{z}_r] \tag{13}$$

where $\mathbf{x}_r^l$ and $\mathbf{y}_r^l$ are the input weight vectors for hypernetworks Hyper_{conv}^l and Hyper_{comp}^l of the l-th layer, respectively. The length of $\mathbf{v}_r$ is agnostic to the dimension d of entity embedding. Each segment represents the distinct role of relation r during graph convolution and is non-interfering from each other.

$\mathbf{z}_r \in \mathbb{R}^{d_z}$ appeared in Eq. (13) is utilized by the hypernetwork Hyper_{rel} to generate the target relation embedding which is later used in the scoring function to estimate the probability of query triple $< h, r, ? >$:

$$\mathbf{q}_r = \mathbf{M}_{rel}\mathbf{z}_r \tag{14}$$

where $\mathbf{M}_{rel} \in \mathbb{R}^{d' \times d_z}$ is the global projection matrix.

3.4 Scoring Function

To estimate the probability of query triple (h, r, t), GNN-based methods typically employ convolution-based models as the scoring function like ConvE [8], ConvKB

[33], CapsE [34], etc. In this work, we utilize ConvE to validate our model's effectiveness. As revealed by [43], the evaluation process applied in ConvE is rigorous and fair when dealing with candidate triples with the same score. In contrast, the biased evaluation protocol adopted in ConvKB and CapsE can lead to inappropriate performance improvement.

Given query entity embeddings $\mathbf{e}_h^L$, $\mathbf{e}_t^L$ and relation embedding $\mathbf{q}_r$, the scoring function can be written formally as:

$$p = \sigma\left(f\left(vec\left(f(\left[\overline{\mathbf{e}_h^L}; \overline{\mathbf{q}_r}\right] * \boldsymbol{\omega})\right)\mathbf{W}\right)\mathbf{e}_t^L\right) \tag{15}$$

where $f(\cdot)$ and $\sigma(\cdot)$ are *ReLU* and *sigmoid* activation functions. $\overline{\mathbf{e}_h^L}$ and $\overline{\mathbf{q}_r}$ denote 2D reshapings of $\mathbf{e}_h^L$ and $\mathbf{e}_t^L$. $[;]$ represents the concatenation operation and W is the projection matrix. The model is trained using cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_i t_i log(p_i) + (1 - t_i) log(1 - p_i) \tag{16}$$

where t_i is the label of triple i and p_i is the corresponding score.

3.5 Training Strategy

In this study, two different training strategies are applied to the HKGN to make a trade-off between GPU memory footprints and learning time:

1. **Parallel**: Performing message construction and propagation functions of multiple single-relational graphs simultaneously. In this case, all triples in the KG will be assigned heterogeneous parameters based on their binary relations and be processed in parallel. The strategy leads to higher memory consumption and less training time.
2. **Iterative**: Processing single-relational subgraphs iteratively. Triples with one type of relation are handled during each iteration. The strategy requires a lower GPU memory footprint with a longer learning time.

HKGN adopts the 1-N scoring developed by ConvE [8], where all entity embeddings will be updated by message propagation in each training step. Though it has an expensive GPU memory requirement (as revealed in Sect. 4.6), we find that 1-N scoring achieves better results than 1-n (10, 100, etc.) scoring.

4 Experiment

4.1 Datasets

To evaluate the performance of HKGN on link prediction task, two commonly used benchmark datasets (FB15k-237 [44] and WN18RR [8]) are employed in this study. FB15k-237 and WN18RR are derived from FB15k and WN18 datasets [5], respectively, where original inverse relations are excluded to prevent the test leakage problem. The statistics of the two datasets are summarized in Table 1.

Table 1. Statistics of datasets.

	#Entities	#Relations	#Training	#Validation	#Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Note that we have added new inverse relation r^{-1} and self-loop $\top$ relation, as presented in Sect. 3. The "inverse relation" described here is different from that mentioned above. Because the inverse triples are created for train/valid/test set separately, no test leakage problem will be caused. In that case, HKGN handles 475 relations in FB15k-237 and 23 relations in WN18RR. The number of edge types is far more than those in graphs often analyzed by HGNNs, such as IMDB (2 edge types) [23], ACM (4 edge types) [57], and DBLP (3 edge types) [49].

4.2 Evaluation Protocol

Following the evaluation protocol applied broadly in previous works, each test triple (h, r, t) is estimated in two different scenarios: head entity prediction $(?, r, o)$ and tail entity prediction $(h, r, ?)$. The head entity prediction is performed in the form of $(t, r^{-1}, ?)$ with the corresponding inverse relation. The head or tail entity is replaced by every other entity $e' \in \mathcal{E}$, and then each candidate triple is assigned a predictive value by the scoring function. Subsequently, we sort these scores in descending order to obtain the exact rank of the correct triple in the candidates. Similar to most baselines, we report the experimental results using the *filtered* setting introduced by [5], where all true triples in KG are excluded before ranking. Three standard metrics are reported to evaluate performance, Mean Reciprocal Rank (MRR), Mean Rank (MR), and Top 1, 3, 10 (Hits@1, Hits@3 and Hits@10).

4.3 Baselines

Many studies on learning knowledge graph embedding have emerged. To demonstrate the effectiveness of our model, we compare it with SOTA baselines categorized as the following groups:

– Shallow KGE models with low time and space complexity, including TransE [5], DistMult [53], and DualE [6].
– Convolution-based methods like ConvE [8], ConvR [22], HypER [1] and CoPER [42].
– Methods utilizing graph neural networks, which include R-GCN [40], SACN [41], VR-GCN [54], A2N [2], KBGAT [32], CompGCN [46] and HRAN [27].

4.4 Hyper-parameter Settings

We evaluate the results of hyperparameter choices on the validation splits. In HKGN, the relation embedding size d_x of $\mathbf{x}_r$ is selected from $\{50, 100, 200, 300\}$,

d_y of $\mathbf{y}_r$ from $\{2, 4, 6, 8\}$, d_z of $\mathbf{z}_r$ from $\{50, 100, 200, 300\}$. Finally, we find that the following choices work well on both datasets: $d_x = 100$, $d_y = 2$ and $d_z = 100$. The initial entity embedding size is chosen to be 100. The number of convolution filters is set to 32 with kernel size 3×3. We use Adam [24] with an initial learning rate $lr = 0.001$ to optimize the model up to 1200 epochs. The number of HKGN layers L is set to 2, 1; the batch size b is set to 1024, 256 for FB15K-237 and WN18RR, respectively.

4.5 Results of Link Prediction

Table 2. Link prediction results of HKGN and baselines on FB15k-237 and WN18RR. Results of [△] are taken from [39], [◊] from [43]. CoPER [†] is reevaluated by using the authors' open-source code. Other results are taken directly from the corresponding original papers.

Model	FB15k-237					WN18RR				
	MRR	MR	Hits@N			MRR	MR	Hits@N		
			1	3	10			1	3	10
TransE [△]	0.313	-	0.221	0.347	0.497	0.228	-	0.053	0.368	0.520
DistMult [△]	0.343	-	0.250	0.378	0.531	0.452	-	0.413	0.466	0.530
DualE	0.330	-	0.237	0.363	0.518	0.482	-	0.440	0.500	0.561
ConvE [△]	0.339	-	0.248	0.369	0.521	0.442	-	0.411	0.451	0.504
ConvR	0.350	-	0.261	0.385	0.528	0.475	-	0.443	0.489	0.537
HypER	0.341	250	0.252	0.376	0.520	0.465	5798	0.436	0.477	0.522
CoPER [†]	0.320	390	0.234	0.351	0.491	0.442	5315	0.418	0.450	0.487
R-GCN	0.248	-	0.153	0.258	0.414	-	-	-	-	-
SACN	0.350	-	0.260	0.390	0.540	0.470	-	0.430	0.480	0.540
VR-GCN	0.248	-	0.159	0.272	0.432	-	-	-	-	-
A2N	0.317	-	0.232	0.348	0.486	0.450	-	0.420	0.460	0.510
KBGAT [◊]	0.157	270	-	-	0.331	0.412	**1921**	-	-	0.554
CompGCN	0.355	197	0.264	0.390	0.535	0.479	3533	0.443	0.494	0.546
HRAN	0.355	**156**	0.263	0.390	0.541	0.479	2113	**0.450**	0.494	0.542
HKGN	**0.365**	171	**0.272**	**0.402**	**0.552**	**0.487**	2468	0.448	**0.505**	**0.561**

Table 2 summarizes the results of link prediction of HKGN and baselines. Ruffinelli et al. [39] have recently performed extensive experiments using popular KGE model architectures and training strategies with a wide range of hyperparameter settings. They found that many shallow models can achieve competitive performance when trained appropriately. Here we take their reported results of TransE, DistMult, and ConvE. Sun et al. [43] investigated the inappropriate evaluation and test data leakage problem in KBGAT. Hence, we take the results of KBGAT from [43] when issues are fixed. The performance of DualE is reported without prior type constraints. CoPER is closely related to our method. Nevertheless, we find that the original model performance is just estimated in the tail prediction task. To ensure a fair comparison, we evaluate the performance of CoPER in head and tail prediction tasks and report the average results using the authors' open-source code[1].

From Table 2, we observe that: (**i**) HKGN consistently outperforms all baselines on most metrics in two benchmark datasets, demonstrating the effectiveness of our proposed method. Compared with HRAN, a recent HGNN model which introduces a heterogeneous relation attention mechanism to aggregate neighbor features, HKGN delivers better results. The improvement indicates HKGN's ability to model complex structures in heterogeneous KGs. (**ii**) HKGN outperforms HypER and CoPER, which also utilize hypernetworks in learning KGE, showing that the graph structure information is beneficial for link prediction.

Table 3. Experimental results on FB15k-237 by relation category.

		1-1(1.5%)	1-N(4.6%)	N-1(18.6%)	N-N(75.2%)
MRR	CoPER	0.427	0.249	0.409	0.297
	CompGCN	**0.469**	0.278	0.435	0.334
	HKGN	0.400	**0.282**	**0.452**	**0.345**
Hits@10	CoPER	0.526	0.360	0.494	0.501
	CompGCN	**0.593**	0.414	0.531	0.549
	HKGN	0.591	**0.416**	**0.562**	**0.561**

We investigate the performance of HKGN for different relation categories on FB15k-237. Following [51], all relations are categorized into four classes: 1-to-1, 1-to-N, N-to-1, and N-to-N. Table 3 presents the results of HKGN on different relation categories. We reproduce CompGCN[2] and CoPER based on publicly available source codes. Table 3 shows that the HKGN performs better for 1-to-N, N-to-1, and N-to-N relation categories, which shows that HKGN is effective at handling complex relations. We also notice that CompGCN outperforms our model on 1-to-1 relations. The phenomenon can be attributed to the reason that the characteristic information of 1-to-1 relations is corrupted by noises from other relations. It reminds us that the performance of HKGN may be boosted by introducing residual connection [17] or gate-based structures into hypernetworks to control the flow of information from other relations. Overall, triples with 1-to-1 relation cover only around 1.5% of all edges in the FB15k-237 training set. Hence, the stronger ability of HKGN in modeling complex relations makes it more applicable to KGs.

[1] https://github.com/otiliastr/coper.
[2] https://github.com/malllabiisc/CompGCN.

4.6 Ablation Study

Table 4 shows the results of the ablation study. Conv. denotes the relational convolution applied in the message function and Hyper. represents all hypernetworks adopted in HKGN.

Table 4. Ablation study on FB15k-237 and WN18RR.

Dataset	Model	q value	MRR	Hits@1	Hits@3	Hits@10	#param.
FB15k-237	w/o Conv.	150.9	0.355	0.264	0.39	0.536	1.64M
	w/o Hyper.	-	0.358	0.265	0.394	0.544	415.19M
	HKGN	212.3	0.365	0.272	0.402	0.552	3.40M
WN18RR	w/o Conv.	7.4	0.459	0.41	0.479	0.553	4.15M
	w/o Hyper.	-	0.483	0.444	0.497	0.558	13.52M
	HKGN	10.8	0.487	0.448	0.505	0.561	4.96M

Effect of Relational Convolution. As shown in Table 4, removing relational convolution leads to all metrics (MRR, Hits@1, 3, and 10) degradation on FB15k-237 and WN18RR datasets, demonstrating the effectiveness of capturing feature combinations. We further examine how the prediction results of entities with different degrees can be affected by the relational convolution. The entity with a larger degree is connected to more neighbor entities, and this kind of entity is expected to receive more semantic information from neighbors. For each entity e, we compute the degree (indegree and outdegree) $deg(e) = deg_{in}(e)+deg_{out}(e)$ by counting the corresponding training triples. Figure 3 presents the average results of Hits@10 for different sets of entities with different degree scopes. It can be observed that HKGN achieves better performance across all degree scopes by leveraging the relation-specific convolution. Along with the increase of degrees, the average value of Hits@10 increases initially but declines abruptly in a high-degree scope (e.g., $deg(e) \geq 1000$ in FB15k-237 and $deg(e) \geq 100$ in WN18RR). We conjecture this is primarily due to the representative information being covered by excessive messages from too many neighbors. One solution to this problem may be neighborhood sampling functions as in [16].

Effect of Hypernetworks. The empirical results in Table 4 show that the model eliminating hypernetworks obtains 122x (FB15k-237) and 2.7x (WN18RR) more parameters and simultaneously worse performance than the original HKGN. The significant parameter reduction and consistent performance improvement indicate that hypernetworks are powerful in keeping parameters efficient. The average training loss and MRR evaluated on the validation set for FB15k-237 during the training process are reported in Fig. 4. Obviously, the HKGN with hypernetworks achieves lower loss and higher MRR results. Using

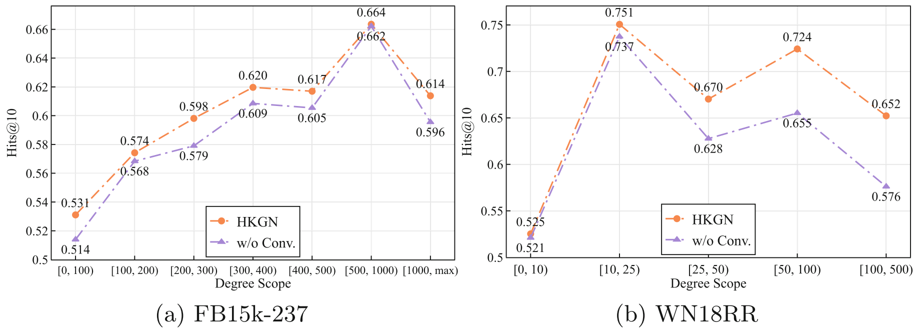

Fig. 3. Node degree study using FB15k-237 and WN18RR datasets.

hypernetworks helps reduce numerous network parameters and facilitates the weights converging to a more optimal solution. Without hypernetworks, the number of trainable parameters in the HKGN for handling FB15k-237 has reached 415M, which is even larger than some recent pre-trained language models, including GPT (110M) [37] and BERT (340M) [9]. Imagine we build HGNNs equipped with more complicated graph learning mechanisms such as transformer-style attention [18], metapath-based learning [12], and heterogeneous graph structure learning [57] to deal with web-scale knowledge graphs like Wikidata (4k relations) [48] and DBpedia (60k relations) [26], as summarized in [11]. There will be a blowup in the number of trainable parameters. Our experimental results suggest that it is possible to control the parameter counts and even boost performance by exploiting the underlying correlations among relations.

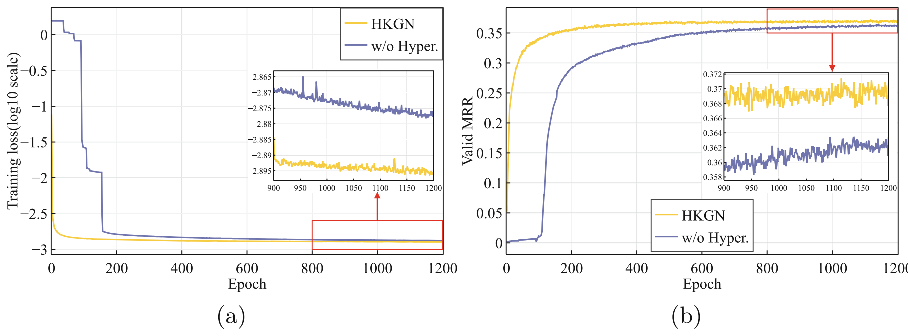

Fig. 4. Convergence results of HKGN and its counterpart without hypernetworks on FB15k-237 dataset.

The statistics of the maximum GPU memory allocated and the training time for HKGN with/without hypernetworks are shown in Table 5. The results are reported under the experimental environment with a single NVIDIA A100 GPU, CUDA 11.2, and PyTorch [36] 1.8.0. With sufficient GPU memory, executing the message functions of multiple single-relational graphs in parallel gets nearly 12x and 3.8x speedup on FB15k-237 and WN18RR. When processing single-relational graphs iteratively, introducing hypernetworks into the HKGN does not slow the training speed because all hypernetworks are implemented linearly. Utilizing hypernetworks has a more prominent influence on GPU memory consumption. When we tried to run the model without hypernetworks in a parallel way, the procedure attempted to load 852 GB and 327 GB of memory for FB15k-237 and WN18RR, respectively, and got out of memory error. The memory consumption goes far beyond the limit of our resources and can be reduced remarkably to 21 GB (FB15k-237) and 5 GB (WN18RR) through applying hypernetworks. Also, the HKGN (iterative) with hypernetworks reduces up to 23.8% of memory footprint compared with its counterpart removing hypernetworks on FB15k-237.

Table 5. Statistics of the maximum GPU memory allocated (in GB) and training time (in seconds per epoch) of HKGN with/without hypernetworks on FB15k-237 and WN18RR datasets. OOM denotes the Out Of Memory error.

Dataset		FB15k-237 (2-layer)		WN18RR (1-layer)	
Training strategy	Models	Memory	Time	Memory	Time
Iterative	HKGN	14.6	960	2.8	260
	w/o Hyper.	19.2	970	2.9	260
Parallel	HKGN	21.5	80	5.5	68
	w/o Hyper.	OOM		OOM	

5 Conclusion and Future Work

In this paper, we propose HKGN, a novel heterogeneous graph neural network for learning KGE. HKGN introduces hypernetworks to alleviate the problem of the number of heterogeneous parameters in explosive growth, which is fundamental for the model to build a complicated convolution-based message construction function. The developed message function effectively improves prediction accuracy for entities with varying node degrees. As a result, HKGN achieves new SOTA on standard benchmark FB15k-237 and WN18RR. Experimental results show that the proposed hypernetworks significantly reduce the parameter counts and lead to lower GPU memory footprints. In the future, we intend to explore more variants of hypernetworks to make information flow among relations more reasonably. We would also like to combine hypernetworks with more advanced heterogeneous architectures to facilitate KGE learning.

Supplemental Material Statement: Source code for HKGN, detailed hyperparameter settings, datasets FB15k-237 and WN18RR, the data subsets divided by relation categories and entity degrees are available from Github[3].

Acknowledgements. This work was supported by the National Key Research and Development Program of China under Grant 2021YFB3500700.

References

1. Balažević, I., Allen, C., Hospedales, T.M.: Hypernetwork knowledge graph embeddings. In: Proceedings of the ICANN, Munich, Germany, pp. 553–565 (2019). https://doi.org/10.1007/978-3-030-30493-5_52
2. Bansal, T., Juan, D.C., Ravi, S., McCallum, A.: A2N: attending to neighbors for knowledge graph inference. In: Proceedings of the ACL, Florence, Italy, pp. 4387–4392 (2019). https://doi.org/10.18653/v1/P19-1431
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the SIGMOD, Vancouver, BC, Canada, pp. 1247–1250 (2008). https://doi.org/10.1145/1376616.1376746
4. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. In: Proceedings of the EMNLP, Doha, Qatar, pp. 615–620 (2014). https://doi.org/10.3115/v1/D14-1067
5. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of the NIPS, Lake Tahoe, Nevada, United States, pp. 2787–2795 (2013)
6. Cao, Z., Xu, Q., Yang, Z., Cao, X., Huang, Q.: Dual quaternion knowledge graph embeddings. In: Proceedings of the AAAI, pp. 6894–6902 (2021)
7. Das, R., et al.: Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In: Proceedings of the ICLR, Vancouver, BC, Canada (2018)
8. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Proceedings of the AAAI, New Orleans, Louisiana, USA, pp. 1811–1818 (2018). https://doi.org/10.1609/aaai.v32i1.11573
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the NAACL-HIT, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423
10. Dong, X., et al.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the SIGKDD, New York, NY, USA, pp. 601–610 (2014). https://doi.org/10.1145/2623330.2623623
11. Färber, M., Bartscherer, F., Menne, C., Rettinger, A.: Linked data quality of DBpedia, freebase, OpenCyc, Wikidata, and YAGO. Semant. Web **9**(1), 77–129 (2018). https://doi.org/10.3233/SW-170275
12. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of the WWW, New York, NY, USA, pp. 2331–2341 (2020). https://doi.org/10.1145/3366423.3380297

[3] https://github.com/liuxiyang641/HKGN.

13. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the ICML, ICML 2017, pp. 1263–1272. JMLR.org (2017)
14. Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: Proceedings of the ICML, Long Beach, California, USA, vol. 97, pp. 2505–2514 (2019)
15. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. In: Proceedings of the ICLR, Toulon, France (2017)
16. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the NIPS, pp. 1025–1035. Curran Associates Inc., Red Hook (2017)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the CVPR, Las Vegas, NV, USA, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
18. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. In: Proceedings of the Web Conference 2020, New York, NY, USA, pp. 2704–2710 (2020). https://doi.org/10.1145/3366423.3380027
19. Huang, Z., Li, X., Ye, Y., Ng, M.K.: MR-GCN: multi-relational graph convolutional networks based on generalized tensor product. In: Proceedings of the IJCAI, pp. 1258–1264 (2020). https://doi.org/10.24963/ijcai.2020/175
20. Jia, C., Shen, Y., Tang, Y., Sun, L., Lu, W.: Heterogeneous graph neural networks for concept prerequisite relation learning in educational data. In: Proceedings of the NAACL-HIT, pp. 2036–2047 (2021). https://doi.org/10.18653/v1/2021.naacl-main.164
21. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Proceedings of the NIPS, Red Hook, NY, USA, vol. 29 (2016)
22. Jiang, X., Wang, Q., Wang, B.: Adaptive convolution for multi-relational learning. In: Proceedings of the NAACL-HIT, Minneapolis, Minnesota, pp. 978–987 (2019). https://doi.org/10.18653/v1/N19-1103
23. Jin, D., Huo, C., Liang, C., Yang, L.: Heterogeneous graph neural network via attribute completion. In: Proceedings of the Web Conference 2021, pp. 391–400. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3442381.3449914
24. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of the ICLR, San Diego, CA, USA (2015)
25. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the ICLR, Toulon, France (2017)
26. Lehmann, J., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015). https://doi.org/10.3233/SW-140134
27. Li, Z., Liu, H., Zhang, Z., Liu, T., Xiong, N.N.: Learning knowledge graph embedding with heterogeneous relation attention networks. IEEE Trans. Neural Netw. Learn. Syst. 1–13 (2021). https://doi.org/10.1109/TNNLS.2021.3055147
28. Liu, Z., Fang, Y., Liu, C., Hoi, S.C.H.: Node-wise localization of graph neural networks. In: Proceedings of the IJCAI, Montreal, Canada, pp. 1520–1526 (2021). https://doi.org/10.24963/ijcai.2021/210
29. Ma, Y., Crook, P.A., Sarikaya, R., Fosler-Lussier, E.: Knowledge graph inference for spoken dialog systems. In: Proceedings of the ICASSP, South Brisbane, Queensland, Australia, pp. 5346–5350 (2015). https://doi.org/10.1109/ICASSP.2015.7178992

30. Nachmani, E., Wolf, L.: Hyper-graph-network decoders for block codes. In: Proceedings of the NIPS, Vancouver, BC, Canada, pp. 2326–2336 (2019)
31. Nachmani, E., Wolf, L.: Molecule property prediction and classification with graph hypernetworks. Computing Research Repository arXiv:2002.00240 (2020)
32. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. In: Proceedings of the ACL, pp. 4710–4723. Florence, Italy (2019). https://doi.org/10.18653/v1/P19-1466
33. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.: A novel embedding model for knowledge base completion based on convolutional neural network. In: Proceedings of the NAACL-HIT, New Orleans, Louisiana, pp. 327–333 (2018). https://doi.org/10.18653/v1/N18-2053
34. Nguyen, D.Q., Vu, T., Nguyen, T.D., Nguyen, D.Q., Phung, D.: A capsule network-based embedding model for knowledge graph completion and search personalization. In: Proceedings of the NAACL-HIT, Minneapolis, Minnesota, pp. 2180–2189 (2019). https://doi.org/10.18653/v1/N19-1226
35. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the ICML, Bellevue, Washington, USA, pp. 809–816 (2011)
36. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Proceedings of the NIPS, Vancouver, BC, Canada, pp. 8024–8035 (2019)
37. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
38. Riegler, G., Schulter, S., Rüther, M., Bischof, H.: Conditioned regression models for non-blind single image super-resolution. In: Proceedings of the ICCV, Santiago, Chile, pp. 522–530 (2015). https://doi.org/10.1109/ICCV.2015.67
39. Ruffinelli, D., Broscheit, S., Gemulla, R.: You can teach an old dog new tricks! on training knowledge graph embeddings. In: Proceedings of the ICLR, Addis Ababa, Ethiopia (2020)
40. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Proceedings of the ESWC, Heraklion, Crete, Greece, vol. 10843, pp. 593–607 (2018). https://doi.org/10.1007/978-3-319-93417-4_38
41. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: Proceedings of the AAAI, Honolulu, Hawaii, USA, vol. 33, pp. 3060–3067 (2019). https://doi.org/10.1609/aaai.v33i01.33013060
42. Stoica, G., Stretcu, O., Platanios, E.A., Mitchell, T.M., Póczos, B.: Contextual parameter generation for knowledge graph link prediction. In: Proceedings of the AAAI, New York, NY, USA, pp. 3000–3008 (2020). https://doi.org/10.1609/aaai.v34i03.5693
43. Sun, Z., Vashishth, S., Sanyal, S., Talukdar, P., Yang, Y.: A re-evaluation of knowledge graph completion methods. In: Proceedings of the ACL, pp. 5516–5522 (2020). https://doi.org/10.18653/v1/2020.acl-main.489
44. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on CVSC, Beijing, China, pp. 57–66 (2015). https://doi.org/10.18653/v1/W15-4007
45. Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., Talukdar, P.P.: Interacte: improving convolution-based knowledge graph embeddings by increasing feature interactions. In: Proceedings of the AAAI, New York, NK, USA, pp. 3009–3016 (2020). https://doi.org/10.1609/aaai.v34i03.5694

46. Vashishth, S., Sanyal, S., Nitin, V., Talukdar, P.P.: Composition-based multi-relational graph convolutional networks. In: Proceedings of the ICLR, Addis Ababa, Ethiopia (2020)
47. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: Proceedings of the ICLR, Vancouver, BC, Canada (2018)
48. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
49. Wang, P., Agarwal, K., Ham, C., Choudhury, S., Reddy, C.K.: Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. In: Leskovec, J., Grobelnik, M., Najork, M., Tang, J., Zia, L. (eds.) Proceedings of the Web Conference 2021, pp. 2946–2957. ACM/IW3C2, Virtual Event (2021). https://doi.org/10.1145/3442381.3450060
50. Wang, X., et al.: Heterogeneous graph attention network. In: Proceedings of the WWW, San Francisco, CA, USA, pp. 2022–2032 (2019). https://doi.org/10.1145/3308558.3313562
51. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI, Québec City, Québec, Canada, pp. 1112–1119 (2014). https://doi.org/10.1609/aaai.v28i1.8870
52. Xie, Z., Zhou, G., Liu, J., Huang, J.X.: ReInceptionE: relation-aware inception network with joint local-global structural information for knowledge graph embedding. In: Proceedings of the ACL, pp. 5929–5939 (2020). https://doi.org/10.18653/v1/2020.acl-main.526
53. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the ICLR, San Diego, CA, USA (2015)
54. Ye, R., Li, X., Fang, Y., Zang, H., Wang, M.: A vectorized relational graph convolutional network for multi-relational network alignment. In: Proceedings of the IJCAI, Macao, China, pp. 4135–4141 (2019). https://doi.org/10.24963/ijcai.2019/574
55. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of the SIGKDD, New York, NY, USA, pp. 793–803 (2019). https://doi.org/10.1145/3292500.3330961
56. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the SIGKDD, New York, NY, USA, pp. 353–362 (2016). https://doi.org/10.1145/2939672.2939673
57. Zhao, J., Wang, X., Shi, C., Hu, B., Song, G., Ye, Y.: Heterogeneous graph structure learning for graph neural networks. In: Proceedings of the AAAI, pp. 4697–4705. AAAI Press (2021)
58. Zhu, J.-Z., Jia, Y.-T., Xu, J., Qiao, J.-Z., Cheng, X.-Q.: Modeling the correlations of relations for knowledge graph embedding. J. Comput. Sci. Technol. **33**(2), 323–334 (2018). https://doi.org/10.1007/s11390-018-1821-8